**Tesis doctoral**

# Detección de objetos en entornos dinámicos para videovigilancia

Francisco Javier López Rubio

2016


Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

UNIVERSIDAD
DE MÁLAGA

AUTOR: Francisco Javier López Rubio

http://orcid.org/0000-0002-9891-266X

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga


Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

El Dr. D. Ezequiel López Rubio, Profesor Titular de Universidad perteneciente al área de Ciencia de la Computación e Inteligencia Artificial de la E.T.S. Ingeniería Informática de la Universidad de Málaga,

Certifica que,

D. Francisco Javier López Rubio, Ingeniero en Informática, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

# Detección de objetos en entornos dinámicos para videovigilancia

Revisado el presente trabajo, estimo que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en la legislación vigente, autorizo la presentación de este trabajo en la Universidad de Málaga.

Málaga, Marzo de 2016

Fdo.: Dr. Ezequiel López Rubio

*Dedicado a*
*Irene y Miguel*

# Agradecimientos

No puedo ni quiero empezar estos agradecimientos por otra persona más que por mi director de tesis, Ezequiel. Como me consta, otras personas que han trabajado con él han alabado su paciencia, dedicación y disponibilidad. Cualidades todas ellas que sin duda posee. Para mi, mi hermano Ezequiel es sencillamente una persona brillante. Lo es por haber destacado siempre en el campo académico, pero también por ser la luz que ha guiado la mayor parte de mi etapa estudiantil. Él fue el que me animó a comenzar este largo y fructífero camino que concluye con esta tesis. Su conocimiento y pedagogía a la hora de resolver las múltiples dudas que me han surgido, han sido asombrosas y sólo se puede entender hasta qué punto esto es cierto si se ha tenido el privilegio de ser alumno suyo. Me ha hecho ver cuan valioso es el trabajo que he desarrollado, lo que me ha infundido coraje en momentos de abatimiento. Así pues, aunque sé que por más que escriba, las palabras de agradecimiento se quedarán cortas: gracias por todo.

Quiero dar las gracias a mis padres. Han demostrado devoción por todos y cada uno de sus cuatro hijos. Como a mis hermanos, me han procurado siempre los medios materiales durante la mayoría de mi etapa estudiantil y sin ese comienzo, nada de esto hubiera sido posible. Mi padre ha sido un referente al haber demostrado con su esfuerzo y dedicación que era posible completar unos estudios superiores aún cuando las circunstancias eran más complejas que las que yo he vivido. Mi madre ha mostrado un gran interés por mi trabajo y lo ha seguido de cerca, esto me ha estimulado para seguir adelante. Ambos me han enseñado a valorar el conocimiento en sus múltiples formas, tanto con sus palabras como con sus obras, y sin esto es posible que no me hubiera animado a invertir todo el tiempo que he dedicado a esta tesis.

Debo agradecer a Luis, actual jefe de informática de la delegación de turismo de Málaga y antiguo compañero de trabajo, su interés por mi investigación, lo cual ha sido sin quererlo, un apoyo y una motivación más. Como todos sus compañeros de trabajo saben, es un ejemplo de responsabilidad y buen hacer. Por lo tanto, haber compartido unos años de trabajo con él me ha servido para valorar aún más el trabajo bien hecho y esforzarme por realizar esta tesis de forma correcta y completa. Sé que si puede se la leerá, lo cual agradezco. Quiero que sepa que admiro y respeto su incansable sed de conocimiento.

También deseo dar las gracias a Irene, mi esposa. Ella ha sido mi compañera en este largo viaje. Ha estado ahí cuando los frutos tardaban en llegar más de lo esperado, me ha apoyado cuando lo necesitaba y se ha interesado todo lo que ha podido. Ha

sabido tener la suficiente paciencia como para no dejar que abandonase. Además, sé que valora el trabajo que he realizado y esto me ha reconfortado mucho cuando la meta aún no se veía.

Por último, quiero dedicarle unas palabras a mi hijo Miguel. Él ha venido a este mundo cuando estaba a mitad de mi investigación. Desearía que cuando tenga algunos años, eche un vistazo a lo que su padre escribió mientras él era todavía un bebé y comprenda que con esfuerzo y dedicación se pueden alcanzar metas que uno ni siquiera se imagina cuando se es más joven. Quiero que sepa que este camino ha sido largo, pero al mirar atrás reconozco que ha sido estimulante, por lo que le animo a que afronte todos los retos que le puedan surgir en su vida con perseverancia y optimismo.

# Índice general

# 1 Introducción

El mundo actual demanda cada vez más información sobre el entorno que le rodea. La necesidad de seguridad y control ha motivado la proliferación de sistemas que tratan de extraer información en diferentes ámbitos. En particular ha surgido una gran cantidad de sistemas de videovigilancia que proporcionan información visual sobre áreas de interés. Estos sistemas suelen ser baratos de adquirir e instalar, además son capaces de funcionar de manera ininterrumpida monitorizando una amplia variedad de entornos, como por ejemplo: vías de comunicación, el perímetro o el interior de edificios, plazas, centros de ocio, etc.

El uso tradicional de estos sistemas ha consistido, bien en la vigilancia manual por parte de operadores humanos, bien en el almacenamiento en archivos de los datos recopilados para que, en caso necesario, ser visionados posteriormente. Sin embargo estos enfoques presentan numerosos inconvenientes, especialmente si se trata de reaccionar en tiempo real a lo que se está monitorizando. Los objetos observados pueden ser muy diversos y en algunos casos tanto su complejidad como su número son elevados. En general el gran volumen de datos que se genera hace que el tratamiento manual sea costoso, ineficiente e incluso impracticable en casos especialmente complejos. Así por ejemplo, un operador humano difícilmente puede controlar varios monitores a la vez, además es susceptible de sufrir distracciones o ser abrumado en presencia de muchos objetos en movimiento. El almacenamiento y posterior visionado en ocasiones puede necesitar el conocimiento de un hecho relevante, lo que no siempre es posible y además, no evita que la gran cantidad de información almacenada pueda ser sea poco útil y difícil de procesar. Es por todo ello que el procesamiento automático de estos datos es deseable. Tanto es así que en los últimos años se ha convertido en una área de conocimiento especialmente relevante, como avalan la multitud de publicaciones que versan sobre ella.

La videovigilancia por medios automáticos no está exenta de dificultades. Las propuestas existentes sufren limitaciones que las hacen propensas a cometer errores. En este sentido las situaciones en las que deben funcionar correctamente son muy variadas, por lo que las soluciones deben de ser suficientemente robustas para que el resultado final del tratamiento no sea perjudicado. Esta tesis aborda varias de estas dificultades y plantea nuevos enfoques que, en la gran mayoría de las ocasiones, superan a otras propuestas pertenecientes al estado del arte.

## 1.1. Contexto

La videovigilancia consiste en la vigilancia a través de un sistema de cámaras. En la segunda mitad del siglo XX se empleaban cámaras analógicas conectadas mediante circuito cerrado a un televisor, de ahí que se les conozca también como CCTV (*Closed-circuit television*). Actualmente la industria ha evolucionado gracias a la investigación y abaratamiento de las tecnologías. El uso generalizado de tecnologías de comunicación entre dispositivos electrónicos ha hecho posible la conexión de las cámaras de vigilancia mediante IP, surgiendo así las denominadas cámaras IP que permiten entre otras cosas su control remoto o también las cámaras inalámbricas, que prescinden de una conexión cableada. De la misma manera son habituales cámaras con modernos sensores que graban vídeos de alta calidad, infrarrojos o panorámicos; siendo también una opción posible el uso de cámaras PTZ (*Pan-Tilt-Zoom* por sus siglas en inglés) que permiten abarcar una área mayor gracias a que posibilitan el giro y zoom de la misma.

Desde el punto de vista de la informática la videovigilancia automática consiste en procesar los datos captados por las cámaras para extraer información útil para el usuario. En función del objetivo buscado el procesamiento puede ser de un nivel más o menos básico. El nivel más básico es la **detección de movimiento,** el cual es el objeto de estudio de esta tesis y que consiste en diferenciar qué píxeles forman parte de los objetos en movimiento, es decir, del primer plano. Recíprocamente, este problema puede verse cómo la identificación de los píxeles que componen el fondo. Niveles superiores son la identificación de objetos, el seguimiento de los mismos, el reconocimiento de acciones y por último, el análisis de comportamiento.

Hoy día la videovigilancia puede utilizarse prácticamente en cualquier situación donde haya una zona que necesite ser vigilada o controlada. El tratamiento manual de las cámaras sigue siendo habitual especialmente en sistemas pequeños. No obstante existen multitud de casos reales en los que se utiliza un software capaz de analizar, indexar y buscar información automáticamente dentro de la enorme cantidad de datos que gestionan, por ejemplo identificando personas y vehículos o detectando determinados comportamientos de los objetos que hay en la escena. En la actualidad grandes ciudades por todo el mundo cuentan con una infraestructura de cámaras que les permite supervisar las áreas más importantes de las mismas o incluso en algunos casos monitorizando la práctica totalidad de las calles como el proyecto *Operation Virtual Shield* de Chicago. Tanto el perímetro como el interior de los edificios, así como los medios de transporte públicos son habitualmente vigilados con cámaras. Por otro lado un uso muy extendido de la videovigilancia son los SIT (Sistemas Inteligentes de Transportes) para el control del tráfico. La DGT en España y ciudades como París, Berlín, Madrid, Barcelona o Granada llevan años contando con este tipo de sistemas que ayudan a gestionar, vigilar y regular el tráfico de forma automática. Además de esto existen proyectos comerciales como el *Smart Surveillance Solution* de IBM o DEPA de Sony que ofrecen soluciones integrales para el tratamiento automatizado de la videovigilancia.

**Figura 1.1:** Ejemplos de cámaras de videovigilancia. De izquierda a derecha: cámaras en el tejado de un edificio, cámaras de tráfico en una autovía y cámara inalámbrica en el interior de un edificio. *Fuente: Wikipedia.*

Existen otras situaciones en las que los fundamentos de la videovigilancia automatizada pueden aplicarse para fines distintos de los vistos anteriormente. Así por ejemplo los vídeos que se encuentran contenidos en Internet en sitios como Youtube o Flickr son cada vez más numerosos. En este sentido es deseable extraer automáticamente metadatos que permitan indexar esta información y realizar búsquedas más precisas [1]. El reconocimiento gestual es una característica que ya tienen algunos dispositivos domésticos y que se plantea extender a otros entornos como los automóviles. En gran medida la tecnología desarrollada para videovigilancia es aplicable en ambos casos [2, 3].

En los últimos años se han realizado un gran número de publicaciones científicas sobre videovigilancia [4, 5, 6]. La mayoría de ellas se centran en alguna de las etapas en las que se puede dividir la videovigilancia automática, tratando de mejorar las propuestas ya existentes. Debido a que no existe una solución global al problema de la videovigilancia, existe una importante producción científica que trata de solucionar problemas y retos cada vez más ambiciosos.

## 1.2. Objetivos

El objetivo de esta tesis es el estudio de la detección automática de objetos en entornos dinámicos. Para cumplir este objetivo se persiguen los siguientes objetivos específicos:

- Estudiar el modelado y segmentación de fondo. La detección de objetos en movimiento es clave ya que, al ser la primera etapa, dependen de ella el resto de etapas y en buena medida el resultado final del procesamiento automático.
- Analizar situaciones y entornos típicamente complejos durante la segmentación de fondo. Se han utilizado conjuntos de datos en los que los métodos de segmentación de fondo tienen dificultades con el fin de estudiar dichas limitaciones. Las situaciones estudiadas han sido básicamente:
    - Ruido durante la adquisición de vídeo.

  - Fondos de escena dinámicos.
  - Cambios de iluminación.
  - Cámara en movimiento.

- Realizar un estudio crítico del rendimiento de los métodos de segmentación de fondo existentes ante condiciones adversas. En este sentido hemos utilizado algoritmos de referencia tanto de segmentación de fondo en general, como específicos de la problemática concreta que se ha estudiado en cada caso.
- Proponer, diseñar y desarrollar nuevos algoritmos de segmentación de fondo que obtengan mejores resultados que otras propuestas ya existentes.
- Extender y complementar la funcionalidad de los algoritmos de segmentación de fondo para en su caso ser capaces de afrontar situaciones más complejas.

## 1.3. Metodología

Para el diseño e implementación de nuevos métodos de segmentación de fondo seguiremos los siguientes principios metodológicos:

- Número de parámetros a ajustar. Se pretende que el número de parámetros a ajustar sea el menor posible, de modo que el conjunto de configuraciones óptimas sea igualmente reducido.
- Dificultad de implementación. Cuanto más fácil sea de implementar el algoritmo, más se acortará el diseño de sistemas reales que lo utilicen.
- Requisitos de tiempo real. La mayoría de los sistemas de videovigilancia necesitan tomar decisiones en tiempo real. Por ello se ha prestado especial atención a que los algoritmos implementados no sólo obtengan buenos resultados respecto de otras propuestas, si no que además sean capaces de trabajar en tiempo real.

Para el estudio del rendimiento de los métodos propuestos tendremos los siguientes principios metodológicos:

- Comparación con métodos de segmentación de fondo existentes y pertenecientes al estado del arte. Se trata de ver si los métodos propuestos dan mejores resultados que los demás para las medidas de rendimiento establecidas. Se ha evitado en la medida de lo posible utilizar resultados ya existentes para realizar las comparaciones entre métodos con el fin de que las comparaciones sean los más justas posibles, controlando que no se realice pre-procesado de los datos de entrada ni post-procesado de la salida. Hemos simulado los métodos de otros autores usando códigos y ejecutables proporcionados por dichos autores o bien, cuando esto no ha sido posible, propuestas de código abierto disponibles en Internet.

- Evaluación del rendimiento. Se busca establecer claramente las medidas de rendimiento empleadas para comparar los resultados obtenidos de forma objetiva.
- Conjunto de configuraciones probadas. Se ha buscado que el conjunto de configuraciones probado en los métodos competidores sea igual o mayor que el utilizado para nuestras propuestas con el fin de que las comparaciones sean lo más justas posibles. Así mismo han sido empleadas preferentemente aquellas configuraciones recomendadas por los autores de los métodos estudiados.
- Conjuntos de datos para pruebas. En ocasiones los métodos existentes para la segmentación de fondo utilizan conjuntos de datos propios para las pruebas, lo cual dificulta la comparación de sus resultados con otras propuestas. Con el fin de evitar esta situación hemos empleado conjuntos de datos variados, públicos y conocidos que faciliten futuras comparaciones. De la misma manera, en los casos en los que ha sido necesario generar nuevos datos para las pruebas, estos se han subido a Internet para su libre disponibilidad.
- Reproducibilidad de las pruebas realizadas. Hemos hecho hincapié en que todas las pruebas realizadas para sustentar nuestro estudio sean reproducibles, con este fin se han subido a Internet los códigos plenamente funcionales de nuestras propuestas. Además los parámetros empleados tanto por nuestros métodos como por los competidores han sido igualmente publicados.

## 1.4. Estructura de la tesis

La presente tesis se estructura en un primer bloque de antecedentes, otro bloque formado por cinco capítulos en los que se desarrollan estudios y propuestas relacionadas con la detección de movimiento y por último, un capítulo de conclusiones y trabajos futuros.

El primer gran bloque es el que constituye el Capítulo 2 de antecedentes, en él se estudian los elementos fundamentales que intervienen en la videovigilancia automática en general y en la detección de movimiento en particular. Su importancia es clave porque nos proporciona la base sobre la que desarrollar el resto de la tesis. En la Sección 2.1 se exponen conceptos tan importantes como el espacio de color, haciendo una revisión de aquellos que son utilizados en el ámbito del procesamiento de imágenes y centrándonos en los que se usarán a lo largo de la tesis. En la Sección 2.2 se define una amplia variedad de conceptos relacionados con cada una de las etapas en las que se puede subdividir la videovigilancia automática, mostrando ejemplos básicos de métodos y modelos que corresponden a cada una de ellas. En la Sección 2.3 se profundiza en la etapa de la videovigilancia en la que se centra esta tesis, es decir, en la detección de movimiento. Aquí se exponen las dificultades habituales a las que se enfrentan los métodos de detección, lo cual servirá de guía cuando hablemos de los problemas que se están observando en las pruebas llevadas a cabo.

Además se hace una extensa revisión de los métodos de detección de movimiento de referencia, muchos de los cuales se usarán en varias ocasiones en los experimentos que realizaremos. Para complementar esta sección, se habla de post-procesado, que si bien no ha sido aplicada en los resultados finales, puede ser útil como última etapa de la detección de movimiento. La Sección 2.4 es fundamental ya que en ella se definen las medidas que se usan en la literatura para medir el rendimiento de los métodos de detección de movimiento. Por tanto, serán las que utilicemos para comparar los resultados de las diferentes alternativas que pongamos a prueba. La Sección 2.5 versa sobre el modelado del fondo mediante aproximación estocástica, en ella se expone un modelo que será utilizado en varias ocasiones, bien como base para proponer nuevos métodos, bien como modelo de ejemplo con el que comparar otros métodos de detección. Para concluir este capítulo (Sección 2.6), se presenta la biblioteca BGS, la cual ha sido utilizada en repetidas ocasiones debido a que implementa una gran cantidad de métodos de detección pertenecientes al estado del arte, muchos de los cuales compararemos con nuestras propuestas. En particular en esta sección se hace un repaso de las diferencias que hay entre la implementación de los métodos que ofrece dicha biblioteca y la definición de los métodos en los artículos originales.

Los cinco capítulos siguientes (Capítulos 3-7) están dedicados a estudiar o resolver diferentes problemas relacionados con la detección de movimiento. Como todo procesamiento, la detección de los objetos en movimiento puede subdividirse en tres etapas: entrada, proceso (detección) y salida. A grandes rasgos, en esta tesis hemos hecho hincapié en las dos primeras, es decir, en la entrada y en el proceso.

Los Capítulos 3 y 4 se centran en las características de la entrada, estudiando cómo afecta el ruido y el espacio de color a los algoritmos de detección existentes. En la Sección 3.1 se exponen los efectos que produce el ruido durante la adquisición, registro y transmisión en una imagen o vídeo. A priori esto supone un problema para el procesamiento de la imagen. En concreto, en la Sección 3.2 se presenta un conjunto de experimentos que tratan de poner de relieve el efecto que produce en la detección de movimiento que, como veremos, no siempre es el esperado. Para ello se ponen a prueba varios métodos de detección actuales en diferentes situaciones frente a varios niveles de ruido.

El Capítulo 4 versa sobre la influencia de los espacios de color en la detección. En la Sección 4.1 se explica brevemente que hasta ahora la elección del espacio de color ha sido importante de cara a lograr un buen rendimiento de los métodos, sin embargo este aspecto no se ha explotado lo suficiente. Por ello, se propone un marco de trabajo común que facilita la elección del espacio de color más adecuado para codificar el color de la secuencia de entrada de manera que se optimice la detección de movimiento. Este marco será aplicado a un modelo probabilístico basado en redes neuronales, el cual se define en la Sección 4.2. La Sección 4.3 presenta el marco común que permite evaluar la relevancia de las componentes de color de cara a la detección y además, en esta misma sección se usa dicho marco para adaptar el modelo base referido previamente. La importancia de los espacios y componentes de color queda

acreditada en la Sección 4.4, donde se definen experimentos con varios espacios de color, diferentes ponderaciones de las componentes de color y distintas secuencias de prueba.

Los dos capítulos siguientes (Capítulos 5 y 6) se centran en la etapa de proceso. Se exponen sendas propuestas que atenúan los efectos negativos de varios problemas que afectan habitualmente a la detección de movimiento. El primero de ellos, propone un método nuevo capaz de extraer y manejar rasgos del vídeo de entrada distintos de los que se usan habitualmente. En la Sección 5.2 se define el modelo que emplea. No está restringido a utilizar las componentes de color, siendo capaz de utilizar cualquier número de rasgos de entrada y no sólo tres, como en el caso del color. En la Sección 5.3 se proponen nuevos rasgos y se lleva a cabo un estudio sobre el rendimiento que se consigue al utilizar diferentes combinaciones de ellos. Para sustentar la relevancia del modelo propuesto se ha diseñado y ejecutado un conjunto de experimentos en los que se utilizan secuencias de diverso tipo (Sección 5.4). Además, los resultados son comparados con los obtenidos por varios métodos pertenecientes al estado del arte.

El Capítulo 6 aborda la problemática que suponen los cambios de iluminación para la detección de movimiento. En la Sección 6.1 se justifica porqué es un problema y se realiza una revisión de las propuestas actuales que tratan de solucionarlo. Por nuestra parte, proponemos un método de detección de cambios de iluminación (Sección 6.2) que se debe integrar en un método de detección existente, con el objetivo de que el rendimiento en este tipo de situaciones sea mayor que el obtenido sin su ayuda. Para saber hasta qué punto esto es así, se elige un conjunto de secuencias caracterizadas por cambios de iluminación de diferente tipo y se compara el rendimiento de los métodos modificados con nuestra propuesta tanto con los métodos sin modificar, como con varios métodos diseñados específicamente para la detección de cambios de iluminación (Sección 6.3).

En los capítulos comentados previamente se asume que la cámara de videovigilancia es estática. Sin embargo, en la primera parte del Capítulo 7, se expone que cada vez la necesidad de detectar movimiento en secuencias grabadas con cámaras no estáticas es mayor. En este capítulo se explica porqué el enfoque tradicional para detectar movimiento no es válido. En la Sección 7.1 se lleva a cabo una revisión de las propuestas que abordan este problema. Nosotros planteamos un modelo para la detección de movimiento en secuencias grabadas con cámaras no estáticas que no ha sido considerado hasta ahora (Sección 7.2). Para estudiar si realmente esta propuesta es competitiva o no, se realiza una comparativa con varios métodos actuales diseñados para detectar movimiento en este tipo de secuencias y, como es habitual, se usa un amplio conjunto de secuencias de prueba (Sección 7.3).

Por último, en el Capítulo 8 se exponen las conclusiones finales de esta tesis y se dibujan las líneas de investigación a seguir en trabajos futuros.

# 2 Antecedentes

En este capítulo se exponen los conceptos, algoritmos y modelos matemáticos básicos que serán empleados a lo largo de la tesis. En la Sección 2.1 se explica qué es un espacio de color y qué tipos de espacios se emplean habitualmente para el almacenamiento y procesamiento de imágenes. La Sección 2.2 versa sobre los conceptos básicos de la videovigilancia automática y también se describen cada una de las etapas en las que puede dividirse. La Sección 2.3 profundiza en la etapa de detección de movimiento, en particular muestra los tipos de modelos de fondo de referencia en la actualidad. La manera de medir el rendimiento de los métodos de detección de movimiento se estudian en la Sección 2.4. En la Sección 2.5 se expone el modelado de fondo mediante aproximación estocástica, el cual será utilizado en varias ocasiones a lo largo de la tesis. Por último, la Sección 2.6 versa sobre la biblioteca BGS la cual ha sido básica para probar los métodos de detección de movimiento ya existentes.

## 2.1. Espacios de color

Las imágenes digitales están compuestas por un conjunto de píxeles en los que cada uno de ellos tiene un color determinado. De cara a la interpretación de la escena, el estudio del color de los píxeles es clave ya que es lo que define cómo cambia la escena. Como veremos, en función de la manera de representar internamente el color podremos usar herramientas que nos permiten analizar de forma objetiva algunas características de la imagen. Así pues, la forma de representar el color debe ser tenida en cuenta tanto para definir el problema como para resolverlo.

El color es la interpretación que el sistema visual humano realiza de la luz percibida. La luz es una radiación electromagnética y el ser humano es sensible a unas longitudes de onda de entre 380 y 700nm. En el ojo humano existen dos clases de células fotosensibles, los conos y los bastones. Los bastones son útiles en condiciones de baja luminosidad, pero no detectan colores. Los conos son los que intervienen en la detección del color. Existen tres tipos de conos, denominados L, M y S. Cada uno de estos es sensible a longitudes de onda largas (tipo L), medias (M) y cortas (S). De manera aproximada los conos de tipo S son más sensibles al color azul, los tipo M al verde y los tipo L al rojo. La teoría tricromática de Young-Helmholtz sostiene que la estimulación de estos receptores es lo que el cerebro interpreta como color. Por ello para representar el color es necesario y suficiente emplear tuplas de tres componentes que representan puntos en un espacio tridimensional. En ocasiones, dichas componentes también se denominan canales o canales de color.

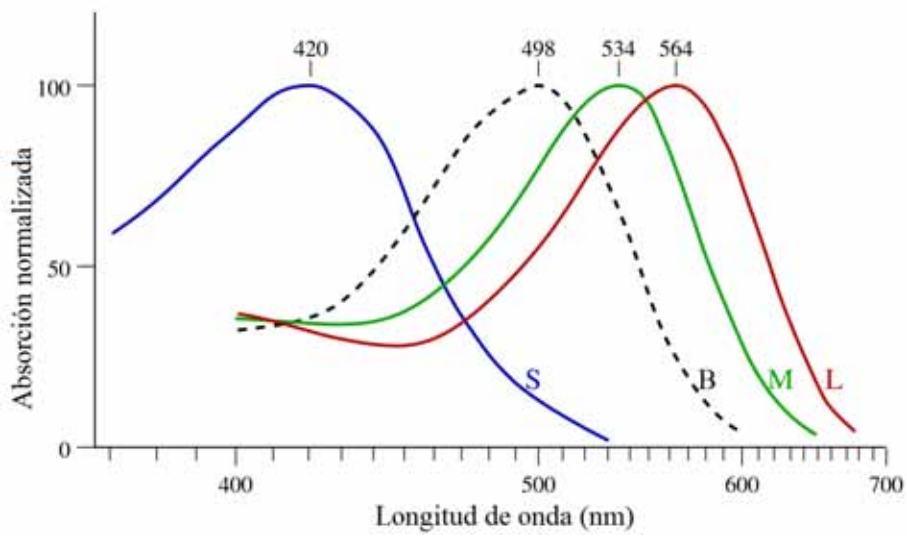



**Figura 2.1:** Sensibilidad de las células fotosensibles del ojo (bastones y conos) en función de la longitud de onda de la luz incidente. En color azul los conos tipo S, en verde los tipo M y en rojo los tipo L. En color negro discontinuo para los bastones. *Adaptado de Wikipedia.*

En los sistemas digitales el color necesita ser representado de alguna manera que permita ser procesado, almacenado y transmitido. Para este fin se utilizan los espacios de color. Un espacio de color está compuesto por un conjunto de colores y una función que asigna un color a una tupla. Como hemos dicho, para representar el color se suelen utilizar tuplas de tres componentes, por lo que la mayoría de los espacios de color suelen tener tres dimensiones.

Las componentes del color codificadas en sistemas digitales tienen una precisión determinada debido a la naturaleza discreta de los propios dispositivos, por lo que solamente es posible representar un conjunto limitado de colores. El significado del valor de cada componente de una tupla puede ser más o menos intuitivo, por lo que expresar numéricamente un color puede ser complicado para una persona. En ocasiones es deseable establecer una medida cuantitativa de lo parecidos que son dos colores visualmente. Por todo ello, se han definido distintos espacios de color atendiendo a estas y a otras limitaciones o necesidades.

Desde el punto de vista de la visión por computador una clasificación de los espacios de color podría ser la siguiente:

- **Espacios de color independientes del hardware**. El espacio de color es especificado independientemente de las características del dispositivo o aplicación donde se vaya a utilizar. Este tipo de espacios es útil cuando es necesario referirnos inequívocamente a un color en concreto, por ejemplo para realizar comparaciones de colores entre diferentes dispositivos.
- **Espacios de color basados en el usuario**. Estos espacios de color están

inspirados en las características del sistema visual humano. El color se especifica, o bien mediante atributos que son fácilmente comprensibles por el ser humano como el tono, la saturación y el brillo, o bien basándose en características fisiológicas como son el nivel de color rojo, verde y azul, por analogía con los tipos de fotorreceptores del ojo humano.

- **Espacios de color dependientes del hardware**. Algunos dispositivos tienen limitaciones a la hora de captar, procesar o reproducir colores. En estos casos es necesario definir espacios de color que se ajusten a estas características para así poder trabajar con ellos.

Cabe destacar que esta clasificación es no excluyente, por lo que existen espacios de color que están en dos categorías. A modo de ejemplo sRGB estaría incluido tanto en espacios de color independientes del hardware como en los basados en el usuario. En la literatura se pueden encontrar otras clasificaciones de los espacios de color [7, 8]. En esta tesis seguiremos la clasificación anterior para guiar la descripción de los espacios de color más conocidos e importantes. Destacaremos en cada caso los aspectos más característicos de cada espacio.

### 2.1.1. Espacios de color independientes del hardware

Algunos espacios de color como sRGB, Adobe RGB o Pantone son independientes del hardware en el sentido de que determinan inequívocamente un color, sin embargo no representan todos los colores existentes, están ideados únicamente para ser utilizados por ciertos dispositivos. Los espacios de color definidos por la Comisión Internacional de la Iluminación (CIE) no solo determinan inequívocamente cada color, si no que además modelan todos los colores existentes. Esta es la razón por la que entre otras cosas son utilizados como referencia por la mayoría de los demás espacios de color [8]. A continuación vamos a describir cada uno de los espacios de color definidos por la CIE.

#### 2.1.1.1. CIE XYZ

En 1931 la CIE definió matemáticamente unas curvas del color que permiten transformar en un tupla de tres componentes la luz incidente en el ojo de un hipotético observador estándar. Estas tres componentes, conocidas como valores triestímulo (*tristimulus values*) y denotadas como XYZ, determinan inequívocamente cualquier color visible por el ser humano bajo unas condiciones del entorno específicas [9].

La componente Y es aproximadamente una medida del brillo o luminosidad del color, siendo X y Z las componentes cromáticas. En la práctica para especificar un color suelen usarse los valores normalizados de XYZ, dando lugar al denominado espacio de color CIE xyY:

$$x = \frac{X}{X + Y + Z} \tag{2.1}$$

$$y = \frac{Y}{X + Y + Z} \tag{2.2}$$

$$z = \frac{Z}{X + Y + Z} \tag{2.3}$$

donde $x + y + z = 1$.

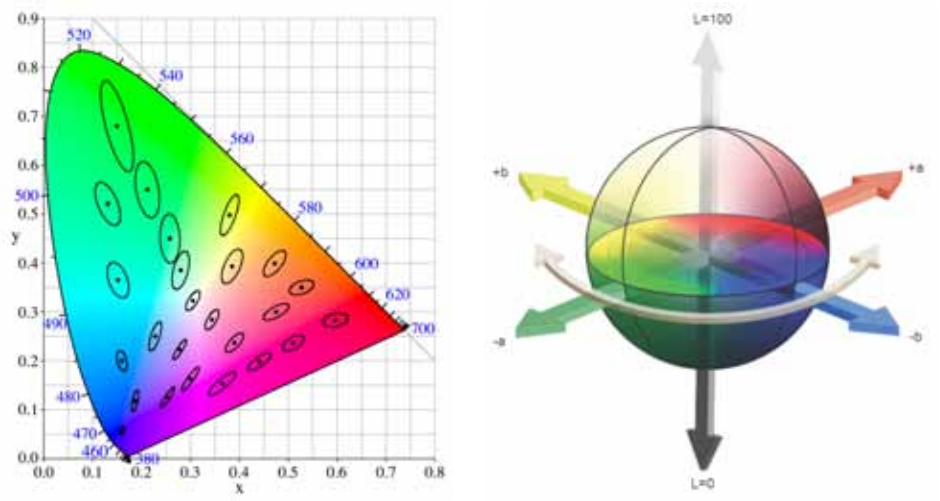
La normalización del espacio de color CIE XYZ permite dibujar el denominado diagrama de cromaticidad CIE, como se muestra en la Figura 2.2. Este diagrama es una representación bidimensional del espacio CIE xyY en ausencia de brillo. El valor de $z$ es calculado implícitamente mediante $z = 1 - (x + y)$. La curva exterior se conoce como locus espectral y se corresponde con el color que tendría una luz monocromática de las longitudes de onda marcadas, estos colores también se llaman colores espectrales. Aproximadamente en el punto $(x, y) = (1/3, 1/3)$ nos encontramos con el color blanco para ese nivel de brillo. $D_{65}$ también conocido como blanco de referencia fue definido por la CIE, es aproximadamente el color de la luz del día. No obstante cabe puntualizar que esta representación solo nos permite tener una idea aproximada de la gama de colores o gamut que existe para un brillo dado, ya que existen factores como la luz con la que se está visualizando el diagrama y las características del medio que los reproduce, por ejemplo un monitor o una impresora, que condicionan los colores que realmente están siendo percibidos por el ojo.

Una de las conclusiones que se pueden obtener del diagrama de cromaticidad CIE es que usando tres colores primarios arbitrarios no es posible representar todos los colores visibles, sin embargo para todos los colores visibles existen tres colores primarios que los representan [7]. Esto se puede ver gráficamente en el diagrama CIE si representamos un triángulo donde los vértices del mismo son los colores primarios elegidos, los colores que hay dentro del triángulo se pueden representar con los colores primarios elegidos, pero no los que quedan fuera.

En el diagrama se observa que los colores no se distribuyen uniformemente. Por ejemplo, en la zona con valores de $y$ grandes, hay áreas donde la diferencia visual entre colores cercanos es pequeña, mientras que en la zona correspondiente a valores de $y$ pequeños, las diferencias entre colores cercanos es elevada. Esta situación es estudiada formalmente mediante las elipses de MacAdam [10], las cuales determinan áreas del espacio de color donde los colores son visualmente indistinguibles. Si el espacio fuera uniforme, todas las elipses de MacAdam serían círculos, situación que no se da en el caso de este espacio de color (ver elipses en la Figura 2.2).

La principal ventaja del espacio de color CIE XYZ es que modela el conjunto de todos los colores visibles por el sistema visual humano, lo que ha significado que sea utilizado ampliamente como referencia para el resto de espacios de color. Por

contra, el hecho de que sea altamente no uniforme lo hace poco práctico para la visión por computador, en particular a la hora de realizar comparaciones numéricas entre diferentes colores. Además el significado de las componentes de un color no es especialmente intuitivo, dificultando su uso por parte de las personas.




(a) (b)

**Figura 2.2:** En (a) se muestra el diagrama de cromaticidad CIE y las elipses de MacAdam, (b) es una representación gráfica del espacio de color CIELAB. *Fuente: Wikipedia y www.lacie.com, respectivamente.*

#### 2.1.1.2. CIELAB y CIELUV

Como acabamos de comentar, el espacio de color CIE XYZ y su derivado CIE xyY tienen la característica de que dos colores espacialmente lejanos pueden ser muy parecidos visualmente, de la misma manera algunos colores cercanos pueden ser muy diferentes, esto se conoce como no uniformidad del espacio de color. Para evitar esto, en 1976 la CIE desarrolló dos espacios de color basados en CIE XYZ: el CIELAB ($L^*a^*b^*$) y el CIELUV ($L^*u^*v^*$). Al igual que CIE XYZ representan cualquier color visible, pero con la particularidad de que son modelos perceptualmente uniformes. Para lograr esta propiedad existen dos restricciones fundamentales a tener en cuenta:

- Adaptación cromática.
- Respuesta visual no lineal.

Gracias a la uniformidad del espacio de color, las parejas de colores visualmente parecidos tienen una distancia entre ellos pequeña al usar la distancia euclídea, mientras que si los colores son muy diferentes, la distancia también es grande. Esto hace que la distancia proporcione una medida ajustada de lo que el ser humano percibe como colores más o menos diferentes.

Desde el punto de vista de la visión por computador la uniformidad perceptual puede ser muy útil porque permite tener una medida fácilmente calculable de lo diferentes que son dos colores. Lo cual es útil por ejemplo si queremos saber si un conjunto de píxeles adyacentes forman parte de la misma región.

Ambos espacios de color están basados en la iluminación percibida $L^*$ y dos componentes cromáticas opuestas, aproximadamente rojo-verde frente a amarillo-azul (ver Figura 2.2). Gracias a esta característica, además de poder representar cualquier color, obtenemos dos ventajas adicionales:

- Dado un par de componentes cromáticas, podemos obtener versiones más o menos iluminadas de un mismo color modificando únicamente la componente $L^*$.
- Dado un par de colores aparentemente diferentes podemos saber si realmente uno es una versión oscurecida o iluminada del otro ignorando la componente de iluminación.

CIELAB y CIELUV son similares, de hecho la componente de iluminación $L^*$ es idéntica en ambos casos, sin embargo el modelo de adaptación cromática empleado es diferente. Los dos normalizan sus valores cromáticos empleando el blanco de referencia, pero CIELAB utiliza la división, mientras que CIELUV emplea la diferencia. La transformación entre el espacio de color CIE XYZ y CIELAB viene dada por las siguientes ecuaciones:

$$L^* = 116 f\left(\frac{Y}{Y_n}\right) - 16 \tag{2.4}$$

$$a^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right] \tag{2.5}$$

$$b^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right] \tag{2.6}$$

$$f(x) = \begin{cases} x^{1/3} & x > 0,008856 \\ 7,787x + \frac{16}{116} & en\, otro\, caso \end{cases} \tag{2.7}$$

siendo $(X_n, Y_n, Z_n)$ el valor triestímulo del blanco de referencia.

Para obtener las componentes cromáticas del espacio CIELUV a partir del espacio CIE XYZ se deben utilizar las siguientes ecuaciones:

$$u^* = 13L^*(u' - u'_n) \tag{2.8}$$

$$v^* = 13L^*(v' - v'_n) \tag{2.9}$$

donde los valores intermedios $u', u'_n, v'$ y $v'_n$vienen dados por:

$$u' = \frac{4X}{X + 15Y + 3Z} \tag{2.10}$$

$$u'_n = \frac{4X_n}{X_n + 15Y_n + 3Z_n} \tag{2.11}$$

$$v' = \frac{9Y}{X + 15Y + 3Z} \tag{2.12}$$

$$v'_n = \frac{9Y_n}{X_n + 15Y_n + 3Z_n} \tag{2.13}$$

El rango de valores posibles para $L^*$ va desde el 0 al 100 que son, respectivamente, el negro y el blanco de referencia. Mientra que las componentes cromáticas tanto de CIELAB como de CIELUV varían aproximadamente desde el -100 al 100.

La magnitud de la diferencia perceptual entre dos colores viene dada por la distancia euclídea de dos puntos en un espacio tridimensional:

$$\delta^e\left(a^*, b^*\right) = \left[\left(\Delta L^*\right)^2 + \left(\Delta a^*\right)^2 + \left(\Delta b^*\right)^2\right]^{1/2} \tag{2.14}$$

$$\delta^e\left(u^*, v^*\right) = \left[\left(\Delta L^*\right)^2 + \left(\Delta u^*\right)^2 + \left(\Delta v^*\right)^2\right]^{1/2} \tag{2.15}$$

Para ser exactos estas fórmulas no siempre dan una medida ajustada de la diferencia perceptual de colores. En concreto, cuando los valores de cromaticidad son extremos, las diferencias en los ejes principales de los elipsoides de MacAdam pueden llegar a ser de 1:6 [11], por lo que realmente no es un espacio completamente uniforme. A raíz de esta situación la CIE recomendó en 1994 unas nuevas medidas de distancia. No obstante, para cumplir los objetivos de esta tesis consideraremos que es un espacio uniforme, por lo que será suficiente el uso de la distancia euclídea.

### 2.1.2. Espacios de color basados en el usuario

Los colores basados en el usuario son aquellos que han sido definidos tratando de asemejarse a la manera en la que el ser humano percibe o entiende el color. En este sentido, existen básicamente espacios de color basados, bien en cómo procesa el color el sistema visual humano, bien en cómo describir los colores de manera intuitiva. A continuación hablaremos de los espacios RGB y Ohta como representantes del primer tipo y de los espacios HSV y HSL para el segundo tipo.

#### 2.1.2.1. RGB

La teoría tricromática sostiene que la luz que incide en las células fotosensibles del ojo (conos), genera un estímulo en ellas que el cerebro interpreta directamente como un color concreto. De forma análoga se define el modelo de color RGB. El color se modela simulando los tres tipos de conos existentes: S, M y L. Dado que cada tipo de cono se corresponde aproximadamente con el rojo, verde y azul, en el modelo RGB se utilizan estos tres colores primarios para especificar el color.

RGB es un modelo de color aditivo en el que la suma de las magnitudes de cada una de las componentes es lo que determina el color. De esta manera si utilizamos un espacio normalizado en el que el máximo de cada componente es 1 y el mínimo 0, obtenemos un cubo de lado 1 en que el color negro se corresponde con el valor $(0, 0, 0)$, el blanco es $(1, 1, 1)$, la escala de grises es la línea que une ambos puntos, mientras que las esquinas son los colores primarios y saturados (ver Figura 2.3).

Este modelo define una manera de representar el color, sin embargo no especifica qué colores exactamente son los colores primarios. Como ya hemos visto, con tres colores primarios no es posible representar todos los colores. Además las características físicas de los dispositivos de entrada o de salida limitan los colores que son capaces de reproducir o capturar. Por todo ello se han desarrollado varios espacios de color estándar basados en RGB que permiten referirnos a un color concreto de forma inequívoca y que sirven para cubrir unas necesidades concretas. Ejemplos de estos espacios son entre otros: Adobe RGB, Apple RGB, CIE RGB o Adobe Wide-Gamut RGB [12]. Quizá uno de los espacios RGB más conocidos y utilizados es el sRGB. Se diseñó originalmente como estándar de color para todos los dispositivos que manejasen imágenes digitales. La idea es que su adopción requiriera el mínimo coste tanto software como hardware [13] en una época en la que los monitores CRT eran los dispositivos de salida más utilizados.

| | *Rojo* | *Verde* | *Azul* | $D_{65}$ |
|---|---|---|---|---|
| $x$ | 0,64 | 0,30 | 0,15 | 0,3127 |
| $y$ | 0,33 | 0,60 | 0,06 | 0,3290 |
| $z$ | 0,03 | 0,10 | 0,79 | 0,3583 |

**Cuadro 2.1:** Cromaticidades CIE para los colores primarios y blanco de referencia de ITU-R BT.709.

Básicamente los diferentes espacios de color RGB quedan determinados tanto por su blanco de referencia como por los colores primarios que utilizan. sRGB utiliza la especificación $RGB_{709}$ o ITU-R BT.709 que establece los valores de cromaticidad CIE para los colores primarios y blanco de referencia (ver Cuadro 2.1). Para convertir desde el espacio CIE XYZ a sRGB se usa la siguiente ecuación:

$$\begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix} = \begin{bmatrix} 3,2410 & -1,5374 & -0,4986 \\ -0,9692 & 1,8760 & 0,0416 \\ 0,0556 & -0,2040 & 1,0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.16}$$

Para convertir del espacio sRGB a CIE XYZ se utiliza la inversa de la matriz anterior.

En este punto cabe notar que los sistemas de entrada de imágenes como las cámaras están expuestos a radiaciones lineales de la luz procedente de los objetos. Las intensidades lineales RGB en el rango $[0, 1]$ son transformadas a un espacio RGB no lineal, denotado como R'G'B', mediante una función de transferencia similar a:

$$R' = \begin{cases} 4.5R & R \leq 0.018 \\ 1.099R^{\frac{1}{\gamma C}} & en\, otro\, caso \end{cases} \tag{2.17}$$

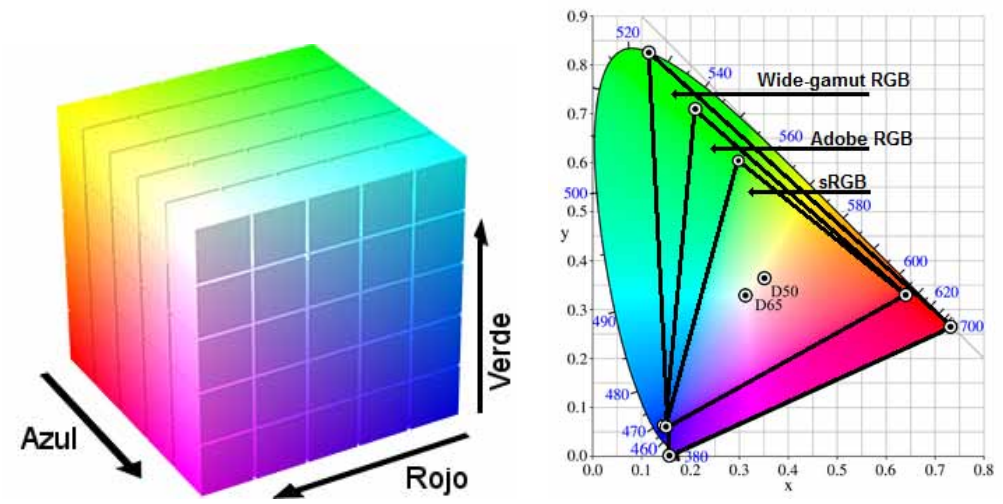
$$G' = \begin{cases} 4.5G & G \leq 0.018 \\ 1.099G^{\frac{1}{\gamma C}} & en\, otro\, caso \end{cases} \tag{2.18}$$

$$B' = \begin{cases} 4.5B & B \leq 0.018 \\ 1.099B^{\frac{1}{\gamma C}} & en\, otro\, caso \end{cases} \tag{2.19}$$

donde $\gamma C$ es el factor gamma del dispositivo de adquisición, habitualmente es $\frac{1}{0.45}$. Posteriormente el dispositivo de salida realiza el proceso inverso convirtiendo nuevamente de R'G'B' a RGB [7].

La mayor limitación del espacio sRGB es que su gamut está restringido a la gama de colores reproducibles por los CRT [14]. No obstante, existen otras otras propuestas con un gamut mucho mayor como son: Adobe RGB, Adobe Wide-Gamut RGB o ProPhoto RGB. En la Figura 2.3 podemos observar gráficamente como cada conjunto de colores primarios define un gamut diferente, siendo el color $D_{65}$ el blanco de referencia para sRGB y para Adobe RGB, mientras que $D_{50}$ lo es para Adobe Wide-Gamut RGB.

Los colores que un espacio de color puede representar queda definido por los colores primarios que utiliza. En teoría es un espacio continuo que se puede representar en el diagrama CIE. Sin embargo, debido a la naturaleza discreta de los sistemas digitales, este gamut necesita ser muestreado y cada color codificado con un número determinado de bits. Esto es lo que en la práctica determina la cantidad total de colores que puede representar un espacio de color. Así conforme más bits se utilicen por canal, más colores diferentes se podrán codificar. Lo habitual es utilizar 8 bits por canal, también conocido como *truecolor*, permitiendo representar más de 16 millones de colores. En espacios de color con un gamut grande como Adobe Wide-gamut RGB, la densidad de colores codificados con 8 bits es sensiblemente menor que en sRGB, por lo que puede ser recomendable usar 16 bits por canal y así evitar los problemas de bandas con el color[14].




(a) (b)

**Figura 2.3:** En (a) se muestra una representación gráfica del modelo de color RGB. En (b) se muestra el gamut de sRGB, Adobe RGB y Wide-gamut RGB sobre el diagrama de cromaticidad CIE. *Adaptado de www.ramonpage.com y Wikipedia, respectivamente.*

En la actualidad el modelo RGB es el más utilizado para especificar el color, sin embargo presenta algunas desventajas que condicionan su uso en función de las necesidades de cada caso:

- Si bien el modelo está basado en el sistema visual humano, en general no se considera especialmente intuitivo para especificar el color. Existen otras alternativas más intuitivas como HSV o HSL.
- Es un modelo más uniforme que CIE XYZ, pero no es lo es tanto como los espacios CIELAB o CIELUV [15], por lo que según el caso puede ser recomendable transformar el color de un espacio a otro.
- Existe una alta correlación entre las componentes RGB. Así la correlación entre cada par de canales es: $r_{BR} = 0.78$ (correlación cruzada entre el canal B y R), $r_{RG} = 0.98$ y $r_{GB} = 0.94$ [8].

#### 2.1.2.2. Ohta

En el siglo XIX el psicólogo alemán Ewald Hering observó que ciertas combinaciones de color nunca se daban en la realidad, como los verdes rojizos o los azules amarillentos. Basándose en ello propuso la teoría de los colores opuestos que, a diferencia de la teoría tricromática, postula la existencia de tres tipos de fotorreceptores: blanco-negro, amarillo-azul y rojo-verde. Estudios posteriores a Hering demostraron que realmente el ojo tiene tres tipos de fotorreceptores destinados a percibir el color como afirma la teoría tricromática: los conos tipo L, M y S. Sin embargo, también se

comprobó que los estímulos de los conos son procesados en capas posteriores mediante las células ganglionares de la retina, dando como resultado un vector de colores opuestos formado por una componente acromática y dos cromáticas (amarillo-azul y rojo-verde), de forma análoga a la teoría de los colores opuestos [8].

Existen varias propuestas inspiradas en la teoría de colores opuestos que, al igual que aquella, modelan el color mediante una componte acromática más dos cromáticas opuestas. En la literatura podemos encontrar transformaciones tanto lineales como logarítmicas del espacio RGB que modelan el color de esta manera. El espacio de color Ohta es uno de estos espacios [16]. Fue desarrollado específicamente para el procesamiento de imágenes. Está basado en la transformada Karhunen-Loeve que se relaciona con el análisis de componentes principales (PCA), razón por la cual sus componentes están poco correlacionadas, a diferencia de lo que ocurre con el espacio RGB [17]. Para calcular sus componentes desde el espacio RGB se utilizan las siguientes ecuaciones:

$$I_1 = \frac{R + G + B}{3} \tag{2.20}$$

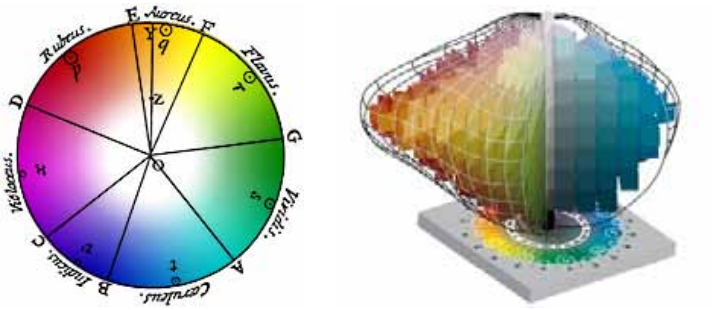
$$I_2 = \frac{R - B}{2} \tag{2.21}$$

$$I_3 = \frac{2G + R + B}{4} \tag{2.22}$$

#### 2.1.2.3. HSI y HSV

Isaac Newton fue uno de los primeros científicos en estudiar el color. En sus trabajos comprobó que la luz se podía dividir en siete bandas de color, lo que hoy conocemos como los tonos (*hues*) que componen el espectro visible o simplemente colores espectrales. Además verificó que estos tonos se podían mezclar entre sí dando lugar a otros colores. En base a estos experimentos definió el círculo del color de Newton en el que los colores espectrales se sitúan en los bordes del mismo y el blanco en el centro (ver Figura 2.4). El color formado por la combinación de dos colores espectrales se encuentran situado en la cuerda que une dichos colores, siendo el color resultante menos saturado que los que hay en el borde del círculo, lo que coincide con lo que ocurre cuando mezclamos dos haces de luz de tonos diferentes.

Si bien el círculo del color de Newton omite la propiedad del brillo (*brightness*), emplear el tono y la saturación (*saturation*) resulta la forma más natural de describir el color. Con natural nos referimos a la manera en la que el ser humano tiende a caracterizar los colores. Este forma de representar el color se conoce como espacios de color fenoménicos (*phenomenal color spaces*). Todos estos espacios describen el color basándose en los tres atributos citados previamente:



(a) (b)

**Figura 2.4:** En (a) se muestra el círculo del color de Newton y en (b) una representación gráfica del espacio de color de Munsell. *Fuente www.gusgsm.com y munsell.com respectivamente.*

- Tono: Es la cualidad del color que nos permite diferenciar entre rojo, amarillo, verde, azul, púrpura, etc. Se corresponde con los sectores en los que se divide el círculo de color de Newton.
- Saturación: Caracteriza el color como pálido o vivo. Es una medida de lo blanco que es un color. A veces también se le denomina croma (*chroma*). Un color muy saturado no tiene nada de blanco, serían los colores espectrales que están en el borde del círculo de Newton. Los colores más cercanos al centro del círculo de Newton son menos saturados.
- Brillo: Es una medida de la intensidad de la luz, por lo que también es conocida como intensidad. Nos permite describir un color como brillante u oscuro. Si tenemos dos haces de luz con un color idéntico y disminuimos la intensidad de la fuente de uno de ellos, los colores resultantes se diferencian en el brillo.

En la literatura podemos encontrar diversos espacios de color fenoménicos que son transformaciones del espacio RGB, como son: HSV (*hue, saturation, value*), HSI (*hue, saturation, intensity*) o HCI (*hue, chroma, intensity*). Todos forman parte de los denominados espacios HSL (*hue, saturation, lightness*) y no existe una única definición de ellos [18].

Existen otros espacios de color fenoménicos además de los HSL, como es el espacio de color de Ostwald y el de Munsell (ver Figura 2.4), sin embargo no son espacios continuos, si no atlas en las que las muestras de color están distribuidas de manera aproximadamente uniforme. El espacio de color de Munsell es muy utilizado sobretodo por artistas, pero no suele emplearse para el procesamiento de imágenes. La conversión desde RGB a Munsell se realiza mediante tablas de búsqueda o transformaciones no lineales [7, 19].

La tranformación desde RGB a HSI que se sugiere en [18, 7] es:

$$H = \begin{cases} 360^{o}\text{-}H' & \frac{B}{I} > \frac{G}{I} \\ H' & en\,otro\,caso \end{cases} \tag{2.23}$$

$$S = 1 - \frac{3}{(R+G+B)} \left[\text{mín}\,(R,G,B)\right] \tag{2.24}$$

$$I = \frac{1}{3}\,(R+G+B) \tag{2.25}$$

$$H' = \cos^{-1}\left[\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)\,(G-B)\right]^{1/2}}\right] \tag{2.26}$$

$H$ es posteriormente normalizado al intervalo $[0,1]$ dividiendo su valor por $360^{o}$. Si $S = 0$, $H$ no está definido ya que no tiene sentido hablar de tono cuando la saturación es nula.

Como hemos dicho, existen otras posibles transformaciones desde RGB a espacios HSL. En esta tesis empelaremos la siguiente transformación para convertir un color desde RGB a HSV [18, 7]:

$$H = \begin{cases} \frac{G-B}{MAX-MIN} & R = MAX \\ \frac{B-R}{MAX-MIN} & G = MAX \\ 4 + \frac{R-G}{MAX-MIN} & B = MAX \end{cases} \tag{2.27}$$

$$S = \frac{(MAX-MIN)}{MAX} \tag{2.28}$$

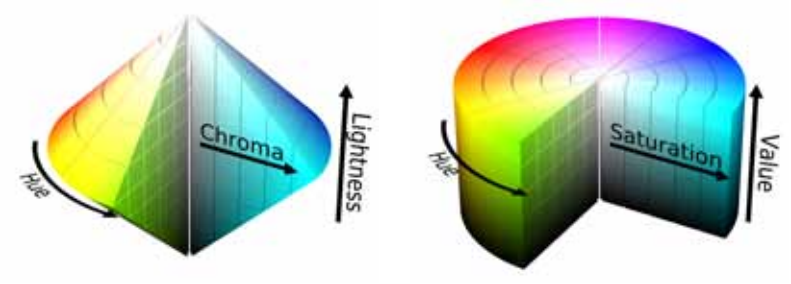
$$V = MAX \tag{2.29}$$

$$MAX = \text{máx}\,(R,G,B) \tag{2.30}$$

$$MIN = \text{mín}\,(R,G,B) \tag{2.31}$$

donde $H$ puede ser también expresado en grados multiplicando por 60. Igualmente cuando $S = 0$ el valor de $H$ es indefinido.

A pesar de que los espacios HSL son intuitivos de cara a la especificación numérica del color (ver Figura 2.5), hoy día no es una ventaja destacable, ya que los entornos gráficos actuales permiten elegir visualmente el color. Además de esto, existen numerosas desventajas que desaconsejan su uso práctico [18, 20, 7, 15]:




(a) (b)

**Figura 2.5:** Representación gráfica de espacios de color (a) HSL y (b) HSV. *Fuente Wikipedia.*

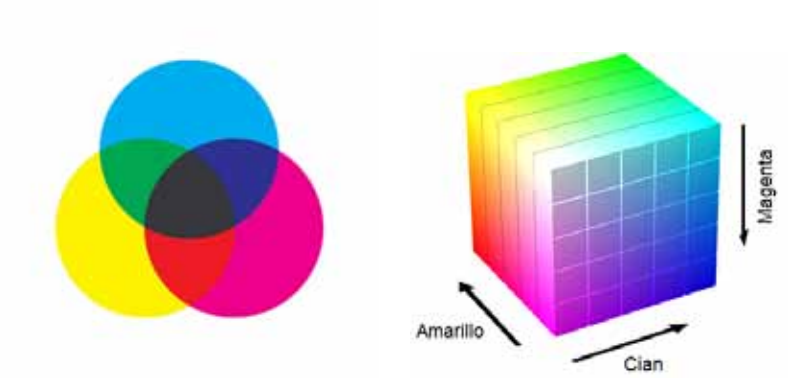
- Se cree que son espacios independientes del dispositivo, sin embargo al ser transformaciones del modelo RGB, dependen del espacio RGB concreto que se use.
- Existen singularidades en las transformaciones: el tono es indefinido cuando la saturación es nula.
- Los espacios son perceptualmente no uniformes, por lo que en general no son aconsejables para comparar colores numéricamente.
- Inestabilidad numérica en las operaciones que involucren $H$ debido a la naturaleza angular de la misma, existe discontinuidad en la componente del tono alrededor de $360^{o}$.
- Falta de estandarización en la definición de los espacios HSL. No sólo hay muchos espacios HSL, si no que a veces existen varias definiciones para el mismo espacio.
- La componente de iluminación, denominada *value, lightness* o *intensity* dependiendo del caso, es solo una aproximación a la iluminación real del color. Para calcular el valor exacto de dicha característica es necesario transformar a un espacio de color apropiado como CIELAB o CIELUV y usar la componente $L^*$.
- Las transformaciones desde RGB a un espacio HSL son no lineales y, en algunos casos, muy complejas.

### 2.1.3. Espacios de color dependientes del hardware

Aunque la mayoría de los dispositivos electrónicos trabajan con el modelo de color RGB, existen algunos que debido a sus características físicas no pueden utilizarlo. A continuación veremos los casos más representativos: los dispositivos de impresión, la televisión y la fotografía.

#### 2.1.3.1. Espacios de color para impresión

En la industria de la fotografía y en general para la impresión a color se da una situación opuesta a la de los monitores: el medio donde se visualiza el color, es decir el papel donde se imprime, no emite luz sino que la absorbe. Por ello es necesario utilizar un modelo de color sustractivo en lugar de uno aditivo como es RGB. Un modelo sustractivo es aquel en el que los colores utilizados reducen el espectro de luz visible, es decir, si no se añaden colores, el resultado es el color blanco, mientras que si se añaden todos los colores en una cantidad máxima, obtendríamos el color negro (ver Figura 2.6).




(a) (b)

**Figura 2.6:** En (a) se muestran los colores primarios CMY y las combinaciones de ellos. En (b) se muestra una representación gráfica del espacio CMY. *Adaptado de fotolia.com y www.ramonpage.com.*

El objetivo de los modelos de color sustractivos es representar cualquier color eliminando las longitudes de onda no deseadas del espectro visible [21]. En un modelo de este tipo deben emplearse colores que absorban los colores primarios. Los colores cían, amarillo y magenta, son aproximadamente el resultado de eliminar el rojo, verde y azul respectivamente de la luz blanca. Por lo tanto, estos colores deben estar presentes en cualquier modelo sustractivo de tres componentes que pretenda representar una gama de colores amplia, de ahí el modelo CMY (*cyan, magenta, yellow*) utilizado para la impresión.

Actualmente para la impresión se utiliza un modelo de cuatro componentes denominado CMYK (*cyan*, *magenta*, *yellow*, *black/key*) en el que el negro se añade por dos motivos fundamentales [20]. En primer lugar, porque es más barato utilizar una tinta en lugar de tres para representar tanto el negro como los grises, siendo todos ellos muy habituales en la impresión de documentos e imágenes. Y en segundo lugar, porque la visión humana es muy sensible a las zonas tanto blancas como negras y en la práctica la combinación de los colores cian, magenta y amarillo no logran un color completamente negro.

Resulta intuitivo convertir de RGB a CMY restando 1 a cada componente de RGB, suponiendo que el espacio está normalizado:

$$C = 1 - R \tag{2.32}$$

$$M = 1 - G \tag{2.33}$$

$$Y = 1 - B \tag{2.34}$$

Sin embargo esta transformación no logra el resultado esperado y produce resultados visualmente mediocres [20]. Esto se debe a dos motivos fundamentales. Por un lado las longitudes de onda que absorben las tintas CMY sufren cierta superposición, por lo que al emplear dos o más tintas se producen interacciones entre las longitudes de onda que controla cada componente. Por otro lado, la respuesta de cada canal no es lineal (al igual que ocurre con los monitores), por ejemplo, añadir un nivel de tinta negra de 128 (en una escala de 0 a 256) produce una reflectancia aproximada de 0.26 en lugar de 0.5 que sería lo esperable en un sistema lineal. En la práctica para transformar de RGB a CMYK lo que se utilizan son *Look-up Tables* (tablas LUT) que aunque son más costosas, permiten adaptarse a las características concretas de las tintas empleadas por cada dispositivo.

Una vez se transforma desde RGB a CMY, se pueden usar las siguientes ecuaciones para inferir el valor $K$ correspondiente a la tinta negra:

$$K = \text{mín}\,(C, M, Y) \tag{2.35}$$

$$C' = \frac{(C - K)}{(1 - K)} \tag{2.36}$$

$$M' = \frac{(M - K)}{(1 - K)} \tag{2.37}$$

$$Y' = \frac{(Y - K)}{(1 - K)} \tag{2.38}$$

Habitualmente los dispositivos de impresión que utilizan un espacio CMYK pueden representar una gama de colores más reducida que los que utilizan RGB. Esto se debe a las características físicas de la tinta, por ejemplo, la interacción entre tintas condiciona los colores que se pueden representar. Por ello el valor transformado de ciertos colores desde RGB a CMYK es una aproximación, lo que en algunos casos hace que la diferencia entre la imagen RGB original y la impresión en papel pueda ser visualmente significativa.

#### 2.1.3.2. Espacios de color para televisión

Históricamente se puede hablar de dos tipos de televisión: la SDTV y la HDTV. La SDTV (*standard-definition television*) ha sido el tipo de televisión habitual durante el siglo XX, se caracteriza por tener una resolución de imagen inferior a 1280×720 píxeles, en cambio la HDTV (*high-definition television*) iguala o supera dicha resolución.

En cuanto al color, las primeras televisiones representaban la imagen considerando únicamente la iluminación, de ahí que fueran conocidas como televisiones en blanco y negro. Posteriormente conforme el uso de la televisión en color fue en aumento, surgió la necesidad de codificar la imagen en color, pero que siguiera siendo compatible con los televisores más antiguos. Los investigadores estudiaron la manera de codificar los valores $RGB_{709}$ empleando un canal para la iluminación, denotado como $Y$'. La solución fue añadir dos canales cromáticos: $R' - Y'$ y $B' - Y$', donde $R'$ y $B'$ son componentes no lineales. Además como el sistema visual humano es menos sensible al color que a la iluminación, los canales cromáticos podían usar un ancho de banda menor, lo cual era relevante en los inicios de la televisión en color porque el ancho de banda de la señal de vídeo era muy limitado.

Durante los inicios de la televisión a color se usaba una única señal denominada señal de vídeo compuesto. Los espacios de color que se usaban en el vídeo compuesto eran: $Y'IQ$ o $Y'UV$. $Y'IQ$ está asociado a NTSC (*National Television System Committee*) que es el sistema de codificación de vídeo compuesto para la SDTV en la mayor parte de América y Japón. $Y'UV$ es un espacio de color que se utiliza tanto en NTSC como en PAL (*Phase Alternating Line*), que es el sistema de codificación habitual en Europa.

La componente lumínica $Y'$ es común a $Y'IQ$ y $Y'UV$ y puede ser calculada a partir de $R'G'B'$ mediante:

$$Y' = 0.299R' + 0.587G' + 0.114B' \tag{2.39}$$

Las componentes cromáticas de $Y'UV$ se calculan como sigue:

$$U = -0.147R' - 0.289G' + 0.437B' = 0.493\,(B' - Y') \tag{2.40}$$

$$V = 0.615R' - 0.515G' - 0.1B' = 0.877\,(R' - Y') \tag{2.41}$$

De forma similar las componentes cromáticas de $Y'IQ$ se definen como:

$$I = 0.596R' - 0.274G' - 0.322B' = 0.74\,(R' - Y') - 0.27\,(B' - Y') \tag{2.42}$$

$$Q = 0.211R' - 0.523G' + 0.312B' = 0.48\,(R' - Y') + 0.41\,(B' - Y') \tag{2.43}$$

Con la irrupción de la señal de vídeo por componentes, es decir, una señal por cada componente, se definieron nuevos espacios de color: $Y'P_BP_R$ para los interfaces analógicos y $Y'C_BC_R$ para los digitales, denominado también YCbCr. La transformación desde $R'G'B'$ a $Y'P_BP_R$ en la SDTV es:

$$Y' = 0.299R' + 0.587G' + 0.114B' \tag{2.44}$$

$$P_B = -0.169R' - 0.331G' + 0.500B' \tag{2.45}$$

$$P_R = 0.500R' - 0.419G' - 0.081B' \tag{2.46}$$

esta transformación es también empleada para codificar el color en las imágenes JPEG.

Para convertir un color de $R'G'B'$ a $Y'C_BC_R$ en la SDTV se utiliza:

$$Y' = 16 + 65.481R' + 128.553G' + 24.966B' \tag{2.47}$$

$$C_B = 128 - 37.797R' - 74.203G' + 112B' \tag{2.48}$$

$$C_R = 128 + 112R' - 93.786G' - 18.214B' \tag{2.49}$$

A pesar de que PAL y NTSC han sido los estándares para la señal de vídeo compuesto para la SDTV durante muchos años, la irrupción definitiva de HDTV ha provocado que hayan quedado obsoletos, al igual que los espacios de color $Y'IQ$ y $Y'UV$.

En el caso de HDTV, para convertir de $R'G'B'$ a $Y'P_BP_R$ donde cero es el negro y uno es el blanco, se emplean las siguientes ecuaciones:

$$Y' = 0.213R' + 0.715G' + 0.072B' \tag{2.50}$$

$$P_B = -0.115R' - 0.385G' + 0.500B' \tag{2.51}$$

$$P_R = 0.500R' - 0.454G' - 0.046B' \tag{2.52}$$

Mientras que para calcular las componentes de $Y'C_BC_R$ podemos usar:

$$Y' = 16 + 46.559R' + 156.629G' + 15.812B' \tag{2.53}$$

$$C_B = 128 - 25.664R' - 86.336G' + 112B' \tag{2.54}$$

$$C_R = 128 + 112R' - 101.730G' - 10.270B' \tag{2.55}$$

Para conocer más detalles sobre transformaciones entre espacios de color para la televisión se puede consultar [15].

#### 2.1.3.3. Espacios de color para fotografía

Dentro de los espacios de color adaptados para dispositivos específicos se encuentra Kodak PhotoYCC. Este estándar fue definido en 1992 para codificar el color de la imágenes digitales en los sistemas *PhotoCD*. Se trata de un espacio basado en la recomendación ITU-R BT.709 y, al igual que $Y'C_BC_R$, sus componentes son una ponderación de $R' - Y'$ y $B' - Y$', empleando un canal para la luminancia y dos para la cromaticidad.

Para convertir desde $R'G'B'$ a Kodak PhotoYCC se utilizan las siguientes ecuaciones:

$$Y' = 0.299R' + 0.587G' + 0.114B' \tag{2.56}$$

$$C_1' = -0.299R' - 0.587G' + 0.886B' \tag{2.57}$$

$$C_2' = 0.701R' - 0.587G' - 0.114B' \tag{2.58}$$

## 2.2. Fundamentos de la videovigilancia

Los sistemas de videovigilancia tradicionales tienen una capacidad limitada para detectar y prevenir actividades o situaciones de los objetos que hay en la escena. Los operadores humanos no pueden permanecer atentos a los monitores continuamente, o al menos no son capaces de hacerlo con un grado de atención adecuado. En situaciones con gran cantidad de objetos, el operador puede verse desbordado. Este tipo de enfoque además limita la cantidad de cámaras que se pueden usar, ya que si es necesario que cada cámara esté operada por una persona, el coste de utilizar un número elevado de cámaras también es alto. Esta problemática ha llevado a que la videovigilancia automatizada haya surgido como un campo de investigación muy activo en los últimos años.

La videovigilancia automática consiste en un sistema de computadores que procesa continuamente una o varias entradas de vídeo en busca de situaciones que puedan

ser de interés para el usuario. El procesamiento de la entrada se divide en etapas. En cada etapa, conforme mayor es el nivel, el tipo de información que se extrae es más cercano a la forma de interpretar la escena que tiene el ser humano. De este modo podemos distinguir las siguientes etapas:

- **Detección de movimiento**: Es el nivel de procesamiento más bajo. Consiste en averiguar qué píxeles forman parte de objetos que están en movimiento.
- **Identificación de objetos**: Se agrupan los píxeles donde haya movimiento en regiones (*blobs*) atendiendo al objeto al que pertenecen.
- **Seguimiento de objetos**: Consiste en especificar donde está cada objeto durante la secuencia de imágenes.
- **Reconocimiento de acciones**: En esta etapa se determinan qué acciones están llevando a cabo los objetos existentes.
- **Análisis de comportamiento**: Se analiza el comportamiento de los objetos a lo largo del tiempo en busca de situaciones que sean de interés para el usuario.

La Figura 2.7 muestra un diagrama con cada una de estas etapa. En las siguientes subsecciones se describirán detenidamente todas ellas.

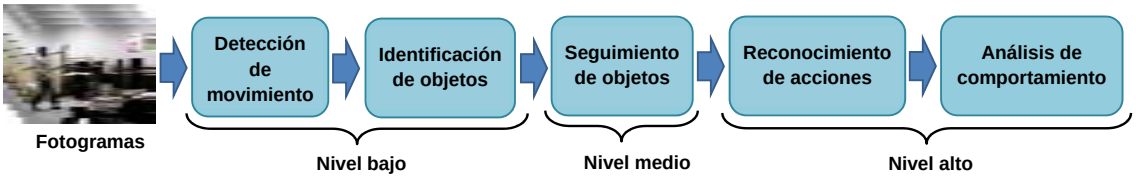



**Figura 2.7:** Marco de trabajo de la videovigilancia automática.

Los avances en la videovigilancia automatizada han permitido solventar de forma eficiente problemas ya existentes y han abierto la puerta para resolver otros nuevos, entre ellos podemos destacar los siguientes:

- Seguridad y vigilancia: Este ha sido el uso tradicional de la videovigilancia automatizada. Consiste en detectar determinados comportamientos de los objetos en un espacio controlado mediante cámaras. La automatización ha permitido que la complejidad del análisis llevado a cabo sea muy superior al que es posible mediante operadores humanos. Habitualmente se emplea tanto para controlar personas como vehículos. Los comportamientos detectados no siempre han de ser no deseados, como por ejemplo una pelea entre personas o un vehículo que realiza una maniobra prohibida; también pueden tratarse de otros comportamientos, por ejemplo, un sistema que regule el tráfico automáticamente en función del uso de cada vía o que estudie cuántas personas llevan equipaje al entrar en una estación.
- Conducción asistida: Mediante el uso de cámaras para el análisis del entorno de un vehículo es posible evitar de forma automática colisiones con otros objetos o

facilitar la conducción manteniendo automáticamente la distancia de seguridad con el vehículo precedente.

- Análisis de vídeo basado en contenido: Los métodos utilizados para videovigilancia posibilitan la generación automática de metadatos a partir de vídeos mediante el análisis del comportamiento de los objetos que aparecen en ellos [1]. De esta manera es posible realizar búsquedas complejas basándose en estos metadatos, lo que cada vez es una necesidad mayor debido a la gran cantidad de horas de grabación que pueden almacenarse en los sistemas de videovigilancia. No obstante su aplicación no sólo se restringe al ámbito de la videovigilancia, hoy día existe un gran volumen de vídeos tanto en Internet como en el entorno doméstico, por lo que en ambos casos las técnicas de videovigilancia pueden ser útiles para indexar la información y realizar búsquedas complejas.
- Interacción hombre-máquina: Mediante el reconocimiento de acciones se abren nuevas vías de comunicación hombre-máquina, permitiendo que las personas puedan interactuar mediante gestos o miradas con los ordenadores. Así el desarrollo de la videovigilancia puede potenciar la funcionalidad de entornos interactivos como por ejemplo las habitaciones inteligentes [22], videojuegos como Microsoft's Kinect o aplicaciones para el ocio como Aqu@theque [23].
- Biometría: La biometría tradicional ha consistido en identificar las personas mediante características fisiológicas como el iris, la huella dactilar o la cara. La limitación de este enfoque consiste en que requiere de la colaboración de los sujetos a identificar para poder acceder a dichas características, por ejemplo, para obtener la huella dactilar es necesario que el usuario coloque su mano en un lector. La videovigilancia ofrece nuevas posibilidades para identificar personas mediante el análisis del comportamiento propio de cada individuo, lo cual no requiere la colaboración directa del individuo, por ejemplo, una cámara puede grabar la entrada a un edificio e identificar directamente a las personas por sus características físicas sin que estas tengan que hacer nada particular. Un caso de uso muy representativo es el *HumanID Gait Challenge Problem* [24] que aborda el problema de la identificación de personas basándose en su forma de caminar.

### 2.2.1. Detección de movimiento

Prácticamente todos los sistemas de videovigilancia tienen un primer nivel de detección de movimiento. En él se deben detectar los objetos que hay en la escena. Aunque el resto de niveles sean muy sofisticados, si un objeto no es reconocido como tal, difícilmente se podrá interpretar bien lo que está ocurriendo en la escena, de ahí que realizar una buena detección de movimiento sea clave para la videovigilancia. Si bien la calidad de la detección es importante, también lo es que la detección sea suficientemente rápida como para que sea posible analizar el vídeo de entrada en

tiempo real, por tanto idealmente los métodos de detección de movimiento deben superar los 15 fps.

Esencialmente durante la detección de movimiento se buscan los píxeles que corresponden a objetos de la escena. Para ello se analizan los fotogramas que componen el vídeo de entrada y se comprueba si ha habido cambios cuyo origen es el movimiento de un objeto. Si hay objetos en movimiento, se marcan los píxeles correspondientes como parte del primer plano y el resto se consideran parte del fondo, por ello esta etapa también se conoce como segmentación del fondo o simplemente segmentación.

#### 2.2.1.1. Definición

La detección de movimiento puede ser vista como un proceso en el que existe una salida y una entrada. La entrada consiste en un vídeo en color que registra una escena real a lo largo del tiempo. En el caso más sencillo la cámara es estática, es decir, no gira sobre sus ejes ni tampoco realiza zoom. En esta tesis consideraremos el vídeo de entrada como una secuencia de imágenes o fotogramas de tamaño $fil \times col$ donde cada píxel tiene $D$ componentes de color. Cada fotograma en el instante $t$ será representado como una matriz $\boldsymbol{I}_t$ de tamaño $fil \times col \times D$. Para mayor sencillez denotaremos directamente como $\boldsymbol{p}_t$ al vector de color en el instante $t$ del píxel cuya posición es $p \in \{1...fil\} \times \{1...col\}$.

La salida es un vídeo en el que se distingue claramente donde están los objetos del primer plano. Al igual que para la entrada, la salida será considerada como una secuencia de imágenes. En concreto, cada fotograma de salida será una imagen binaria en la que el fondo se marcará con un color y los objetos del primer plano con el otro.

Por todo ello, el objetivo de los métodos de detección de movimiento consiste en calcular una función que asigne un valor a cada píxel para un fotograma dado en base a si pertenece o no al primer plano. Esta función, denominada función de pertenencia al primer plano y denotada como $\mathcal{M}(p,t)$, viene dada por la siguiente expresión:

$$\mathcal{M}(p,t) = \begin{cases} 1 & si\ \boldsymbol{p}_t\ es\ primer\ plano \\ 0 & en\ otro\ caso \end{cases} \tag{2.59}$$

Calculando el resultado de $\mathcal{M}(p,t)$ para todos los píxeles de la entrada en el instante $t$ obtendremos una imagen binaria denominada máscara del primer plano o simplemente salida. La secuencia de todas estas máscaras a lo largo del tiempo componen el vídeo de salida del proceso.

#### 2.2.1.2. Etapas

En general los métodos de detección que se emplean en la actualidad usan un modelo del fondo de la escena con el fin de compararlo con el fotograma actual para deducir si se han producido cambios que denoten la existencia de objetos en movimiento. Este modelo de fondo debe ser inicializado y mantenido a lo largo del procesamiento, dando lugar a cuatro etapas diferentes (ver Figura 2.8): inicialización, detección del primer plano, post-procesado y actualización. Antes de entrar en profundidad vamos a definir cada una de estas etapas.

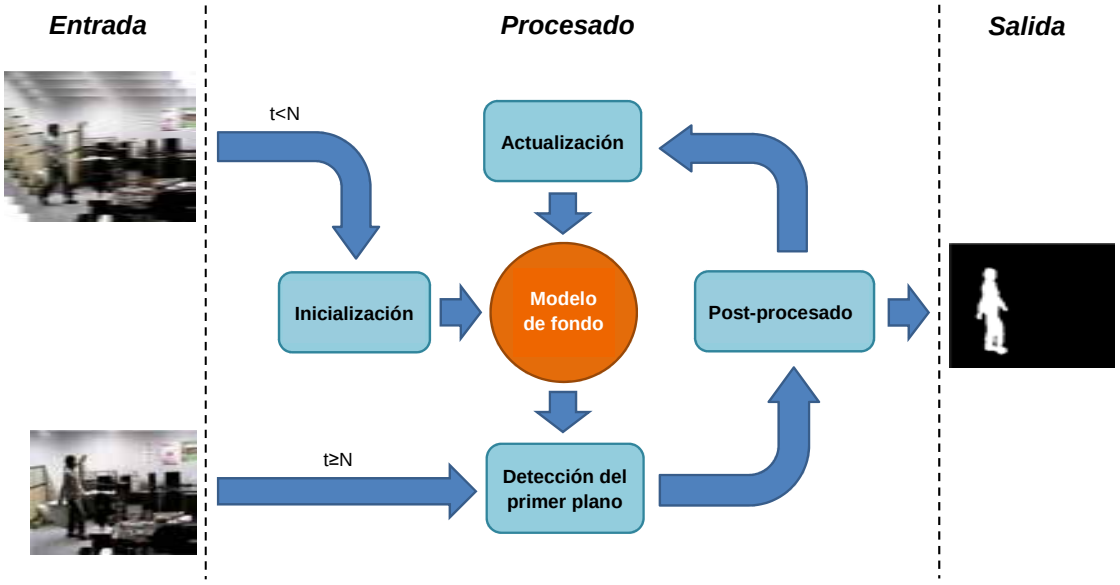



**Figura 2.8:** Diagrama de flujo para la detección de objetos en movimiento correspondiente a un fotograma concreto.

El objetivo de la **inicialización** consiste en generar un modelo de fondo inicial lo más ajustado posible a la realidad, esto se consigue cuando en dicho modelo sólo hay representados valores relacionados con el fondo de la escena. En caso contrario pueden producirse errores que previsiblemente se arrastrarán durante el resto del procesamiento, deteriorando tanto los modelos de fondo sucesivos como las máscaras del primer plano generadas. Los métodos actuales implementan la inicialización analizando las características de un conjunto inicial de fotogramas. Un posible modelo de fondo inicial básico podría ser una imagen compuesta por la media de las intensidades del color de los primeros 10 fotogramas. En función de cómo se elijan los fotogramas para la inicialización tenemos, o bien métodos que utilizan total o parcialmente un conjunto determinado de fotogramas, siendo este el caso más habitual, o bien métodos en los que el número de fotogramas a emplear no está fijado de antemano, generando de manera progresiva un modelo inicial [25]. En ocasiones se le presta poca atención a la inicialización del modelo de fondo asumiendo que para crear el modelo inicial se podrán emplear fotogramas sin objetos del primer

plano, situación que no suele ocurrir en la realidad y que por tanto no aceptaremos en nuestras propuestas.

La **detección del primer plano** utiliza el modelo de fondo y el fotograma actual para determinar qué píxeles corresponden al primer plano y cuales al fondo. Puede verse como un proceso de clasificación. Si nuestro modelo de fondo es una imagen compuesta con las intensidades de color de los píxeles del fondo, una implementación básica consistiría en marcar como fondo aquellos píxeles de la imagen de entrada que son iguales a los del modelo de fondo y el resto marcarlos como primer plano.

Para adaptar el modelo de fondo a los cambios que ocurren a lo largo de la secuencia se emplea una etapa de **actualización**. Si el fondo de la escena es mayormente estático, su importancia no es muy alta, ya que con una buena inicialización no habría que corregir el modelo de manera significativa. Sin embargo en entornos dinámicos como es el caso de esta tesis, su importancia es clave ya que de una buena adaptación a los cambios del fondo dependerá el rendimiento global del método de segmentación. Los puntos más destacados de la actualización son:

- **Ecuación de actualización**. Define cómo se van a actualizar los valores que forman el modelo de fondo. Existen tres tipos de esquemas de actualización [26]: a ciegas, selectiva y difusa. Sea $\boldsymbol{B}_t(p)$ el valor del modelo de fondo del píxel $p$ en el instante $t$. Para la actualización a ciegas tendríamos la siguiente ecuación de actualización:

$$\boldsymbol{B}_{t+1}(p) = (1-\alpha)\,\boldsymbol{B}_t(p) + \alpha \boldsymbol{p}_t \tag{2.60}$$

  donde $\alpha \in [0,1]$ es la tasa de aprendizaje. Este esquema es muy simple, sin embargo no tiene en cuenta si el píxel en cuestión pertenece o no al fondo, por lo que si el valor de $\alpha$ es cercano a 0 puede ser perjudicial cuando el fondo es dinámico ya que el modelo de fondo no se adapta correctamente. Mientras que si es mayor, distorsionaremos el modelo de fondo con las intensidades de los objetos del primer plano. Para solventar este problema se define el esquema selectivo como sigue:

$$\boldsymbol{B}_{t+1}(p) = \begin{cases} (1-\alpha_1)\,\boldsymbol{B}_t(p) + \alpha_1 \boldsymbol{p}_t & \boldsymbol{p}_t\, es\, fondo \\ (1-\alpha_2)\,\boldsymbol{B}_t(p) + \alpha_2 \boldsymbol{p}_t & en\, otro\, caso \end{cases} \tag{2.61}$$

  donde $\alpha_2 < \alpha_1$ con la idea de que sean las intensidades de los píxeles marcados como fondo los que adapten el modelo de fondo y no los del primer plano. De esta manera es habitual que $\alpha_2 = 0$, quedando la siguiente ecuación de actualización:

$$\boldsymbol{B}_{t+1}(p) = \begin{cases} (1-\alpha)\,\boldsymbol{B}_t(p) + \alpha \boldsymbol{p}_t & \boldsymbol{p}_t\, es\, fondo \\ \boldsymbol{B}(p) & en\, otro\, caso \end{cases} \tag{2.62}$$

  El desempeño del esquema selectivo depende de lo buena que sea la segmentación, ya que los píxeles mal clasificados distorsionarán el modelo de fondo. Este

problema puede ser atenuado utilizando un esquema difuso en el que la pertenencia de un píxel al fondo se mide mediante un valor difuso $\mu_B$, obteniendo la siguiente ecuación de actualización:

$$\boldsymbol{B}_{t+1}(p) = (1-\mu_B)\,\boldsymbol{B}(p) + \mu_B\left((1-\alpha)\,\boldsymbol{B}(p) + \alpha \boldsymbol{p}_t\right) \tag{2.63}$$

Como puede observarse el esquema selectivo es un caso particular del esquema difuso en el que $\mu_B = 1$ cuando el píxel es marcado como fondo y $\mu_B = 0$ en otro caso.

- **Tasa de aprendizaje**. Es el parámetro que controla lo pronunciados que son los cambios en el modelo de fondo. Puede ser un valor tanto fijo como ajustado dinámicamente a lo largo de la secuencia.
- **Frecuencia de actualización**. Determina cada cuanto tiempo se actualiza el modelo de fondo. Lo habitual es que en todos los fotogramas se realice algún ajuste del modelo de fondo y sea la tasa de aprendizaje quien controle la magnitud estos cambios, sin embargo hay algunas propuestas que no actualizan el modelo cuando los cambios son pequeños [27].
- **Mecanismos de ajuste**. Si bien la tasa de aprendizaje controla lo rápido que se adapta el modelo de fondo a la escena actual, en ocasiones pueden ocurrir situaciones difíciles de modelar, por ejemplo, cuando un objeto del primer plano se queda estático. En esta situación el modelo podría marcar como primer plano a ese objeto de manera indefinida, lo cual puede deteriorar el rendimiento del método al no actualizar el modelo de fondo con la información real. Por ello en ocasiones se definen mecanismos para trata esta problemática, por ejemplo, en el caso anterior podríamos utilizar contadores que limiten el número máximo de fotogramas consecutivos que un mismo píxel puede ser considerado primer plano.

El propósito de la etapa de **post-procesado** es mejorar la máscara generada durante la detección del primer plano. Para ello se emplean técnicas a nivel de píxel o región, en la Subsección 2.3.3 se exponen algunas de ellas. El post-procesado no siempre está presente en los métodos de segmentación, pudiendo pasar de la detección de primer plano a la actualización directamente. En caso de que se considere oportuno, también puede añadirse una etapa de post-procesado externa al procesamiento que mejore la máscara generada. En este caso el proceso de actualización del modelo de fondo no se verá afectado, por lo que las máscaras generadas durante el proceso de detección de primer plano de los fotogramas sucesivos seguirá igual que si no hubiera post-procesado.

#### 2.2.1.3. Modelos básicos

La idea básica para detectar los objetos del primer plano consiste en obtener un modelo de fondo de la escena y marcar como primer plano aquellos píxeles cuyo

color no es similar al del modelo. Tradicionalmente se ha restado el valor del color de imagen actual al del modelo de fondo para después aplicar un umbral. Las diferencias que superen el umbral denotarán que el píxel pertenece al primer plano, en caso contrario se marcarán como fondo. Este método se conoce como **sustracción del fondo**. De este modo la función $\mathcal{M}(p,t)$ sería la siguiente:

$$\mathcal{M}(p,t) = \begin{cases} 1 & \|\boldsymbol{p}_t - \boldsymbol{B}_t(p)\| > T \\ 0 & en\ otro\ caso \end{cases} \tag{2.64}$$

donde $T$ es el umbral constante y $\|\boldsymbol{p}_t - \boldsymbol{B}_t(p)\|$ es una medida de lo diferente que es la intensidad actual del píxel $p$ respecto a la del modelo de fondo.

Este método de detección del primer plano es muy simple, sin embargo la calidad de la segmentación está condicionada por la elección de un buen modelo de fondo y por el valor del umbral. Como $T$ es constante y común para toda la imagen, no contempla lo dinámica que pueda ser la escena. Así la elección del valor $T$ está completamente condicionada a las características propias de cada escena. Si las diferencias entre los objetos del primer plano y el fondo son pequeñas, el valor $T$ deberá ser pequeño; sin embargo un valor de $T$ demasiado pequeño puede hacer que leves variaciones en la intensidad del fondo sean marcadas como primer plano. Esto hace que su aplicación práctica esté limitada a entornos donde el fondo de la escena sea muy estático y el color de los objetos del primer plano sea suficientemente distinto al del fondo.

Mediante la sustracción del fondo sabemos cómo detectar el movimiento una vez tenemos un modelo del fondo. No obstante, modelar el fondo es un problema muy complejo por sí mismo, como tendremos ocasión de explicar más adelante (Subsección 2.3.1). En la literatura existen propuestas básicas para realizar esta tarea, buena parte de ellas utilizan la sustracción de fondo como método de detección del primer plano. A continuación vamos a hablar de los modelos básicos más representativos:

- **Diferencia entre fotogramas**

  Consiste en utilizar como modelo de fondo el fotograma inmediatamente anterior al actual, $\boldsymbol{B}_t(p) = \boldsymbol{p}_{t-1}$, de lo que se deriva la siguiente función de pertenencia al primer plano:

  $$\mathcal{M}(p,t) = \begin{cases} 1 & \left\|\boldsymbol{p}_t - \boldsymbol{p}_{t-1}\right\| > T \\ 0 & en\ otro\ caso \end{cases} \tag{2.65}$$

  Es un modelo que se adapta bien a los cambios en la escena. Es muy rápido y fácil de implementar, por lo que sirve como primera aproximación para modelar el fondo. Asume que el fondo es completamente estático, de lo contrario cualquier cambio en el fondo será marcado como primer plano en función del umbral.

  Al considerar todas las intensidades de los píxeles del fotograma anterior como fondo, se pueden dar dos situaciones problemáticas. En primer lugar, si los

píxeles que en $t-1$ pertenecen a un objeto en movimiento, en $t$ también pertenecen, se marcarán o no en función de lo diferentes que sean sus colores, por lo que si el objeto tiene áreas grandes de un color parecido, no se marcarán como primer plano. En segundo lugar, las zonas donde en $t-1$ había un objeto y en $t$ no, se marcarán como primer plano, lo cual también es un error (regiones fantasma). De esta manera, si un objeto se mueve despacio es probable que se produzca el primer tipo de error, mientras que si lo hace deprisa se daría el segundo tipo.

En la secuencia (a) de la Figura 2.9 vemos un objeto que se desplaza despacio con un color uniforme, el interior del mismo es segmentado pobremente, además el fondo de la escena es dinámico por lo que también se producen fallos. En la secuencia (b) se observa claramente el problema de las regiones fantasma. En general este modelo funciona correctamente sólo en aquellos píxeles donde en $t-1$ no hay un objeto en movimiento.

**Figura 2.9:** Resultados del modelo basado en la diferencia entre fotogramas. De izquierda a derecha: modelo de fondo, fotograma actual, resultado de utilizar el modelo basado en la diferencia de fotogramas. En color negro el fondo y en blanco el primer plano. En la primera fila el objeto se mueve despacio, en la segunda lo hace despacio. *Fuente: elaboración propia.*

- **Modelo de doble diferencia** [28]

  Este modelo puede considerarse como una mejora del anterior. Emplea los tres últimos fotogramas para modelar el fondo. Calcula la diferencia entre el fotograma actual $t$ y el $t-1$ y la diferencia entre el $t$ y el $t+1$. Los píxeles que superen el umbral en ambas diferencias serán candidatos a pertenecer al primer plano.

  Para evitar la segmentación de píxeles aislados, se utilizan bloques de 4x4 píxeles. Aquellos bloques cuya mayoría de píxeles sean candidatos a ser primer

plano, se marcarán definitivamente como primer plano (ver Figura 2.10). Este procesamiento a nivel de bloque permite eliminar los píxeles aislados, lo que en ocasiones es beneficioso, por ejemplo en presencia de ruido. Sin embargo puede ser perjudicial si los objetos del primer plano tienen partes pequeñas en comparación con el tamaño del bloque, en este caso podrían eliminarse estos detalles en el resultado final.

La ventaja principal de este modelo frente al de la diferencia de fotogramas es que previene las regiones fantasma (segunda fila de la Figura 2.10). Sin embargo al considerar una historia de tan solo tres fotogramas, no mejora los resultados cuando el objeto se mueve tan despacio que los píxeles del primer plano coinciden durante esos fotogramas y el objeto tiene áreas de un color uniforme (primera fila de la Figura 2.10).

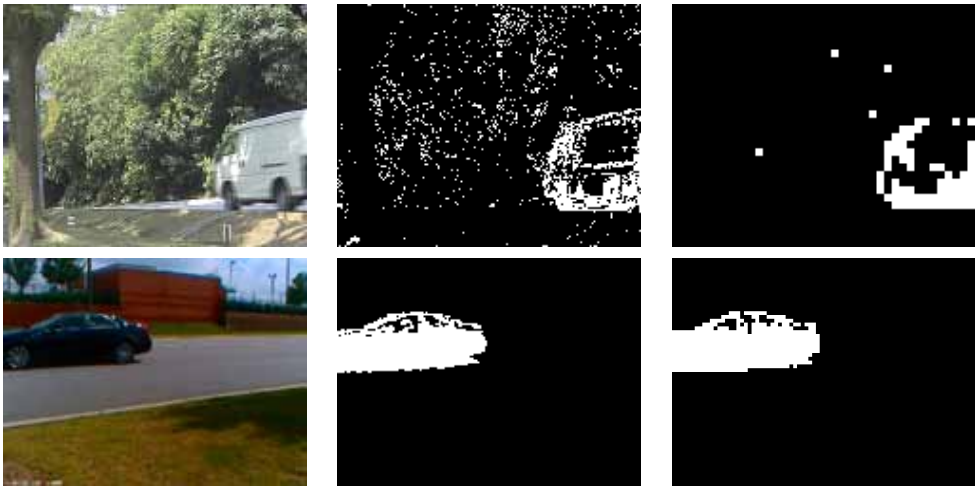


**Figura 2.10:** Resultados del modelo de doble diferencia. De izquierda a derecha: fotograma actual, resultado al utilizar el modelo de doble diferencia sin usar bloques y resultado usando bloques 4x4. *Fuente: elaboración propia.*

- **Modelo basado en la media** [29]

  Este tipo de modelo consiste en calcular el modelo del fondo realizando la media aritmética de las intensidades de un píxel a lo largo de un periodo de tiempo. Se fundamenta en que las intensidades que mayoritariamente tendrá un píxel a lo largo de la secuencia serán las que se corresponden con el fondo, ya que los objetos en movimiento suelen ocupar un píxel durante pocos fotogramas. El modelo de fondo que considera las intensidades desde el inicio de la secuencia sería el siguiente:

  $$\boldsymbol{B}_t(p) = \frac{1}{t-1}\sum_{i=1}^{t-1}\boldsymbol{p}_i \tag{2.66}$$

  Este modelo es muy simple, sin embargo tiene varios problemas que limitan su utilidad práctica. Al ser necesario almacenar las intensidades de los últimos

$t$ fotogramas, su complejidad temporal y espacial dependen de lo larga que sea la secuencia. Las secuencias habitualmente no duran un periodo de tiempo definido. Por lo que si se trata de implementar directamente como indica la expresión anterior, en general no sería posible su uso. Otro problema adicional es que al considerar todas las intensidades con igual peso, si la intensidad del fondo ha experimentado algún cambio en un periodo de tiempo corto, el modelo no se adaptará a ello hasta que hayan pasado suficientes fotogramas como para que la media lo refleje.

Una segunda variante de este tipo de modelos se basa en utilizar una ventana de tiempo de tamaño $W$. De esta manera solo es necesario almacenar los últimos $W$ fotogramas, por lo que se superan las limitaciones comentadas en el caso anterior. El modelo de fondo vendría dado por la siguiente ecuación:

$$\boldsymbol{B}_t(p) = \frac{1}{W} \sum_{i=t-W}^{t-1} \boldsymbol{p}_i \tag{2.67}$$

En este modelo todos los fotogramas dentro de la ventana de tiempo tienen el mismo peso, $\frac{1}{W}$.

Aunque el uso de una ventana de tiempo es una ventaja de cara a las necesidades de tiempo y espacio, puede ser muy perjudicial en situaciones donde haya muchos objetos del primer plano o alguno que se mueva despacio. Por muy bueno que sea el modelo de fondo obtenido, si en un periodo de tiempo del orden de $W$ un píxel es ocupado mayoritariamente por objetos del primer plano, el modelo de fondo correspondiente a dicho píxel se degradará hasta el punto que puede llegar a considerarse como modelo de fondo la intensidad media de los objetos del primer plano. De ahí que la elección de $W$ sea muy importante (ver Figura 2.11), ya que si la ventana es pequeña, el modelo será más sensible a los objetos del primer plano que se mueven despacio, llegando a asumirse la intensidad del objeto como modelo de fondo, lo cual llevaría a errores del tipo regiones fantasma. En el caso de que la ventana sea grande podemos atenuar esta limitación, pero perderíamos adaptabilidad a los posibles cambios en la intensidad del fondo, además de aumentar los requisitos de tiempo y espacio.

Si calculamos incrementalmente la media de las intensidades de cada píxel obtenemos otro modelo de fondo definido como se expone a continuación:

$$\boldsymbol{B}_t(p) = (1-\alpha)\,\boldsymbol{B}_{t-1}(p) + \alpha \boldsymbol{p}_{t-1} \tag{2.68}$$

donde el peso del fotograma anterior es ponderado con la tasa de aprendizaje, $\alpha$. Este modelo define una ecuación de actualización a ciegas, por lo que hereda sus limitaciones, es decir, cuando $\alpha$ es cercana a 0 el modelo no se adaptará bien a un fondo dinámico y si $\alpha$ es mayor, el modelo se distorsionará con las intensidades de los objetos en movimiento.

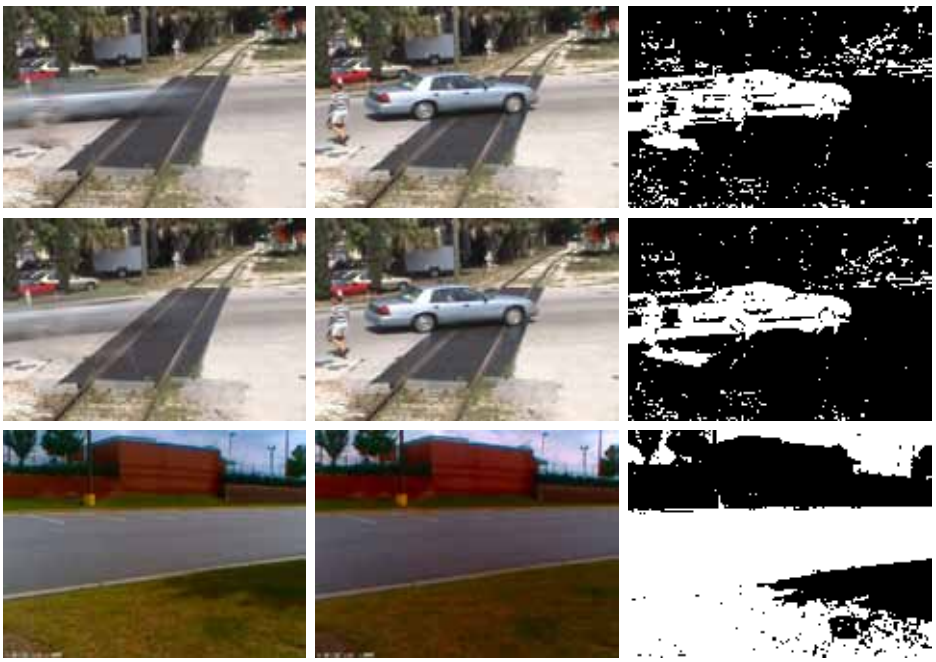


**Figura 2.11:** Resultados del modelo basado en la media usando ventana. De izquierda a derecha: modelo de fondo, fotograma actual, resultado usando $T = 50$. De arriba abajo: primera fila ($W = 25$), como el objeto se mueve despacio, distorsiona el modelo de fondo; segunda fila ($W = 60$), el modelo se adapta mejor; última fila ($W = 60$), el modelo no se adapta bien a un cambio gradual en la iluminación. *Fuente: elaboración propia.*

- **Modelo basado en la mediana** [30, 31, 32, 33]

  Es un modelo similar al de la media, pero utiliza como modelo de fondo la mediana de las últimas intensidades. Si $S_t(p)$ es el conjunto de las $W$ intensidades pasadas de un píxel $p$:

  $$S_t(p) = \left\{ \boldsymbol{p}_{t-W}, ..., \boldsymbol{p}_{t-1} \right\} \tag{2.69}$$

  Ordenando $S_t(p)$ de manera que $b_i$ es el i-ésimo elemento en dicho conjunto ordenado, el modelo de fondo vendría dado por la siguiente expresión:

  $$\boldsymbol{B}_t(p) = b_{\frac{W+1}{2}+1} \tag{2.70}$$

  La característica más destacada que ofrece esta solución es que la mediana puede ser más robusta frente al ruido que la media, siendo inmune a los valores de intensidad extremos si la ventana es suficientemente grande. Como puede verse en la Figura 2.12, si la ventana es pequeña cuando hay objetos que se desplazan despacio, el resultado es incluso más pobre que usando la media, pero al aumentar la ventana de tiempo la segmentación es sensiblemente mejor.

  No obstante, existen dos problemas comunes entre este modelo y el basado en la media. Por un lado, una ventana excesivamente grande también supone un

problema de adaptabilidad a los cambios recientes en el fondo. Por otro lado, almacenar $S_t(p)$ y ordenar cada nueva intensidad, supone unos requisitos de tiempo y espacio que aumentan directamente en función del tamaño de la ventana considerada.

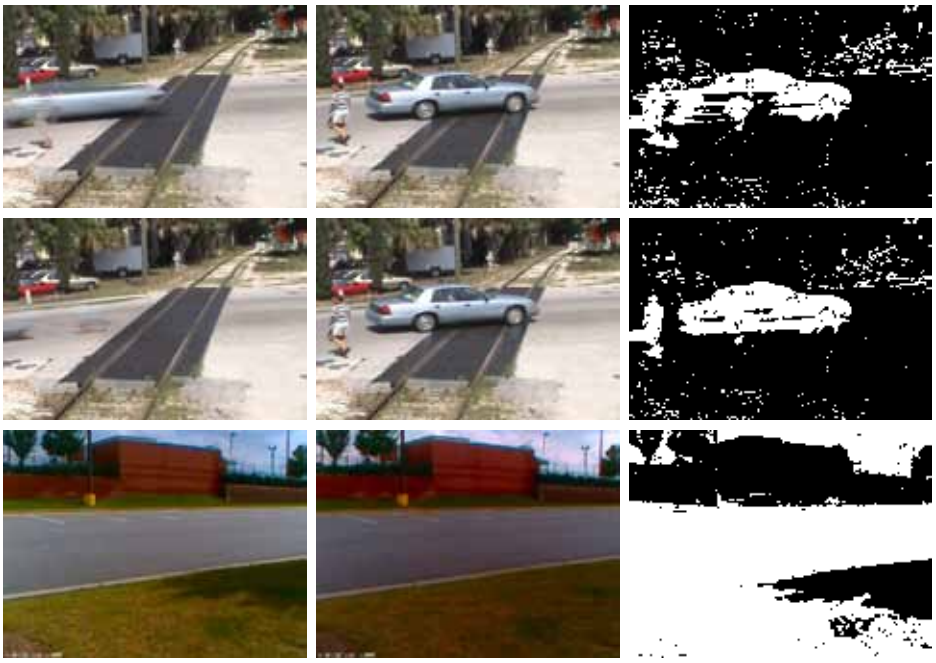


**Figura 2.12:** Resultados del modelo basado en la mediana. De izquierda a derecha: modelo de fondo, fotograma actual, resultado usando $T = 50$. De arriba abajo: primera fila ($W = 25$), igual que con la media el modelo de fondo se distorsiona visiblemente; segunda fila ($W = 60$), el modelo se adapta incluso mejor que usando la misma ventana para la media; última fila ($W = 60$), al igual que con el modelo basado en la media tarda en adaptarse a un cambio gradual en la iluminación. *Fuente: elaboración propia.*

- **Sakbot**

  El método denominado Sakbot (*Statistical And Knowledge-Based Object Tracker*) descrito en [32, 33] utiliza un modelo de fondo basado en la mediana más sofisticado que el descrito anteriormente. El conjunto de elementos que se utilizan para calcular el modelo de fondo es el siguiente:

$$S_t(p) = \left\{ \boldsymbol{p}_{i-\nabla\tau}, ..., \boldsymbol{p}_{i-m\nabla\tau} \right\} \cup w_b \left\{ \boldsymbol{B}_{t-1}(p) \right\} \tag{2.71}$$

  donde $w_b$ es una constante, $m$ es el número de intensidades elegidas de fotogramas pasados y $\nabla\tau$ es la frecuencia de muestreo, típicamente 10. Por tanto no se eligen todas las intensidades pasadas, aumentando el tamaño efectivo de la ventana sin aumentar la cantidad de intensidades almacenadas.

La actualización del modelo de fondo sigue un esquema selectivo, si el píxel pertenece al primer plano, el modelo queda inalterado; en caso de que sea fondo, se actualiza con el valor de la mediana de $S_t(p)$.

Para determinar si un píxel pertenece al primer plano se consideran dos máscaras $M_L$ y $M_H$:

$$M_L(\boldsymbol{p}_t) = \begin{cases} 1 & \|\boldsymbol{p}_t - \boldsymbol{B}_t(p)\| > T_L(\boldsymbol{p}_t) \\ 0 & en\, otro\, caso \end{cases} \tag{2.72}$$

$$M_H(\boldsymbol{p}_t) = \begin{cases} 1 & \|\boldsymbol{p}_t - \boldsymbol{B}_t(p)\| > T_H(\boldsymbol{p}_t) \\ 0 & en\, otro\, caso \end{cases} \tag{2.73}$$

$$T_L(\boldsymbol{p}_t) = \lambda \left( b_{\frac{m+1}{2}+l} - b_{\frac{m+1}{2}-l} \right) \tag{2.74}$$

$$T_H(\boldsymbol{p}_t) = \lambda \left( b_{\frac{m+1}{2}+h} - b_{\frac{m+1}{2}-h} \right) \tag{2.75}$$

donde $l \leq h$, $\lambda$ es una constante y por tanto $T_L(\boldsymbol{p}_t) \leq T_H(\boldsymbol{p}_t)$. $M_L(\boldsymbol{p}_t)$ es utilizado para reducir los efectos negativos del ruido, mientras que $M_H(\boldsymbol{p}_t)$ identifica aquellos píxeles donde existe una gran diferencia entre su color actual y su modelo. De este modo un píxel pertenecerá al primer plano si es marcado como tal en ambas máscaras, o bien si sólo es marcado en $M_L$, pero tiene un vecino marcado en $M_H$:

$$\mathcal{M}\,(p,t) = \begin{cases} 1 & (M_L(\boldsymbol{p}_t) = 1) \wedge ((M_H(\boldsymbol{p}_t) = 1) \vee \exists \boldsymbol{q}_t \ (M_H(\boldsymbol{q}_t) = 1)) \\ 0 & en\, otro\, caso \end{cases} \tag{2.76}$$

donde el píxel $q$ es vecino de $p$.

- **Modelo basado en histograma y moda** [34]

  Además de la media y la mediana es posible utilizar la moda como valor representativo del fondo de la escena. De este modo para cada píxel se considera como modelo de fondo aquella intensidad que aparece más frecuentemente.

  Dado que en la práctica, aunque el fondo sea muy estático, durante la secuencia los colores que componen el fondo pueden sufrir pequeñas variaciones, para caracterizar el fondo con la moda es más fiable agrupar los colores cercanos, de manera que en lugar de considerar 256 intensidades de un canal con 8 bits, se consideran $\frac{256}{s}$ cubos (*bin*s), donde $s$ es la cantidad de intensidades dentro de un cubo. En cada píxel se elegirá el cubo que sea la moda y dentro de este cubo de intensidades, la intensidad que sea la moda será la que se elija como modelo de fondo.

En este caso el valor de $s$ controla la sensibilidad del modelo. Si elegimos un valor pequeño es posible que, si el fondo no es suficientemente estático, intensidades que son del fondo caigan en cubos diferentes, por lo que la segmentación podría empeorar al caracterizar mal el fondo. Por contra, si aumentamos el valor de $s$, se corre el riesgo de incluir en el mismo cubo tanto intensidades del fondo como de objetos del primer plano, por lo que estos últimos puede que no sean detectados.

A pesar de la mejora que supone agrupar las intensidades en cubos, si la ventana es tan pequeña que un objeto que se desplaza despacio ocupa mucho tiempo un píxel, al igual que en los casos anteriores el modelo de fondo se corromperá. De igual forma, una ventana demasiado grande aumentará las necesidades de tiempo y espacio.

- **Modelo basado en PIC (*Pixel Intensity Classification*)** [35, 36]

  Es una propuesta similar a la anterior. En lugar de dividir el espacio de 256 intensidades en cubos de $s$ elementos, para cada píxel se estudian las intensidades pasadas durante un periodo de tiempo determinado y se normalizan todas ellas. Si $M$ es la intensidad máxima durante el intervalo de tiempo y $m$ la mínima, se ponderarán todas la intensidades con $\frac{1}{M-m}$. Este nuevo espacio de intensidades se divide en un número fijo de conjuntos o cubos de igual tamaño.

  Se calcula cuál es la frecuencia con la que aparece cada conjunto de colores y se haya la moda. Las intensidades del conjunto que es la moda se extienden con un cierto rango de intensidades y la media de todas ellas se toma como modelo de fondo.

- **Modelo basado en PCC (*Pixel Change Classification*)** [37]

  Esta propuesta tiene dos diferencias fundamentales con las anteriores: el modelo de fondo se actualiza considerando si el píxel es fondo o primer plano y el umbral para la detección de primer plano no es constante. Para decidir si un píxel es primer plano en el fotograma actual se comprueba si ha habido un cambio importante en alguna de las componentes de color de cada píxel de la siguiente manera:

$$\mathcal{M}(p,t) = \begin{cases} 1 & \forall k, \left|p_t^k - B_t^k\right| < \lambda \sigma_t^k(p) \\ 0 & en\ otro\ caso \end{cases} \tag{2.77}$$

  donde $p_t^k$ y $B_t^k$ hacen referencia, respectivamente, a la intensidad de la imagen de entrada y al modelo de fondo del k-ésimo canal de color del píxel $p$ en el instante $t$. $\lambda$ es una constante y $\sigma_t^k(p)$ es una medida de lo variable que es la intensidad de ese canal en el píxel $p$ y se actualiza con la siguiente fórmula $\sigma_t^k(p) = max\left(\left|p_t^k - p_{t-1}^k\right|, \sigma_{t-1}^k(p)\right)$.

Si un píxel pertenece al fondo se usará la Ecuación 2.68 para actualizar el modelo de fondo. En caso de que sea primer plano se determinará la categoría a la que pertenece para saber cómo actualizar su modelo. Las categorías a las que puede pertenecer un píxel del primer plano son: cambiando, estático, borrado y fondo. En la Figura 2.13 se muestra la evolución de las intensidades que se corresponden con cada una de estas categorías.

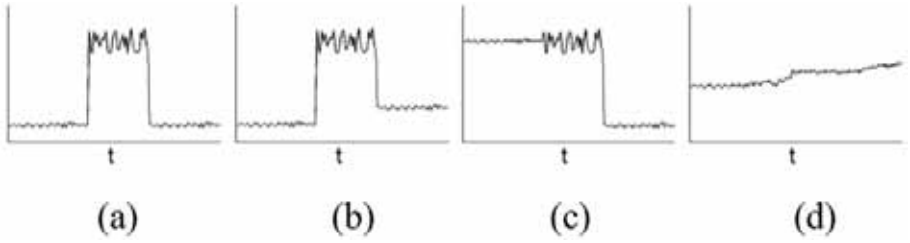



**Figura 2.13:** Modelo PCC. Evolución de las intensidades en función de la categoría a la que pertenece un píxel. En el eje horizontal el tiempo y en el vertical el valor de la intensidad. En (a) se muestra la evolución de un píxel con la categoría cambiando, (b) estático, (c) borrado y (d) fondo. *Fuente* [37]

Para saber a qué categoría pertenece un píxel se definen dos atributos: $estable_t\,(p)$ y $cambio_t\,(p)$. El primero indica si el color del píxel ha sido estable durante varios fotogramas consecutivos, mientras que el segundo detecta cambios bruscos en alguna de las componentes de color:

$$estable_t\,(p) = \begin{cases} 1 & cont \geq S_{estable} \\ 0 & en\,otro\,caso \end{cases} \tag{2.78}$$

$$cambio_t\,(p) = \begin{cases} 1 & \exists k, \left|p_t^k - p_{t-1}^k\right| \geq T_{cambio} \\ 0 & en\,otro\,caso \end{cases} \tag{2.79}$$

$$cont_t\,(p) = \begin{cases} cont + 1 & \forall k, \left|p_t^k - p_{t-1}^k\right| < T_{cambio} \\ 0 & en\,otro\,caso \end{cases} \tag{2.80}$$

siendo $T_{cambio}$ y $S_{estable}$ constantes que controlan, respectivamente, si se ha producido un cambio brusco en la intensidad o por el contrario ha sido estable durante los últimos fotogramas.

En el caso de que pertenezca a la categoría cambiando o fondo, su modelo se actualizará siguiendo la Ecuación 2.68, si no se descartará el modelo actual y se generará un nuevo modelo basado en las últimas intensidades.

### 2.2.2. Identificación de objetos

En esta etapa los píxeles marcados como primer plano deben ser analizados para identificar los objetos que hay en ellos. Se deben clasificar los objetos que son de interés para las etapas posteriores, descartando aquellos que no lo son. Así por ejemplo, en una cámara de tráfico es relevante identificar y clasificar los vehículos y las personas, pero no otros objetos en movimiento como pueden ser los pájaros u objetos arrastrados por el viento. En la Figura 2.14 se muestra un ejemplo de identificación donde los objetos se clasifican en vehículos y personas.


(a) (b) (c)

**Figura 2.14:** Ejemplo de identificación de objetos en movimiento. En (a) se muestra la escena original, (b) representa el resultado de la segmentación, en (c) los objetos han sido identificados y clasificados como vehículo y grupo de personas, denotado como C y P, respectivamente. *Fuente: elaboración propia.*

El primer problema a resolver consiste en encontrar rasgos propios de cada tipo de objeto que se quiere identificar. Básicamente existen dos tipos de rasgos a estudiar: el movimiento y la forma. Debido a que durante la etapa de seguimiento también se utiliza dicha información, en ocasiones la identificación de objetos se realiza después del seguimiento. En la Figura 2.15 se muestran dos ejemplos de rasgos que permiten clasificar los objetos en movimiento.Según el enfoque utilizado tenemos los siguientes tipos de métodos:

- **Basados en el movimiento** [40, 41, 39, 42]: Analizan los objetos a lo largo del tiempo con el fin de extraer rasgos relacionados con el movimiento que experimentan durante la secuencia. De este modo se pueden extraer características derivadas de los movimientos periódicos de algunos objetos. Por ejemplo, al andar las personas y animales tienen un patrón de movimiento de sus piernas característico, lo que permite detectar este tipo de objetos. En el caso de necesitar distinguir entre vehículos y personas se puede utilizar el patrón de velocidad de cada uno de ellos o combinar la posición dentro de la escena con la dirección del movimiento.
- **Basados en la forma** [43, 38, 44, 45, 46]: Extraen rasgos de los objetos basándose en la representación gráfica de cada uno de ellos, es decir, en el conjunto píxeles que ocupan en la escena, también denominado *blob*. Para este

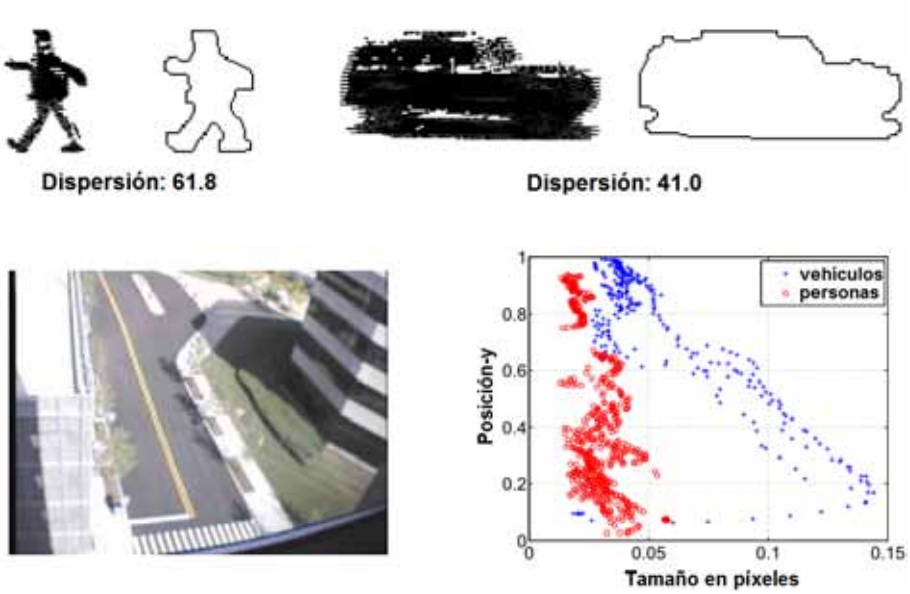



**Figura 2.15:** Ejemplo de rasgos característicos de los objetos en movimiento. Arriba se muestra la dispersión, es decir $\frac{perímetro^2}{área}$, de una persona y de un vehículo. Abajo a la izquierda se muestra una escena sin objetos en movimiento y a la derecha se representan los tipos de objeto en función de su tamaño en píxeles y de su posición a lo largo del eje vertical de la escena. *Fuente: adaptado de* [38] *y* [39], *respectivamente.*

fin se utilizan típicamente medidas tales como: el área del *blob*, la dispersión o perímetro respecto del área del *blob*, la silueta del objeto o la relación de aspecto. Por ejemplo el *blob* correspondiente a una persona suele tener una dispersión mayor que la de un vehículo y la silueta también es sensiblemente distinta.

- **Basados en el movimiento y la forma** [47]: En este tipo de métodos se explotan las bondades de los enfoques anteriores utilizando una combinación de los rasgos basados tanto en el movimiento como en la forma de los objetos del primer plano.

Una vez tenemos recopilados los rasgos de entrada de los objetos detectados, el siguiente paso consiste en utilizar un método que permita relacionar dichos rasgos con la clase a la que pertenece. Se trata de un problema de clasificación en el que los rasgos de los objetos del primer plano son la entrada y la clase a la que pertenecen es la salida. En este sentido es posible utilizar redes neuronales, máquinas de vectores de soporte, descriptores SIFT o el algoritmo AdaBoost entre otros.

### 2.2.3. Seguimiento de objetos

El seguimiento de objetos es una parte importante dentro del proceso de videovigilancia automática. La interpretación de la escena depende en buena medida de conocer cual ha sido la evolución de los objetos a lo largo de la secuencia, para lo cual es fundamental identificar y seguir a los mismos.

El problema a resolver consiste en determinar cual es la posición actual de los objetos que aparecen en los fotogramas anteriores o lo que es equivalente, asignar una etiqueta que identifique a cada objeto a lo largo de la secuencia de forma unívoca. Existen numerosas propuestas para resolver este problema, en principio realizar una búsqueda a ciegas de los objetos de interés en toda la escena puede ser una solución, sin embargo su coste en tiempo la descarta como opción viable. En general son deseables soluciones más sofisticadas como por ejemplo calcular la trayectoria de los objetos para así poder estimar su posición en los fotogramas posteriores. La complejidad del seguimiento de objetos se debe a la existencia de las siguientes dificultades:

- Imágenes con ruido que distorsionan la percepción de los objetos a seguir.
- Pérdida de información asociada a la proyección de la escena 3D real en una imagen 2D.
- Objetos no rígidos o articulados que dificultan la caracterización de los objetos mediante su forma.
- Oclusiones parciales o totales que hacen que los objetos dejen de ser visibles durante un periodo de tiempo.
- Cambios de iluminación que alteran algunos rasgos de los objetos en movimiento como el color.
- Complejidad en la forma y en el movimiento de los objetos.
- Requisitos de tiempo real.

Al igual que para la etapa de identificación, en el seguimiento también se utilizan rasgos para caracterizar a los objetos de interés. Sin embargo esta etapa requiere una caracterización más exacta de los objetos, ya que no solo necesitamos diferenciar entre diferentes tipos de objetos, si no que es necesario identificar a cada objeto individualmente. De este modo además de los rasgos ya vistos en la Subsección 2.2.2 se utilizan otros, los más habituales son los basados en:

- **Color**: El color de un objeto puede ser alterado tanto por las características de la superficie del objeto como por el tipo de luz que lo ilumina. Además el espacio de color utilizado condiciona la fiabilidad de las comparaciones que se hagan entre colores (ver Subsección 2.1). Estas consideraciones condicionan el uso de este rasgo para la identificación y seguimiento de objetos, sin embargo es una de las alternativas más utilizadas para este fin.

- **Bordes**: Los bordes de un objeto tienen la ventaja de que son robustos frente a los cambios de iluminación, por contra su forma puede cambiar debido a las rotaciones o movimiento de los objetos que delimitan. Uno de los algoritmos más utilizados para extraer los bordes de un objeto es el operador de Canny [48].
- **Flujo óptico**: Determina los vectores de movimiento de cada píxel dentro de una región. Se basa en la invariabilidad del brillo en aquellos píxeles que se corresponden entre sí en fotogramas consecutivos. Mediante técnicas como las que usa el método SKT (*Shi, Kanade y Tomasi*) [49] se pueden calcular dichos vectores.
- **Textura**: La textura es una medida de la variación de la intensidad de una superficie que cuantifica la suavidad y la regularidad de la misma. Al igual que los bordes, es una alternativa robusta frente a los cambios de iluminación. Su calculo se realiza utilizando descriptores que se basan por ejemplo en el uso de wavelets [50] o en máscaras como los operadores LBP (*Local binary patterns*).

#### 2.2.3.1. Modelos de objeto

Los objetos de interés pueden ser modelados mediante su forma y aspecto. Existen varios tipos de modelos de objeto. La elección de un modelo u otro depende del tipo de objeto a seguir y del algoritmo de seguimiento utilizado. De esta manera puede ser necesario modelar de forma distinta un objeto rígido como un vehículo que un objeto deformable como es una persona andando. A continuación vamos a describir los cinco tipos de modelos de objeto más utilizados por los algoritmos de seguimiento:

- **Puntos**: El objeto es representado mediante un punto, habitualmente el centroide de los píxeles que lo componen (Figura 2.16.a), o bien mediante un conjunto de píxeles, aquellos que están situados en zonas características y que se pueden seguir a lo largo de la secuencia (Figura 2.16.b). En este sentido existen propuestas ampliamente utilizadas como por ejemplo el método SIFT [51] que detecta puntos de interés con invarianza a la escala, orientación y transformación afín. Este tipo de modelo es adecuado para el seguimiento de objetos que ocupan regiones pequeñas en la imagen.
- **Forma geométrica primitiva**: El objeto es representado mediante una figura geométrica (Figura 2.16.c-d). Habitualmente se utiliza el rectángulo o la elipse para este fin debido a su sencillez. El movimiento con este tipo de objeto es modelado mediante traslaciones o transformaciones afines o proyectivas. Aunque es un modelo de objeto más adecuado para objetos rígidos, también se usa para el seguimiento de objetos no rígidos.
- **Silueta o contorno**: El contorno de un objeto es definido por el borde del mismo (Figura 2.16.h). La silueta es la región que queda dentro del contorno

(Figura 2.16.i). Ambos son utilizados para modelar objetos complejos y no rígidos [52].

- **Modelo articulado**: Este tipo de modelo se usa para aquellos objetos articulados que tienen partes bien diferenciadas y que se unen entre sí mediante uniones. Un caso típico es el cuerpo humano en el que las partes son cada una de las extremidades, la cabeza y el torso, mientras que las uniones son la cadera, los hombros y el cuello. Para representar un modelo articulado se pueden utilizar rectángulos o elipses (Figura 2.16.e). La relación entre cada una de las partes es determinada mediante modelos de movimiento cinemático, por ejemplo, el ángulo de las uniones.
- **Modelo esquelético**: El esqueleto de un objeto se puede extraer a partir de la silueta aplicando a la misma la transformada del eje medio [53] (2.16.f). Este modelo es utilizado habitualmente como representación de los objetos de interés [54]. El modelo esquelético se puede emplear para modelar objetos tanto rígidos como articulados.

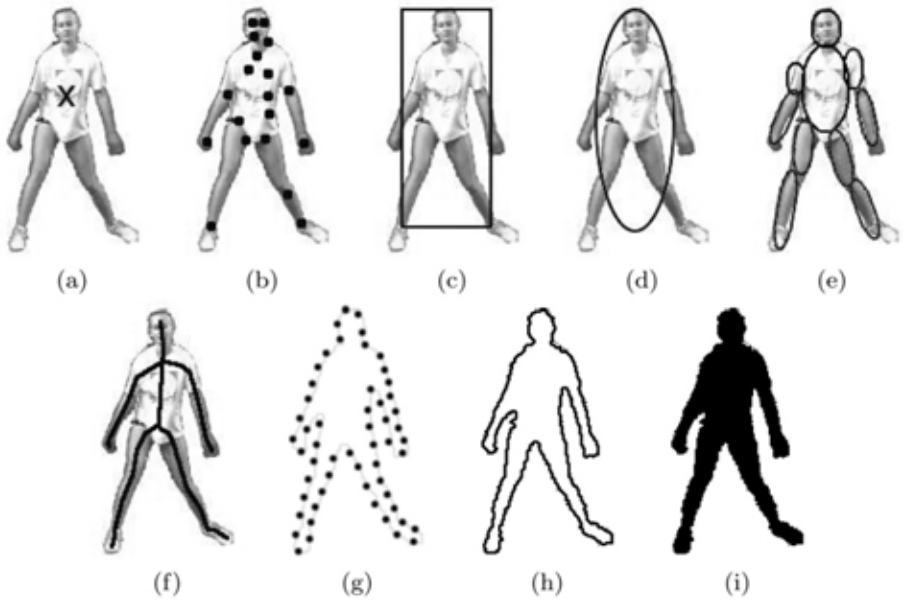



**Figura 2.16:** Ejemplos de modelos de objetos para el seguimiento. En (a) se muestra un modelo basado en el centroide del objeto, (b) es un ejemplo de modelo de puntos, (c) y (d) son formas geométricas, (e) es un modelo articulado, (f) un modelo esquelético, (g) es un contorno, (h) son los puntos de interés del contorno e (i) es una silueta. *Fuente:* [55]

El aspecto o apariencia de un objeto se puede representar de diferentes maneras. Para el seguimiento es deseable combinar el modelo de objeto visto anteriormente con el aspecto o apariencia del mismo para incrementar la robustez del método. En este sentido el aspecto de un objeto suele representarse de alguna de las siguiente maneras:

- **Densidades de probabilidad**: Las densidades de probabilidad de los rasgos de un objeto, como el color o la textura, se pueden calcular en las regiones que conforman el contorno o la silueta de un objeto. Estas densidades de probabilidad pueden ser paramétricas o no. En el caso de las paramétricas se puede utilizar una distribución gaussiana [56, 57] o bien una mixtura de distribuciones gaussianas [58, 59]. Para el caso de las no paramétricas son habituales las ventanas Parzen [60] o los modelos basados en histogramas [61].
- **Plantilla**: Consisten en formas geométricas simples o siluetas. Su ventaja reside en que incluye información del objeto tanto espacial como de apariencia [62, 63]. Sin embargo representan la apariencia de un objeto generada desde una única proyección en el mismo plano, por tanto estos modelos solo son adecuados para el seguimiento cuando la pose de los objetos no varía sustancialmente a lo largo de la secuencia.
- **Modelos de apariencia activa**: Este tipo de modelos son generados modelando simultáneamente la forma y la apariencia del objeto [64]. En general la forma del objeto es definida mediante un conjunto de puntos de referencia. De manera análoga a la representación basada en el contorno y la silueta, los puntos de referencia pueden estar situados tanto en el borde como en el interior del objeto. Cada punto de referencia tiene asociado un vector de rasgos como el color, la textura o la magnitud del gradiente. Los modelos de apariencia activa necesitan una etapa de entrenamiento en la que la forma y la apariencia es aprendida de un conjunto de muestras, por ejemplo utilizando el análisis de componentes principales.
- **Modelos de apariencia múltiple**: A diferencia del caso anterior, los modelos de apariencia múltiple modelan varias vistas de los objetos de interés. Uno de los enfoques consiste en generar un subespacio a partir de cada una de las vistas. Por tanto necesita un conjunto de muestras inicial mayor que los modelos de apariencia activa para poder aprender las diferentes vistas de un objeto. Las técnicas como el análisis de componentes principales (PCA) o el análisis de componentes independientes (ICA) reducen la dimensionalidad del espacio inicial, de ahí que sean utilizadas para representar la forma y la apariencia de los objetos [65, 66]. Otros enfoques habituales consisten en utilizar máquinas de vectores de soporte (SVM) [67] o redes bayesianas [68].

El tipo de objeto que se desea seguir es un factor clave que condiciona el tipo de representación empleado. Para el seguimiento de objetos que son relativamente pequeños en la imagen, la representación idónea suele ser la de puntos, por ejemplo, en [69] se utiliza dicha representación para modelar pájaros en la escena. En el caso de que la forma se asemeje a un rectángulo o elipse, lo adecuado es utilizar la representación basada en formas geométricas primitivas, por ejemplo, en [61] se utiliza una elipse para este fin.

#### 2.2.3.2. Métodos de seguimiento

Existen diferentes clasificaciones de los métodos de seguimiento, por ejemplo las que se hacen en [5, 70]. Dado que la elección del modelo de representación está muy relacionado con el algoritmo de seguimiento que se va a utilizar, usaremos el tipo de modelo de objeto para clasificar los métodos de seguimiento, al igual que se hace en [55]:

- **Seguimiento de puntos**

  Para el seguimiento de objetos representados mediante puntos existen fundamentalmente dos tipos de métodos: los métodos determinísticos y los estadísticos.

  Los métodos **determinísticos** abordan la correspondencia entre los objetos del fotograma actual y el anterior como un problema de minimización de costes. De este modo la asignación final puede ser obtenida aplicando algoritmos heurísticos, por ejemplo algoritmos ávidos como se utiliza en [69] o el algoritmo húngaro utilizado en [71]. En general, estos métodos asumen una serie de restricciones que simplifican el problema, entre las más habituales encontramos las siguientes:

  - Los objetos no varían su posición dentro de la escena de forma significativa.
  - Existe una velocidad máxima de los objetos a seguir.
  - La velocidad y la dirección de los objetos varía de forma suave.
  - Los objetos son rígidos, por lo que la distancia entre dos puntos del mismo objeto no varía a lo largo de la escena.

  Los métodos **estadísticos** tratan de estimar el estado de los objetos en el fotograma actual que consiste en propiedades como la posición, la velocidad y la aceleración. Para este fin consideran el estado anterior de cada objeto, las medidas actuales (habitualmente la posición), el error de medición de dichas medidas y el error de estimación del estado.

  Si definimos como $\boldsymbol{X}^t$ el estado en el instante $t$ de un objeto de la escena, la evolución de dicho estado a lo largo de la secuencia viene dada por la siguiente ecuación:

  $$\boldsymbol{X}^t = f^t\left(\boldsymbol{X}^{t-1}\right) + \boldsymbol{W}^t \tag{2.81}$$

  donde $f^t$ es una función del estado del objeto y $\boldsymbol{W}^t$ es ruido blanco.

  El objetivo del seguimiento basado en métodos estadísticos consiste en estimar el estado actual del objeto a partir de las mediciones:

  $$\boldsymbol{Z}^t = h^t\left(\boldsymbol{X}^t, \boldsymbol{N}^t\right) \tag{2.82}$$

donde $h^t$ es una función del estado del objeto y $\boldsymbol{N}^t$ es ruido blanco e independiente de $\boldsymbol{W}^t$.

El filtro de Kalman [72] se puede usar para realizar el seguimiento de un único objeto, asumiendo que $f^t$ y $h^t$ son lineales y tanto el estado inicial $\boldsymbol{X}^1$ como el ruido siguen una distribución uniforme. En caso de que las variables del estado no sigan una distribución gaussiana, el filtro de partículas [73] puede ser una solución para el problema del seguimiento.

El filtro Kalman se compone de dos pasos: predicción y corrección. Durante la predicción se estima el estado actual de la siguiente manera:

$$\overline{\boldsymbol{X}}^t = \boldsymbol{D}\boldsymbol{X}^{t-1} + \boldsymbol{W}^t \tag{2.83}$$

$$\overline{\boldsymbol{\Sigma}}^t = \boldsymbol{D}\boldsymbol{\Sigma}^{t-1}\boldsymbol{D}^T + \boldsymbol{Q}^t \tag{2.84}$$

donde $\overline{\boldsymbol{X}}^t$ y $\overline{\boldsymbol{\Sigma}}^t$ son, respectivamente, las estimaciones del estado y la covarianza. $\boldsymbol{D}$ es la matriz de transición que relaciona los estados en el instante $t-1$ y el $t$. $\boldsymbol{Q}$ es la covarianza del ruido $\boldsymbol{W}$. En el paso de corrección se emplean las mediciones $\boldsymbol{Z}^t$ para actualizar el estado del objeto:

$$K^t = \overline{\boldsymbol{\Sigma}}^t M^T \left(M\overline{\boldsymbol{\Sigma}}^t M^T + R^T\right)^{-1} \tag{2.85}$$

$$\boldsymbol{X}^t = \overline{\boldsymbol{X}}^t + K^t\left(\boldsymbol{Z}^t - M\right) \tag{2.86}$$

$$\boldsymbol{\Sigma}^t = \overline{\boldsymbol{\Sigma}}^t - K^t M\overline{\boldsymbol{\Sigma}}^t \tag{2.87}$$

donde $M$ es la matriz de medidas. En la primera ecuación se calcula la ganancia Kalman, $K^t$, la cual trata de minimizar la covarianza del error del estado estimado. En la segunda ecuación se calcula el valor del estado actual y en la última se calcula la nueva covarianza del error.

- **Seguimiento de regiones**

  El seguimiento de regiones se caracteriza por calcular el movimiento de los objetos entre un fotograma y otro representándolos mediante plantillas o modelos de apariencia.

  En el caso de que los objetos se representen mediante plantillas es posible utilizar un método de fuerza bruta para realizar el emparejamiento a lo largo de la escena. Estos métodos consisten en buscar la mayor similitud entre la plantilla del fotograma anterior y una región dentro de la escena actual. Para ello se calcula una medida de similitud denotada como $h\,(u, v)$, por ejemplo,

(a) (b) (c)

**Figura 2.17:** Ejemplo de seguimiento de un objeto mediante un método basado en *mean-shift* en el que el objeto es modelado con una plantilla elipsoidal. De izquierda a derecha se muestran los fotogramas 30, 75 y 105 de una secuencia deportiva. *Fuente:* [61].

la correlación cruzada entre la plantilla buscada $\mathcal{T}$ y una región de la imagen centrada en el píxel de coordenadas $u \in 1...fil$ y $v \in 1...col$:

$$h\left(u,v\right) = \frac{\sum_{x,y} \mathcal{I}_t\left(u+x, v+y\right) \times \mathcal{T}\left(x,y\right)}{\sqrt{\sum_{x,y} \mathcal{I}_t\left(u+x, v+y\right)^2}} \tag{2.88}$$

donde las sumas se realizan en todas las coordenadas de la plantilla. Nótese que se ha simplificado la ecuación anterior utilizando un único canal de color. Un valor alto de $h\left(u,v\right)$ indica que existe una alta similitud entre la plantilla buscada y la región enmarcada en las coordenadas $(u,v)$. En general, se puede utilizar la intensidad del color para formar las plantillas, pero al ser sensible a los cambios de iluminación, se han propuesto trabajos que utilizan otros rasgos, por ejemplo el gradiente de la imagen [74].

La mayor limitación de los métodos de fuerza bruta es su alto coste computacional, para atenuar este problema existen propuestas como [75] u otras basadas en *mean-shift* y flujo óptico, las cuales expondremos a continuación.

Los métodos de seguimiento basados en el algoritmo *mean-shift* suponen una mejora respecto de los basados en fuerza bruta. Este es el caso de [61] en donde se utiliza el histograma de una región o plantilla elipsoidal para representar al objeto (ver Figura 2.17). En lugar de realizar una búsqueda en toda la imagen, se busca en las proximidades de donde se estima que debe estar el objeto. Inicialmente la posición candidata es la que ocupaba el objeto en el fotograma anterior, en las siguientes iteraciones se busca en la zona indicada por el vector *mean-shift*. Para saber si la posición que se está estudiando se acerca más a la posición actual del objeto se calcula una medida de similitud dada por la siguiente expresión:

$$\sum_{u=1}^{m} \mathcal{P}\left(u\right) \mathcal{Q}\left(u\right) \tag{2.89}$$

donde $m$ es el número de cubos de los histogramas, $\mathcal{P}$ es el histograma del objeto y $\mathcal{Q}$ es el de la posición candidata.

Otros métodos para el seguimiento son los basados en el flujo óptico. Este tipo de propuestas consiste en calcular el vector de flujo de todos los píxeles de la escena, el cual es calculado asumiendo la condición del brillo constante [76]: $\boldsymbol{I}_t(p) - \boldsymbol{I}_{t+\nabla t}(p + \nabla p) = 0$. Una propuesta similar es la utilizada por el método de seguimiento SKT (*Shi, Kanade y Tomasi*) [49]. Este método calcula iterativamente la traslación de una región centrada en un punto de interés. Una vez que se determina la ubicación del punto de interés, el método evalúa la calidad del seguimiento calculando la transformación afín.

- **Seguimiento de siluetas y contorno**

  Algunos de los objetos a seguir pueden tener formas complejas como por ejemplo cabeza, brazos y piernas, que no se pueden modelar de forma exacta con una figura geométrica simple. Los métodos basados en la silueta o en el contorno eluden esta limitación y proporcionan una descripción ajustada de los objetos.

  Para el seguimiento basado en **contorno** es necesario adaptar el contorno inicial de cada objeto a la nueva posición dentro del fotograma actual. El flujo óptico es una de las opciones elegidas para realizar dicha tarea. Por ejemplo en [77, 78] se utiliza para este fin. En [79] se utiliza para calcular el flujo de los píxeles que hay dentro de un objeto y posteriormente se evalúa la energía del contorno, que también se basa en la condición del brillo constante.

  El seguimiento basado en la **silueta** se puede realizar de forma similar al seguimiento de plantillas. En primer lugar, se extraen las siluetas mediante un método de detección de movimiento. A continuación se realiza el emparejamiento entre los objetos del fotograma anterior y el actual, utilizando como criterio alguna medida de similitud entre las siluetas. Por ejemplo en [80] se modelan los objetos utilizando la información de los bordes del interior de las siluetas. En [81] se utilizan histogramas de color de las siluetas que permiten caracterizar cada objeto.

  En lugar de emparejar las siluetas de cada par de fotogramas consecutivos sin más información que las siluetas en sí, el seguimiento puede refinarse si se calcula el flujo dominante dentro de cada una de estas siluetas. De esta manera es posible determinar la trayectoria de cada silueta con el fin de afinar la búsqueda, este es el caso de [82]. En [83] se calcula el flujo óptico de las siluetas de las personas en movimiento utilizando un método basado en el análisis de la varianza.

### 2.2.4. Reconocimiento de acciones

El reconocimiento de acciones es una de las etapas de mayor nivel dentro del proceso de videovigilancia automática. Su objetivo consiste en determinar qué acciones realiza cada uno de los objetos que hay en la escena.

La capacidad de reconocer las acciones que se están llevando a cabo en un vídeo permite desarrollar sistemas con una amplia variedad de aplicaciones prácticas. El reconocimiento de acciones puede ser de gran utilidad para gestionar de forma automática los sistemas de videovigilancia de espacios públicos como aeropuertos o estaciones, por ejemplo, acciones como dejar abandonada una maleta o meter una bolsa en una papelera, pueden ser consideradas sospechosas. La monitorización de pacientes, niños o personas mayores también se puede beneficiar de este tipo de procesamiento, facilitando la detección de acciones o gestos potencialmente peligrosos.

Las actividades que realizan los objetos de la escena se pueden clasificar en función de la complejidad y el número de objetos implicados. La actividad más básica es el gesto, consiste en un movimiento de una parte del cuerpo, por ejemplo levantar una pierna o mover una mano. Una serie de gestos a lo largo del tiempo conforman una acción, por ejemplo saludar con la mano, correr, andar o saltar. El siguiente paso consiste en la interacción entre más de un objeto, por ejemplo dos personas peleando o alguien robando una maleta. Y por último estaría la interacción entre grupos de objetos, por ejemplo personas protestando ante la policía en una manifestación o vehículos detenidos ante el paso de otros. En general, el reconocimiento de acciones contempla individualmente cada una de las acciones que desarrolla un único objeto, mientras que el análisis de comportamiento es una etapa más general en la que se pretende describir el comportamiento de los objetos de la escena, es decir, no se estudia un único objeto ni una única acción, se pueden analizar varias acciones y la interacción entre varios objetos.

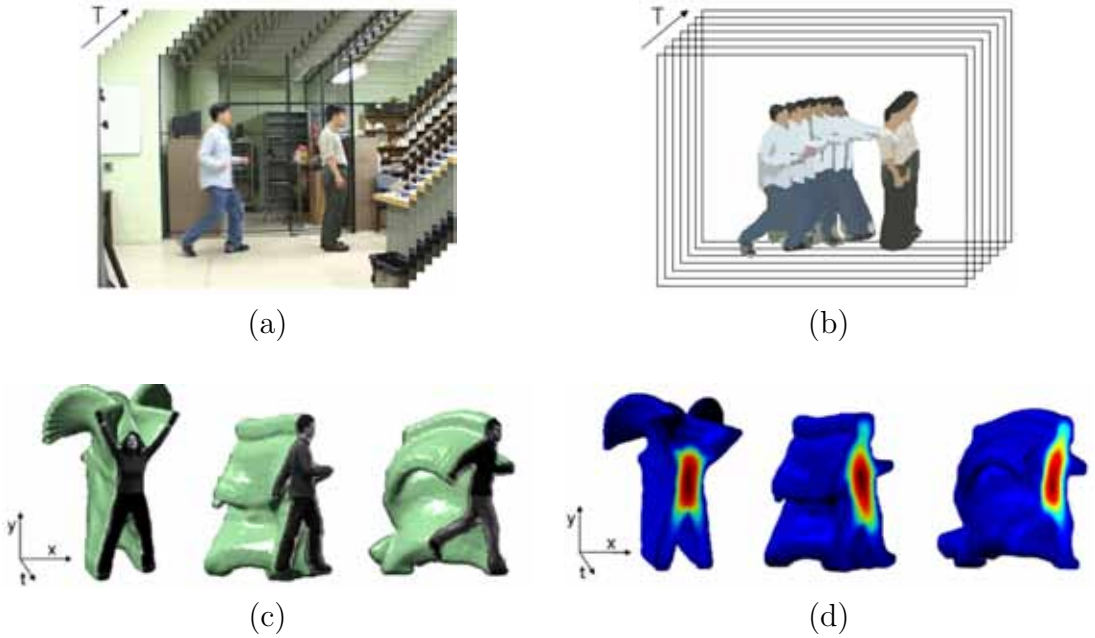
#### 2.2.4.1. Métodos de reconocimiento de acciones

Los métodos de reconocimiento de acciones pueden ser clasificados en jerárquicos y no jerárquicos, tal y como se hace en [84]. Los métodos jerárquicos utilizan un sistema de capas para describir las actividades. Cada capa describe las actividades mediante acciones más simples, habitualmente denominadas sub-eventos. De este modo existen métodos compuestos de varias capas que pueden describir actividades complejas. Los métodos no jerárquicos analizan directamente la secuencia de imágenes que compone el vídeo de entrada, son adecuados para reconocer gestos y actividades de naturaleza secuencial. A continuación expondremos algunos de los métodos de reconocimiento de acciones más representativos basándonos en la taxonomía anteriormente citada:

- **Métodos no jerárquicos**

  Los modelos no jerárquicos consideran cada acción como un tipo particular de secuencia de imágenes, por lo que el reconocimiento consiste en clasificar el

tipo de secuencia que representa el vídeo de entrada. Hay dos tipos de métodos no jerárquicos: las aproximaciones espacio temporales y las secuenciales.




(a) (b)

(c) (d)

**Figura 2.18:** Acciones representadas utilizando un enfoque espacio temporal. (a) es la secuencia de imágenes original para la acción de golpear a una persona, (b) es el blob correspondiente a identificar o aprender. (c) y (d) muestran seis ejemplos de acciones representadas en el espacio XYT. *Fuente:* [84, 85]

Un vídeo es una secuencia de imágenes bidimensionales ordenadas en función del tiempo. En las **aproximaciones espacio temporales** se utiliza un espacio tridimensional, denominado XYT, en el que dos dimensiones están formadas por el ancho y el alto de la imagen, mientras que la tercera es el tiempo. Con esta premisa, los métodos espacio temporales utilizan un conjunto de secuencias de entrenamiento que representan acciones. Estas secuencias son convertidas al espacio XYT con el fin de aprender la correspondencia entre cada acción y su representación dicho espacio (ver Figura 2.18). Una vez entrenado el modelo, las secuencias de entrada son convertidas al espacio XYT. A continuación se intenta deducir el tipo de acción que aparece, basándose en alguna medida de similaridad entre las secuencias de entrenamiento aprendidas y la de entrada.

Las secuencias convertidas al espacio XYT pueden ser consideradas volúmenes en los que para reconocer las acciones hay que asociar un volumen a otro. Es el caso de [86] en el que se generan filtros con los que capturar las características de dichos volúmenes y así poder emparejarlos de forma fiable y eficiente. [87] aplica SVM para reconocer acciones considerando la forma y el flujo de los objetos. [88] propone un algoritmo basado en el emparejamiento de plantillas. En lugar de utilizar volúmenes para representar cada acción, utili-

zan una plantilla compuesta de dos imágenes: una imagen binaria denominada MEI (*motion-energy image*) y una imagen con valores escalares denominada MHI (*motion-history image*). Estas imágenes se construyen a partir de una secuencia de imágenes segmentadas, básicamente proyecciones 2D de los volúmenes XYT referidos anteriormente. Los píxeles ocupados recientemente por los objetos tienen un valor mayor que los más antiguos. Aplicando técnicas de emparejamiento de plantillas es posible reconocer acciones simples como saludar con la mano, agacharse o sentarse.

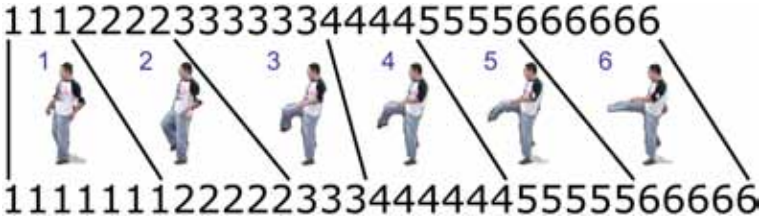
La trayectoria de un objeto puede representarse en un espacio tridimensional XYZ o bien XYT. Basándose en este enfoque, existen diversas soluciones para el problema de reconocimiento de acciones, por ejemplo [89, 90].

Otra aproximación basada en el espacio XYT consiste en extraer rasgos de los volúmenes generados por los objetos en movimiento con el fin de caracterizar las acciones. Esencialmente los volúmenes generados al representar una secuencia en el espacio XYT son objetos 3D rígidos. Los rasgos pueden ser combinados para representar acciones considerando su relación espacio temporal. Ejemplos de este enfoque son trabajos como [90, 91, 92].

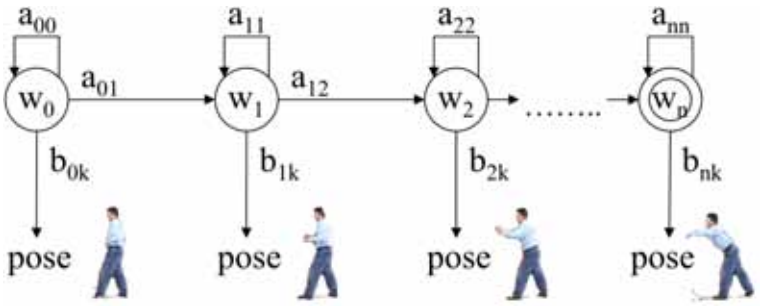
En las **aproximaciones secuenciales** se considera el vídeo de entrada como una secuencia de rasgos, reconociendo las acciones en función del tipo de secuencia observada. En primer lugar, se convierte la secuencia de imágenes en una secuencia de vectores de rasgos. Posteriormente, se calcula la probabilidad de que estos rasgos correspondan a una acción en particular, si la probabilidad es suficientemente alta, el sistema considera que esa acción ha ocurrido.

El algoritmo DTW (*dynamic time warping*), que originalmente fue desarrollado para el reconocimiento del habla, es muy utilizado para el reconocimiento de acciones. Su funcionamiento consiste en emparejar dos secuencias de rasgos con pequeñas variaciones, debe tenerse en cuenta que la misma acción no siempre se realiza de forma idéntica (ver Figura 2.19.a). Algunos de los trabajos que utilizan DTW son [93, 94, 95]. Una de las ventajas del algoritmo DTW es que encuentra una solución óptima al problema de emparejar dos secuencias con una complejidad polinómica.

Una aproximación secuencial que no está basada en el reconocimiento de secuencias de rasgos consiste en modelar el desarrollo de las acciones a lo largo de un vídeo como una secuencia de estados por los que transitan los objetos. Para este enfoque se utilizan habitualmente los modelos ocultos de Markov [96, 97, 98] y las redes bayesianas dinámicas [99, 68]. Tanto en un caso como en otro, la acción se representa como un conjunto de estados ocultos conectados entre sí mediante transiciones (ver Figura 2.19.b). En cada fotograma se asume que todos los objetos están en un estado determinado y que cada estado genera una salida, es decir, una observación o vector de rasgos. El objeto transita de un estado a otro en función de la probabilidad de la transición que asocia a ambos estados. En general las acciones son reconocidas resolviendo




(a)




(b)

**Figura 2.19:** Acciones representadas mediante aproximaciones secuenciales. (a) es un ejemplo de dos secuencias de rasgos para la misma acción: extender una pierna. Los rasgos están representados por los números del 1 al 6. (b) es un ejemplo de modelo oculto de Markov que se corresponde con la acción de extender un brazo. Cada estado representa una de las poses que componen la acción. *Fuente:* [84]

un problema de evaluación. La evaluación consiste en calcular la probabilidad de que una secuencia dada sea generada por un modelo de estado en concreto. Si la probabilidad es suficientemente alta, se asume que ha sucedido la acción correspondiente a ese modelo de estado.

- **Métodos jerárquicos**

  Los métodos jerárquicos reconocen actividades complejas utilizando para ello una serie de capas. Las capas de menor nivel reconocen actividades básicas, mientras que las capas superiores utilizan la información de las capas inferiores para reconocer acciones más complejas. Por ejemplo, para reconocer la acción de "pelearse", se pueden detectar en las capas inferiores acciones como "dar puñetazos" o "dar patadas". Las acciones de capas inferiores se denominan sub-eventos y son tratados por las capas superiores como observaciones. En los métodos jerárquicos es habitual utilizar métodos no jerárquicos para implementar las capas inferiores. Básicamente existen tres tipos de métodos jerárquicos: estadísticos, sintácticos y basados en descripción.

  Los métodos **estadísticos** utilizan modelos estadísticos basados en estados para el reconocimiento de las acciones. Normalmente se utilizan dos capas de modelos estadísticos basados en estados, por ejemplo modelos ocultos de Markov [101, 97, 102] o redes bayesianas dinámicas [103, 104]. La primera capa

```
StretchHand(i) = atomic_action
        ((person i's hand, stretch, other person));
StayStretchedHand(i) = atomic_action
        ((person i's hand,stay stretched, other person's hand));
WithdrawHand(i) = atomic_action ((person i's hand, withdraw, null));

ShakeHandsAction(i) = (
        list( def(x,StretchHand(i)),
                list( def(y,StayStretchedHand(i)),
                        def(z,WithdrawHand(i)))
        ),
        and( meets(x,y),
                and( meets(y,z),
                        and(starts(x,this),finishes(z,this)))
        )
);
```

**Figura 2.20:** Posible definición de la acción "darse la mano" usando la sintaxis del método basado en descripción propuesto en [100].

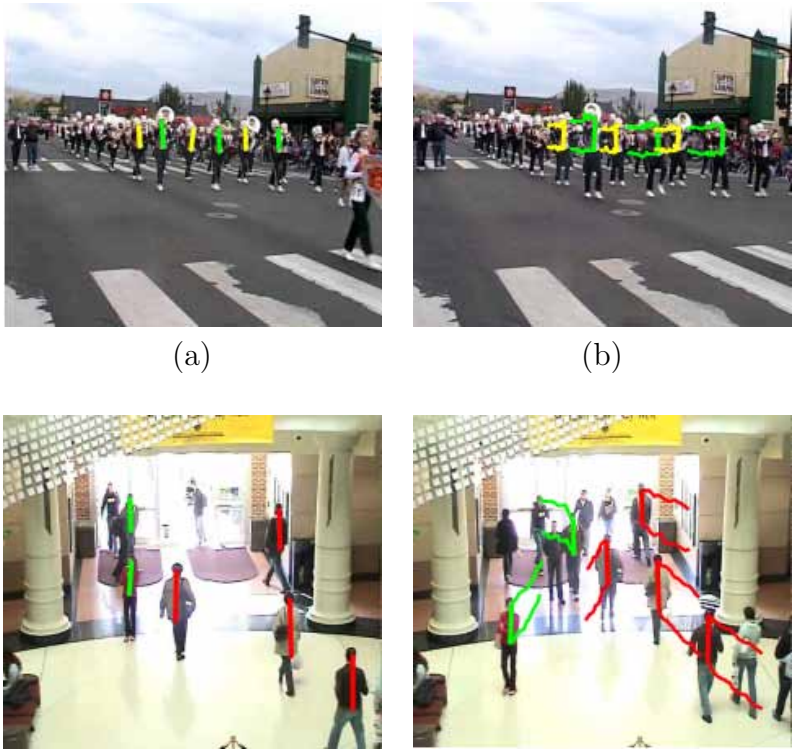
reconoce acciones básicas a partir de los vectores de rasgos y la segunda capa toma estas acciones como si fueran observaciones. Para determinar qué acción se está desarrollando, se debe calcular la probabilidad de que las observaciones, es decir, los vectores de rasgos o acciones básicas según el caso, sean generadas por un modelo en concreto.

Los métodos **sintácticos** representan las acciones como cadenas de símbolos. Cada símbolo se corresponde con una acción básica. Al igual que en los métodos estadísticos, las acciones básicas son reconocidas utilizando alguno de los métodos expuestos previamente. Para reconocer las acciones complejas que representan las cadenas de símbolos, se utilizan técnicas propias del campo de los procesadores de lenguajes. En concreto es habitual el uso de gramáticas de contexto libre o CFG (*context free grammars*) y gramáticas de contexto libre estocásticas o SCFG (*stochastic context free grammars*) [105, 106, 107].

Los métodos basados en **descripción** representan las actividades complejas realizadas por los objetos de la escena, relacionando entre sí actividades más simples mediante relaciones temporales, espaciales y lógicas. Por lo tanto las acciones complejas son detectadas cuando las acciones simples satisfacen ciertas condiciones. En los métodos basados en descripción se definen intervalos de tiempo para relacionar la ocurrencia de diferentes acciones simples. Para describir formalmente cada acción compleja se utilizan habitualmente gramáticas de contexto libre. Nótese que estas gramáticas se utilizan para utilizar una sintaxis concreta con la que describir las acciones, lo que es diferente del uso que se le da en los métodos sintácticos, ya que en estos últimos se utiliza como herramienta para describir la semántica de las acciones. Algunas de las propuestas para el reconocimiento de acciones basadas en descripción son: [108, 109, 100, 110]. A modo de ejemplo, una definición posible de la acción "darse la mano" es la que se muestra en Figura 2.20.

### 2.2.5. Análisis de comportamiento

A diferencia de la etapa anterior, el análisis de comportamiento no busca acciones individuales, si no que se analiza la relación entre varias acciones con la posibilidad de que intervengan varios objetos, lo que se denomina comportamiento. Así pues, el objetivo de esta etapa es la detección de aquellos comportamientos que son de interés para el usuario final. Un uso habitual de este tipo de sistemas consiste en aumentar la seguridad de sitios públicos o privados, por ejemplo detectando comportamientos hostiles o anómalos [111]. Casos típicos de comportamientos hostiles son, por ejemplo, una agresión entre personas o el robo de un objeto. Para lograr este objetivo es habitual describir las interacciones entre los objetos de la escena a alto nivel [112]. De esta manera se pretende especificar de forma inequívoca comportamientos complejos que involucren una o varias acciones y también uno o varios objetos.



(a) (b)

(c) (d)

**Figura 2.21:** Ejemplo de comportamiento de grupos de personas. En (a) y (b) se muestra un desfile, los rasgos del grupo de personas se desplazan en la misma dirección. Los fotogramas (c) y (d) corresponden a una secuencia capturada en el interior de un edificio concurrido, los rasgos del grupo de personas no se desplazan en la misma dirección. *Fuente:* [113]

Una posible taxonomía de los métodos de análisis de comportamiento consiste en

diferenciar aquellos métodos orientados a la interacción bien entre humanos y objetos o bien entre grupos de personas.

Los métodos tradicionales que analizan la interacción entre **humanos y objetos** realizan las etapas de segmentación, reconocimiento, estimación de movimiento, seguimiento y reconocimiento de acciones. En general en estos trabajos se ignora la relación entre el reconocimiento de los objetos y el movimiento, es decir, primero se reconocen los objetos y después se analiza el movimiento de cada uno de ellos. La mayoría de los métodos citados previamente caen en esta categoría [108, 102, 104].

En los últimos años también han surgido métodos que realizan la detección de movimiento directamente sin realizar las etapas previas [114, 115]. De forma similar hay propuestas que analizan las relaciones y dependencias entre los objetos, el movimiento y las actividades que realizan [116, 117, 118]. Dado que todos estos componentes están muy relacionados entre sí, por ejemplo, el tipo de objeto está relacionado con el uso que se le va a dar y por tanto la manera en la que el humano interacciona con el. Una persona interaccionará de forma distinta con una botella de agua que con una botella de espray, en el primer caso es posible que se relacione con la acción de beber, mientras que en segundo caso no será así. Los resultados obtenidos al utilizar este enfoque indican que en la práctica el reconocimiento de objetos se beneficia del análisis de comportamiento y viceversa.

El comportamiento de los **grupos de personas** queda definido por el movimiento de los distintos actores que componen los grupos presentes en la escena. Por ejemplo, si tenemos dos secuencias de una misma calle donde en un caso los individuos se desplazan en la misma dirección, mientras que en el otro caso no existe una única trayectoria mayoritaria, es posible que la primera situación represente una manifestación, mientras que la segunda se corresponda con una situación habitual de personas transitando la calle.

Existen dos enfoques distintos pare llevar a cabo el análisis de comportamiento de los grupos de personas: considerar el comportamiento de cada uno de los individuos que componen el grupo o bien el comportamiento global del mismo.

Los métodos que consideran el comportamiento de cada uno de los individuos que componen el grupo suelen utilizar una aproximación jerárquica, por ejemplo mediante modelos de Markov o redes bayesianas dinámicas [97, 103, 119]. La idea es que el nivel inferior reconozca el comportamiento de cada individuo y el superior el del grupo. Por ejemplo, si el vídeo de entrada muestra una secuencia donde se está llevando a cabo una presentación, habrá varias personas, pero el primer nivel determinará que uno de los individuos habla y el resto presta atención, toma notas o realiza preguntas, por lo que el nivel superior debe concluir que ese grupo de personas está en una presentación.

A diferencia del caso anterior, en los métodos que estudian el movimiento global del grupo no es habitual usar un enfoque jerárquico. Por ejemplo, en [120] se emplean una serie de puntos de interés que determinan la forma geométrica que caracteriza al grupo de personas. Este trabajo estudia el caso concreto de los grupos de pasajeros

que salen de un avión y que deberían caminar hacia la terminal. La evolución de los puntos de interés a lo largo del tiempo permite detectar actividades anómalas, por ejemplo, si algún pasajero se detiene y no camina hacia la terminal. En el mismo sentido en [113] se extraen rasgos de los grupos de personas que aparecen en la escena. A partir de estos puntos se construye un polígono 3D y se estudia la rigidez de dicho polígono a lo largo del tiempo. El objetivo de este trabajo es determinar cuando las personas que aparecen en la escena están desfilando, es decir, desplazándose en la misma dirección y manteniendo una distancia constante entre cada individuo. En caso de que el polígono mantenga una rigidez elevada se considera que el grupo de personas forma parte de un desfile, si por el contrario cada individuo sigue una trayectoria diferente de manera que el polígono no es rígido a lo largo de la secuencia, se asume que los individuos no están desfilando. En la Figura 2.21 se muestran dos fotogramas de una secuencia donde se está desarrollando un desfile y otros dos de una secuencia donde cada individuo se desplaza con una trayectoria diferente.

## 2.3. Modelado del fondo

La detección del movimiento es la etapa más básica de la videovigilancia y su relevancia es muy elevada ya que el resto de etapas dependen en gran medida de la calidad de sus resultados. Por tanto es deseable que se detecten con precisión los objetos del primer plano, pero también que el tiempo necesario para ejecutar el método de detección sea el menor posible. De este modo, el resto de etapas que componen la videovigilancia podrán ejecutarse correctamente y manteniendo los requisitos de tiempo real.

El problema de la detección de fondo consiste en determinar si cada uno de los elementos que componen la imagen pertenecen al primer plano o al fondo. Por lo tanto, para diseñar un método de detección de movimiento es necesario elegir el tamaño de dichos elementos. Este tamaño es un factor que repercute tanto en la calidad del resultado como en el tiempo de procesamiento. Básicamente una secuencia se puede analizar a nivel de píxel, de bloque o de región. Generalmente los métodos que trabajan a nivel de píxel obtienen resultados más precisos que los que utilizan elementos más grandes, sin embargo también son más sensibles al ruido y requieren mayor tiempo que si utilizasen un elemento mayor.

Otro aspecto a tener en cuenta para el diseño de un método de detección de movimiento radica en el tipo de rasgos o características que se van a analizar. En la literatura se utilizan diferentes rasgos y, siguiendo la clasificación hecha en [121], pueden ser clasificados en tres tipos: espectrales, espaciales y temporales. Los primeros hacen referencia al espacio de color utilizado, como vimos en la Sección 2.1 el modelo RGB es el más utilizado para codificar las imágenes, sin embargo no tiene unas características que permitan comparar colores de manera adecuada (no es uniforme y sus componentes están muy correlacionadas). Por ello es habitual utilizar espacios de color distintos al RGB para realizar la detección de movimiento. Utilizar

características espaciales como las texturas facilita que los métodos sean inmunes a problemas relacionados con el color, por ejemplo, la iluminación. De forma similar el estudio de las características temporales de un píxel puede facilitar su modelado, por ejemplo, en el caso de que se produzcan alteraciones periódicas en su color.

Al margen de los rasgos y del tamaño del elemento utilizados por el método de detección de fondo, el aspecto más determinante es el modelo de fondo. El modelo afecta a todas las etapas internas de la detección de movimiento: detección de primer plano, actualización del modelo y en menor medida al post-procesado. Emplear un modelo de fondo en concreto condiciona la calidad de la detección ya que los métodos de detección se basan en buscar diferencias entre el modelo y la imagen actual, por lo que si el fondo real no se corresponde con el modelo, difícilmente se podrá implementar un método que detecte los objetos en movimiento. El tiempo de ejecución también se ve afectado por el modelo elegido, si un modelo es muy complejo, es probable que se necesite mucho tiempo para su actualización o incluso para la propia detección. En este sentido los modelos de fondo básicos expuestos en la Subsección 2.2.1.3 son una aproximación fácil de implementar y con unos requisitos de tiempo generalmente bajos. No obstante, en la realidad existen situaciones en las que no son lo suficientemente robustos como para modelar el fondo de manera precisa. Es por ello que en los últimos años han surgido numerosas propuestas que tratan de superar estas dificultades. Algunos trabajos definen modelos que ofrecen un mejor rendimiento en general que los modelos básicos, mientras que otros están enfocados para afrontar situaciones problemáticas concretas.

En la Subsección 2.3.1 haremos un estudio sobre las dificultades típicas que suelen experimentar los métodos de detección de movimiento. Seguidamente en la Subsección 2.3.2 repasaremos los distintos tipos de modelos que actualmente se utilizan, mostrando ejemplos de modelos concretos. Por último, en la Subsección 2.3.3, describiremos técnicas de post-procesado que permiten mejorar los resultados de los métodos de detección de movimiento.

### 2.3.1. Dificultades

La detección de movimiento es un proceso en el cual una escena es digitalizada por un dispositivo de entrada y el vídeo resultante es procesado por un método de detección. De este modo, para un método de detección en particular, los problemas que pueden dificultar la detección tienen dos orígenes distintos: el proceso de captura de vídeo y la complejidad de la escena real. El primero puede ser corregido cambiando o mejorando las condiciones y dispositivos de captura. No obstante no siempre es posible realizar estos cambios, por lo que es deseable que los métodos de detección sean robustos ante este tipo de problemas. El segundo origen es inalterable ya que es intrínseco al problema que se intenta resolver.

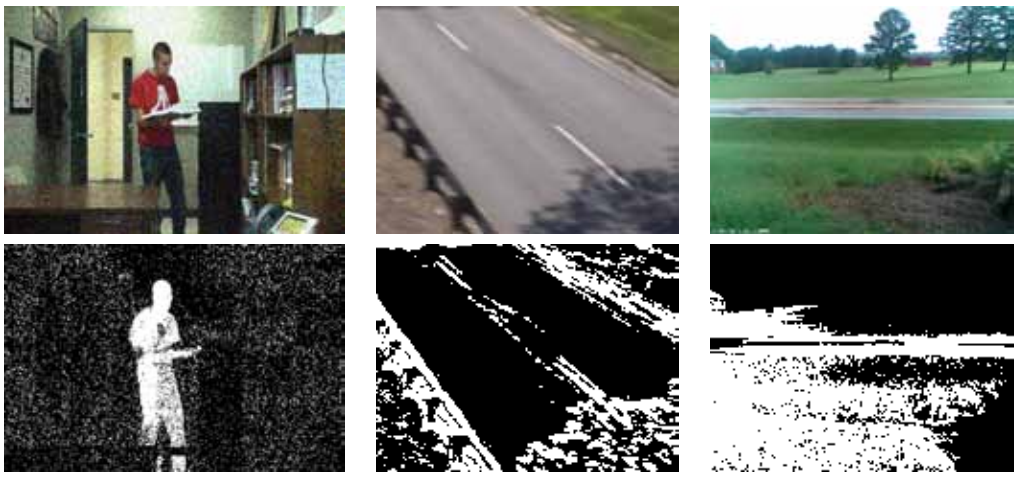
La situación más favorable para los métodos de detección movimiento es aquella en la que el fondo es estático y muy diferente de los objetos, la iluminación es

constante, la cámara es estática y la captura no sufre distorsiones. Si la secuencia tiene estas características, lo habitual es que la mayoría de los métodos logren buenos resultados. Sin embargo, en la práctica es muy difícil encontrar una secuencia de estas características. En [122] se exponen los problemas típicos que dificultan la detección de movimiento. A continuación expondremos estas y otras situaciones problemáticas clasificándolas en base a su origen: el proceso de captura y la dificultad intrínseca de la escena.

#### 2.3.1.1. Dificultades a causa del proceso de captura

El proceso de captura de vídeo es la primera etapa de cualquier sistema de procesamiento digital. Se realiza utilizando un dispositivo de entrada de vídeo, en nuestro caso lo habitual será una cámara de videovigilancia. Gracias a este proceso, la secuencia real es transformada en una secuencia virtual o vídeo. La fidelidad del vídeo capturado es un factor clave para realizar una buena detección. Cualquier diferencia entre la secuencia real y la virtual puede hacer que la salida del método no se corresponda con la realidad. En este sentido existen tres tipos de alteraciones relacionadas con el proceso de captura que pueden reducir el rendimiento de los métodos de detección de movimiento. A continuación se describen estos tipos de alteraciones y en la Figura 2.22 se muestra un ejemplo concreto de cada uno de ellos:

- **Ruido**: El proceso de digitalización consiste en la adquisición, registro y transmisión de vídeo. En las etapas de adquisición y registro está implicado esencialmente el sensor de la cámara. En general estos sensores no sólo son sensibles a la luz, si no que también lo son al calor o al tiempo de exposición, lo que distorsiona en ocasiones el vídeo capturado. La transmisión sobre líneas con ruido también es un factor que afecta a la fidelidad de escena la capturada.
- **Vibraciones de la cámara**: Es habitual que se utilicen cámaras de videovigilancia en exteriores, por ejemplo en calles, autovías o vías de comunicación en general. En estos casos las cámaras pueden sufrir vibraciones debido al viento generado de forma natural o a causa de los propios vehículos. Por tanto el vídeo capturado mostrará desplazamientos más o menos grandes de la escena e incluso desenfoque de la imagen. Si los métodos no son suficientemente robustos en este aspecto, es normal que se detecten como primer plano zonas que realmente pertenecen al fondo.
- **Auto ajuste de la cámara**: Las cámaras actuales realizan una serie de ajustes automáticos para corregir el enfoque, el balance de blancos o la iluminación. Si bien estas técnicas mejoran la calidad de la imagen y facilitan diferenciar los objetos, los métodos de detección, especialmente aquellos que utilizan modelos basados en el color, son muy sensibles en este aspecto al no ser capaces de distinguir entre estos ajustes automáticos y los cambios producidos por la dinámica real de la escena.



(a) (b) (c)

**Figura 2.22:** Ejemplos de situaciones problemáticas a causa del proceso de captura de vídeo. La primer fila muestra el fotograma de entrada y la segunda fila el resultado de la detección. En (a) se muestra una escena con ruido, (b) es una secuencia donde la cámara sufre desplazamientos pequeños pero bruscos y (c) es el caso de una corrección automática de brillo. *Fuente: elaboración propia.*

#### 2.3.1.2. Dificultades propias de la escena

El rendimiento de los métodos de detección está condicionado por la siguiente suposición: existen diferencias entre la imagen actual y el modelo de fondo si y solo si hay un objeto en movimiento. Así pues, los métodos deben procurar que todas las diferencias detectadas estén asociadas a objetos en movimiento. En sentido contrario, si no se detectan diferencias no debe haber objetos en movimiento. Sin embargo, incluso asumiendo que la entrada de vídeo capturada representa fielmente a la escena real, existen situaciones problemáticas que dificultan que se satisfaga la suposición anterior. El origen de estas situaciones está tanto en la propia dinámica de la escena como en las características del método de detección. A continuación se describen los casos más habituales y en la Figura 2.23 pueden verse ejemplos concretos:

- **Iluminación**: Se debe a que la luz que ilumina la escena varía a lo largo de la secuencia. Al cambiar la luz, el color de los píxeles que componen la escena también cambia, por lo que el método puede detectar estos cambios como primer plano. Existen dos tipos:
  - Cambios graduales: se produce un cambio gradual de la iluminación, por ejemplo, el aumento y disminución de la luz solar en una escena exterior como consecuencia del paso de las horas.
  - Cambios bruscos: se producen al encender o apagar una luz en una habitación o con el paso de las nubes en una escena exterior. Los métodos

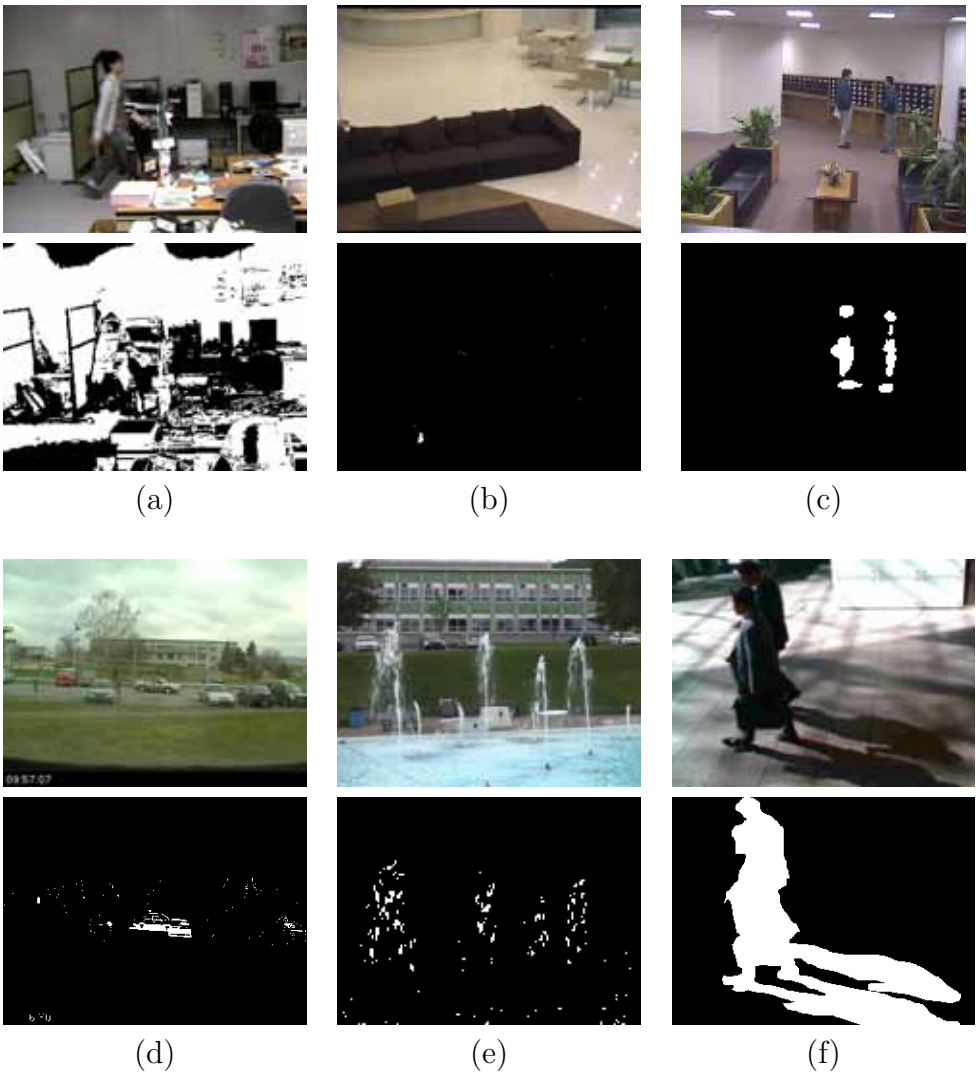


**Figura 2.23:** Ejemplos de situaciones problemáticas a causa de la dificultad de la escena. La primera y tercera filas muestran el fotograma de entrada, la segunda y cuarta filas muestran el resultado de la detección. En (a) (iluminación) se representa un fotograma en el que se han apagado parte de las luces, en (b) (*sleeping person*) la caja lleva unos segundos abandonada en el suelo, (c) es un caso típico de camuflaje en donde la ropa de las personas se confunde con el fondo de la escena, en (d) (*walking person*) la zona donde estaba aparcado el vehículo es detectada como primer fondo, (e) muestra una secuencia con fondo dinámico y (f) es un ejemplo de sombras proyectadas. *Fuente: elaboración propia.*

de detección de movimiento suelen tener más problemas con este tipo de cambio de iluminación que con los graduales.

- **Camuflaje**: Es un tipo de artefacto que se produce cuando las características del fondo y de los objetos del primer plano son parecidas. Cuando se da esta circunstancia suelen marcarse como fondo los píxeles que realmente pertenecen al primer plano. Por ejemplo, supongamos que un método se basa en buscar diferencias de color entre el modelo y la imagen de entrada, si la ropa de una persona es del mismo color que el fondo, esta sera marcada como fondo ya que no encuentra diferencias significativas.
- **Inicialización deficiente** (*bootstrapping*): Se caracteriza por generar un modelo de fondo inicial que no representa correctamente el fondo real de la escena. Es habitual cuando en los fotogramas dedicados a inicializar el modelo, no está representado el fondo real de la escena, por ejemplo porque existen objetos en movimiento. En este caso el modelo considera las características de los objetos del primer plano como propias del fondo. Mientras permanezcan dichos objetos en escena no serán detectados y cuando ya no estén, se marcará la zona donde estuviesen como parte del primer plano, lo que se conoce como regiones fantasma.
- **Fondos dinámicos**: El fondo de la escena puede no ser estático como consecuencia del movimiento periódico de los elementos que lo componen. Casos típicos son el movimiento de las hojas y ramas de los árboles, las olas del mar o el agua que emana de una fuente. En general los métodos de detección deben ser capaces de ignorar estos movimientos y no detectarlos como objetos del primer plano.
- **Objetos del primer plano parados** (*sleeping person*): Consiste en un objeto del primer plano que en un momento dado deja de estar en movimiento. Pasado un tiempo el objeto puede ser considerado parte del fondo. En estos casos depende del contexto si se debe considerar como parte del fondo o del primer plano. Por ejemplo, un coche que aparca en un aparcamiento quizá deba ser considerado parte del fondo, pero una maleta que es abandonada en la terminal de un aeropuerto habitualmente debe ser detectada como primer plano para tomar las medidas de seguridad oportunas.
- **Objetos de fondo en movimiento** (*walking person*): Sucede cuando un objeto que permanece estático comienza a moverse, por ejemplo, una persona sentada que se levanta y comienza a andar. Si bien el objeto que comienza a moverse es detectado, los píxeles que ocupaba cuando permanecía estático se marcan erróneamente como primer plano. Es otra situación donde se producen regiones fantasma.
- **Sombras**: Existen dos tipos de sombras que pueden ser potencialmente problemáticas. El primer tipo tiene lugar cuando un objeto del primer plano proyecta una sombra que se desplaza a la par que él. Esto se conoce como sombra proyectada (*cast shadow*s) y no debe ser considerada como parte del primer plano.

En segundo lugar, también se proyectan sombras a causa de los cambios de iluminación. En este sentido, tanto los cambios de iluminación graduales como los bruscos pueden producir sombras que no tienen que estar relacionados con un objeto del primer plano.

### 2.3.2. Modelos de fondo actuales

Los modelos de fondo básicos son una buena alternativa si las escenas a tratar son sencillas, sin embargo a lo largo de una secuencia pueden acontecer situaciones complejas que mermen su rendimiento. Por ello han surgido numerosos trabajos que proponen modelos de fondo más sofisticados y robustos. Algunos están orientados a obtener mejores resultados que los modelos básicos en general, mientras que otros abordan problemáticas concretas. A continuación haremos una descripción de los tipos de modelos más utilizados, detallando también en cada caso cómo se utilizan para detectar el movimiento.

#### 2.3.2.1. Modelos gaussianos

En el modelo basado en la media (Subsección 2.2.1.3) se utiliza la media de las intensidades pasadas de un píxel para modelar el fondo: $\boldsymbol{B}_t(p) = \boldsymbol{\mu}_t(p)$. Así pues, se está asumiendo que las intensidades que puede experimentar un píxel del fondo a lo largo de la escena se encuentran dentro de un intervalo centrado en la media de dichas intensidades: $[\boldsymbol{\mu}_t\,(p) - T,\ \boldsymbol{\mu}_t\,(p) + T]$. Las escenas cuyo fondo es muy estático pueden ser bien segmentadas utilizando un $T$ pequeño, ya que cualquier intensidad fuera de dicho intervalo se corresponde con un objeto en movimiento. Sin embargo como ya hemos visto, esto exige que el umbral sea ajustado para adaptarse a la variabilidad del fondo característica de cada secuencia. Por otro lado, existen situaciones en las que un único $T$ para toda la escena puede ser problemático, por ejemplo, cuando algunos píxeles del fondo experimentan variaciones bruscas en su intensidad mientras que otros no. Por ello en la práctica no es una buena opción utilizar un umbral fijo y común para todos los píxeles.

Una de las formas de modelar el fondo de la escena que permite subsanar parte de las limitaciones del modelo basado en la media, consiste en asumir que la intensidad del fondo a lo largo de la secuencia es una variable aleatoria que sigue una distribución gaussiana. El método **Pfinder** propuesto en [57] fue uno de los primeros en aplicar este enfoque al utilizar distribuciones gaussianas para modelar las intensidades tanto del fondo de la escena como de cada uno de los objetos en movimiento. En esta propuesta existe una clase $c$ para el fondo y una para cada *blob* (conjunto de píxeles pertenecientes a un objeto). La densidad de probabilidad $\mathrm{p}\,(\boldsymbol{p}_t \mid c)$ asociada a un píxel $p$ en el instante $t$ para una clase $c$ se modela como una función de densidad de probabilidad gaussiana y se define de la siguiente manera:

$$\mathrm{p}\left(\boldsymbol{p}_t \mid c\right) = G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{c,t}(p), \boldsymbol{\Sigma}_{c,t}(p)\right) \tag{2.90}$$

$$G\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = (2\pi)^{-D/2} \det\left(\boldsymbol{\Sigma}\right) \exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right) \tag{2.91}$$

$$\boldsymbol{\mu}_{c,t}(p) = E\left[\boldsymbol{p}_t | c\right] \tag{2.92}$$

$$\boldsymbol{\Sigma}_{c,t}(p) = E\left[\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{c,t}(p)\right)\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{c,t}(p)\right)^T | c\right] \tag{2.93}$$

donde:

- $D$ recordemos que es el número de componentes del espacio de color, por ejemplo, $D = 3$ en el caso de RGB.
- $c$ denota la pertenencia del píxel a la clase $c$.
- $\boldsymbol{\mu}_{c,t}(p)$ es el vector de medias de la distribución gaussiana correspondiente a la clase $c$ en instante $t$.
- $\boldsymbol{\Sigma}_{c,t}(p)$ es la matriz de covarianzas de la distribución gaussiana correspondiente a la clase $c$ en instante $t$.

En el método Pfinder la clase a la que pertenece un píxel se determina buscando la $\mathrm{p}\left(\boldsymbol{p}_t \mid c\right)$ mayor. Si denotamos como $\mathcal{M}_c\left(p,t\right)$ a la función de pertenencia de un píxel a la clase $c$, obtenemos la siguiente expresión:

$$\mathcal{M}_c\left(p,t\right) = \begin{cases} 1 & \text{máx}_i\left(\mathrm{p}\left(\boldsymbol{p}_t \mid i\right)\right) = \mathrm{p}\left(\boldsymbol{p}_t \mid c\right) \\ 0 & en\, otro\, caso \end{cases} \tag{2.94}$$

Para la detección del movimiento es habitual simplificar Pfinder empleando únicamente una distribución gaussiana para modelar el fondo y no distinguir entre los distintos objetos, considerándolos genéricamente como primer plano. De este modo, la densidad de probabilidad asociada a un píxel para el fondo de la escena la denotaremos como:

$$\mathrm{p}\left(\boldsymbol{p}_t \mid B\right) = G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{B,t}(p), \boldsymbol{\Sigma}_{B,t}(p)\right) \tag{2.95}$$

Al existir solo una clase, se simplifica la manera de decidir si un píxel pertenece o no al fondo. Y de forma análoga al modelo de fondo basado en la media, la distancia entre la intensidad actual del píxel y la intensidad media del fondo determina si es o no parte del fondo. Sin embargo no se utiliza un umbral fijo, si no que el píxel se

considerará parte del fondo si la distancia es menor que la desviación típica. Por lo que la función de pertenencia al primer plano quedaría como sigue:

$$\mathcal{M}(p,t) = \begin{cases} 1 & \delta^e\left(\boldsymbol{p}_t,\ \boldsymbol{\mu}_{B,t}(p)\right) > T\boldsymbol{\sigma}_{B,t}(p) \\ 0 & en\ otro\ caso \end{cases} \tag{2.96}$$

siendo $T$ es una constante que permite ajustar la cantidad de píxeles clasificados como fondo. Nótese la simplificación adicional que esto supone respecto de calcular la densidad de probabilidad, como se expresa en la Ecuación 2.94.

Para compensar los cambios de iluminación y los movimientos de los objetos, se usa una ecuación de actualización a ciegas que aproxima el valor de la Ecuación 2.92 tal y como se muestra a continuación:

$$\boldsymbol{\mu}_{c,t}(p) = \alpha\boldsymbol{p}_t + (1-\alpha)\boldsymbol{\mu}_{c,t-1}(p) \tag{2.97}$$

Como ya dijimos en Subsección 2.2.1.3, actualizar la media de la intensidad de esta manera no sólo hace que se adapte mejor a los cambios más o menos graduales del fondo, si no que es más eficiente, tanto en términos de espacio como de tiempo, que calcular la media de todas la intensidades pasadas. Además, permite que se corrijan algunos problemas, por ejemplo, los que se producen cuando un objeto estático comienza a moverse. Gracias a esta ecuación de actualización, la intensidad actual del fondo modificará progresivamente la media del modelo de fondo.

De forma similar puede actualizarse la varianza como se sugiere en [123] mediante la siguiente expresión:

$$\boldsymbol{\sigma}^2_{c,t}(p) = \alpha\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{c,t}(p)\right)^2 + (1-\alpha)\boldsymbol{\sigma}^2_{c,t-1}(p) \tag{2.98}$$

#### 2.3.2.2. Modelos basados en distribuciones de mixtura

Idealmente para modelar la escena debería ser suficiente emplear tres distribuciones: una distribución para la intensidad del fondo, otra para el primer plano y en todo caso una más para las sombras. Esta solución ya fue propuesta en [124] para modelar secuencias con tráfico de automóviles. El método Pfinder utiliza varias distribuciones gaussianas para modelar cada elemento de la escena, concretamente una distribución para el fondo y una para cada objeto del primer plano. Sin embargo, respecto de la detección de movimiento, utilizar una única distribución gaussiana para modelar la intensidad de los píxeles del fondo puede ser insuficiente en situaciones en las que este es muy dinámico. Así por ejemplo, el movimiento de un árbol debido al viento hará que los píxeles del fondo tengan en unos fotogramas el color de las

hojas, en otros el de las ramas e incluso el color de lo que haya detrás del árbol. Esos píxeles deben ser considerados como fondo en todos los casos, sin embargo al utilizar una única distribución gaussiana por píxel, el modelo de fondo sería incapaz de identificar estos estados tan dispares como fondo y además detectar los objetos del primer plano con precisión. La media y la desviación típica del modelo de fondo serían corrompidas por los cambios de color bruscos y finalmente no se obtendrían buenos resultados.

Una manera de aliviar los inconvenientes de utilizar una única distribución para el modelo de fondo consiste en utilizar varias distribuciones de probabilidad en cada píxel, lo que se conoce como **distribuciones de mixtura**. En este enfoque, dado un píxel $p$ con intensidad $\boldsymbol{p}_t$, la probabilidad a posteriori de que pertenezca al fondo $P_B(\boldsymbol{p}_t)$ o al primer plano $P_F(\boldsymbol{p}_t)$, viene dada por las siguientes ecuaciones:

$$P_B(\boldsymbol{p}_t) = P\left(B \mid \boldsymbol{p}_t\right) = \frac{\sum_{i=1}^{K^B} \pi_{i,t} \mathrm{p}\left(\boldsymbol{p}_t \mid B, i\right)}{\mathrm{p}\left(\boldsymbol{p}_t\right)} = \frac{\sum_{i=1}^{K^B} \pi_{i,t}\eta_i}{\sum_{i=1}^{K} \pi_{i,t}\eta_i} \tag{2.99}$$

$$P_F(\boldsymbol{p}_t) = P\left(F \mid \boldsymbol{p}_t\right) = \frac{\sum_{i=K^B+1}^{K} \pi_{i,t} \mathrm{p}\left(\boldsymbol{p}_t \mid F, i\right)}{\mathrm{p}\left(\boldsymbol{p}_t\right)} = \frac{\sum_{i=K^{B+1}}^{K} \pi_{i,t}\eta_i}{\sum_{i=1}^{K} \pi_{i,t}\eta_i} \tag{2.100}$$

$$P_B(\boldsymbol{p}_t) + P_F(\boldsymbol{p}_t) = 1 \tag{2.101}$$

donde $K^B$ es el número de distribuciones utilizadas para modelar el fondo, $K_B < K$. $B$ y $F$ denotan fondo (*Background*) y primer plano (*Foreground*), respectivamente. $\eta_i$ es la función de densidad de la i-ésima componente. $\pi_{i,t} \in [0, 1]$ es la probabilidad a priori de la i-ésima componente en el instante $t$. Nótese que en las probabilidades a priori y en las funciones de densidad se han suprimido las referencias a $p$ para simplificar la notación, pero debe tenerse en cuenta que cada píxel tendrá los suyos propios.

La probabilidad a priori pondera la importancia de cada distribución de cara al resultado final y debe ser positiva. La suma de las probabilidades a priori de un mismo píxel debe ser igual a 1:

$$\sum_{i=1}^{K} \pi_{i,t} = 1 \tag{2.102}$$

En principio al utilizar más componentes en el modelo de fondo, se aumenta la capacidad de modelar estados diferentes del fondo. No obstante en función del número de componentes utilizadas, los requisitos de tiempo y memoria aumentan, por lo que según el caso puede ser importante ajustar $K$ a las características de cada secuencia para no utilizar más componentes de las necesarias.

En la detección de movimiento es habitual utilizar distribuciones gaussianas como componentes de las distribuciones de mixtura, MoG (*Mixture of Gaussians*). En

general, en los modelos MoG la densidad de probabilidad $\mathrm{p}(\boldsymbol{p}_t)$ viene dada por la siguiente expresión:

$$\mathrm{p}(\boldsymbol{p}_t) = \sum_{i=1}^{K} \pi_{i,t} G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{i,t}(p), \boldsymbol{\Sigma}_{i,t}(p)\right) \tag{2.103}$$

Es habitual simplificar la matriz de covarianzas asumiendo que cada uno de los canales que componen la imagen son independientes entres sí y tienen la misma varianza. De esta forma se reduce la carga computacional ya que $\boldsymbol{\Sigma}_{i,t}(p)$ tiene la siguiente forma:

$$\boldsymbol{\Sigma}_{i,t} = \sigma_{i,t}^2 \mathbf{I} \tag{2.104}$$

siendo $\mathbf{I}$ la matriz identidad y $\sigma_{i,t}^2$ la varianza de la i-ésima componente gaussiana en el instante $t$. No obstante ya vimos en la Sección 2.1 que la independencia entre canales no es una característica que tengan la mayoría de los espacios de color habituales, sin embargo esta suposición facilita la inversión de la matriz de covarianzas a costa de un empeoramiento en la detección. Como estudiaremos en la Subsección 5.4.3, este empeoramiento puede llegar a ser significativo, por lo que esta simplificación no debe ser siempre la opción elegida.

Los métodos que usan modelos basados en distribuciones de mixtura y en particular en mixturas de gaussianas son los más habituales para la detección de fondo, por lo que a continuación vamos a hacer un análisis más detallado de estos métodos:

- **GrimsonGMM**

  La propuesta de [125] es de referencia al haber sido pioneros en utilizar varias gaussianas para modelar el fondo, nos referiremos a ella como GrimsonGMM. Este método utiliza $K$ distribuciones gaussianas para modelar la intensidad de cada píxel, de modo que la densidad de probabilidad es la que define la Ecuación 2.103.

  La actualización del modelo sigue una aproximación *K-means*. El primer paso consiste en comparar todas las intensidades nuevas, $\boldsymbol{p}_t$, con cada distribución gaussiana hasta que se encuentre un emparejamiento. La intensidad $\boldsymbol{p}_t$ emparejará con la i-ésima distribución, denotado como $M_{i,t}(p) = 1$, si es la componente con mayor probabilidad a priori cuya distancia entre su media y la intensidad del píxel es menor a 2.5 veces la desviación típica de dicha distribución. De esta forma se obtiene la siguiente ecuación de emparejamiento:

$$M_{i,t}(p) = \begin{cases} 1 & \delta_{i,t}^e(p) < 2.5\sigma_{i,t} \wedge \forall j \neq i \left(\left(\delta_{j,t}^e(p) \geq 2.5\sigma_{j,t}\right) \vee \left(\pi_{j,t} < \pi_{i,t}\right)\right) \\ 0 & \textit{en otro caso} \end{cases} \tag{2.105}$$

donde $\delta^e_{i,t}(p)$ es la distancia euclídea a la media de la i-ésima distribución en el instante $t$.

La manera de actualizar el modelo propuesto por GrimsonGMM es la siguiente:

$$\pi_{i,t} = (1-\alpha)\pi_{i,t-1} + \alpha M_{i,t} \tag{2.106}$$

$$\boldsymbol{\mu}_{i,t} = \left((1-\rho)\boldsymbol{\mu}_{i,t-1} + \rho \boldsymbol{p}_t\right) M_{i,t} + (1 - M_{i,t})\, \boldsymbol{\mu}_{i,t-1} \tag{2.107}$$

$$\sigma^2_{i,t} = \left((1-\rho)\sigma^2_{i,t-1} + \rho \left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right)^T \left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right)\right) M_{i,t} + (1 - M_{i,t})\, \sigma^2_{i,t-1} \tag{2.108}$$

$$\rho = \alpha G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{i,t}, \sigma_{i,t}\right) \tag{2.109}$$

donde $0 \leq \alpha \leq 1$ es la tasa de aprendizaje y $\rho$ es el factor de aprendizaje. Las probabilidades a priori son normalizadas después de cada actualización. Como vemos, si no hay emparejamiento, el peso de esa componente decrece, mientras que la media y la varianza permanecen inalteradas. Por otro lado, si se produce emparejamiento, el peso aumenta y tanto la media como la varianza se adaptan al nuevo valor, todo ello es controlado por la tasa y el factor de aprendizaje que regulan la rapidez de la adaptación.

Una alternativa al emparejamiento propuesto originalmente en GrimsonGMM consiste en relajar la condición de emparejamiento, haciendo que emparejen todas las distribuciones en las que la distancia entre la intensidad actual y su media sea 2.5 veces la desviación típica, eliminando la condición de que sólo empareje la componente con mayor probabilidad a priori. Es una forma de hacer más dinámico el modelo, en principio puede ser una ventaja al atenuar los errores en el emparejamiento, sin embargo también puede que las componentes no se diferencien lo suficiente, por lo que se pierda capacidad de generalización.

En caso de que ninguna distribución empareje con la intensidad de entrada, se eliminará aquella cuya probabilidad a priori sea menor y se creará una nueva con una varianza elevada, un peso bajo y una media igual a la intensidad actual del píxel. Esto permite descartar aquellas distribuciones que ya no están entre las que modelan actualmente la intensidad del píxel en favor de una que sí modela la intensidad actual, la cual es posible que vuelva a aparecer en los siguientes fotogramas.

Dentro de las $K$ distribuciones que modelan la intensidad de cada píxel, sólo algunas se consideran parte del fondo. Para determinar qué distribuciones componen el fondo, GrimsonGMM propone establecer un umbral $T$, ordenar de mayor a menor las distribuciones según el valor $\pi_{i,t}/\sigma^2_{i,t}$ y considerar como

parte del modelo de fondo solamente a las $K_B$ distribuciones gaussianas, donde $K_B$ es:

$$K_B = \arg\min_b \left( \sum_{i=1}^{b} \pi_{i,t} > T \right) \tag{2.110}$$

De esta forma, si las distribuciones están ordenadas de mayor a menor según el criterio anterior: $\pi_{i,t}/\sigma_{i,t}^2$, en el instante $t$ el modelo de fondo será representado por la siguiente distribución de mixtura:

$$\sum_{i=1}^{K_B} \pi_{i,t} G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{i,t}(p), \boldsymbol{\Sigma}_{i,t}(p)\right) \tag{2.111}$$

Al seguir este criterio se pretende eligir como características del fondo aquellas distribuciones que hayan sido emparejadas más veces, es decir, que tengan una probabilidad a priori alta, ya que es un indicio de que modelan bien las intensidades que experimenta el fondo de la escena. Por otro lado, ponderar la probabilidad a priori con la varianza sirve para evitar que una distribución sea emparejada constantemente si posee una varianza muy elevada, lo que degeneraría el modelo.

Para determinar si un píxel forma parte del fondo se debe averiguar si se produce algún emparejamiento entre la intensidad actual y las distribuciones que conforman el fondo. Una vez ordenadas las distribuciones según $\pi_{i,t}/\sigma_{i,t}^2$ , quedaría la siguiente expresión:

$$\mathcal{M}(p,t) = \begin{cases} 0 & \exists l \in \{1, ..., K_B\}\,,\ M_{l,t}(p) \\ 1 & en\ otro\ caso \end{cases} \tag{2.112}$$

Emplear un valor de $T$ pequeño puede llevar a que el modelo de fondo sólo sea representado por una única distribución (fondo unimodal), asemejándose al método Pfinder, por lo tanto puede perderse capacidad de generalización. Valores excesivamente grandes de $T$ pueden llevar a la situación opuesta, que el método detecte como fondo objetos que realmente forman parte del primer plano, ya que podría considerar como parte del fondo a demasiadas distribuciones, en particular a alguna distribución que tenga una probabilidad a priori baja.

A modo de ejemplo, si consideramos las siguientes probabilidades a priori $\{0.55,\ 0.25,\ 0.20\}$ , ordenadas según $\pi_{i,t}/\sigma_{i,t}^2$. Si $T = 0.85$, se usarían las tres distribuciones para modelar el fondo, por lo que si se produce algún emparejamiento $M_{i,t}(p) = 1$, se considerará directamente fondo. En el mismo caso, si $T = 0.25$, solo formaría parte del fondo la primera distribución y si no hay emparejamiento con ella, el píxel se marcará como primer plano.

En general se deben elegir valores de $T$ y $K$ que permitan que el fondo sea modelado con más de una distribución (fondo multimodal) para ser un método robusto cuando el fondo de la escena es dinámico. Como veremos, en la práctica suele usarse un umbral $T = 0.75$ y entre 3 y 5 componentes para la mixtura de distribuciones gaussianas. Añadir más componentes al modelo no mejora los resultados y aumenta los requisitos de tiempo y espacio.

- **KaewGMM**

  El trabajo publicado en [126] propone una extensión de las ecuaciones de actualización planteadas en GrimsonGMM para mejorar la velocidad de adaptación del modelo. Cabe destacar que es uno de los métodos estándar para la detección de movimiento de la biblioteca OpenCV [1] (*Open Source Computer Vision*), la cual es ampliamente utilizada en el ámbito de la visión por computador. Este método contempla dos etapas durante el procesado de la secuencia, la primera orientada al entrenamiento del modelo y la segunda a la convergencia del mismo. De este modo las ecuaciones de actualización para la etapa de entrenamiento son:

$$\pi_{i,t} = \pi_{i,t-1} + \frac{1}{t}(M_{i,t} - \pi_{i,t-1}) \tag{2.113}$$

$$\boldsymbol{\mu}_{i,t} = \boldsymbol{\mu}_{i,t-1} + \frac{M_{i,t}}{\sum_{j=1}^{t} M_{i,j}} \left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t-1}\right) \tag{2.114}$$

$$\sigma_{i,t}^2 = \sigma_{i,t-1}^2 + \frac{M_{i,t}}{\sum_{j=1}^{t} M_{i,j}} \left(\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t-1}\right)\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t-1}\right)^T - \sigma_{i,t-1}^2\right) \tag{2.115}$$

  Mientras que para la etapa de convergencia se proponen las siguientes ecuaciones:

$$\pi_{i,t} = \pi_{i,t-1} + \frac{1}{N}(M_{i,t} - \pi_{i,t-1}) \tag{2.116}$$

$$\boldsymbol{\mu}_{i,t} = \boldsymbol{\mu}_{i,t-1} + \frac{1}{N}\left(\frac{M_{i,t}\boldsymbol{p}_t}{\pi_{i,t}} - \boldsymbol{\mu}_{i,t-1}\right) \tag{2.117}$$

$$\sigma_{i,t}^2 = \sigma_{i,t-1}^2 + \frac{1}{N}\left(\frac{M_{i,t}\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t-1}\right)\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t-1}\right)^T}{\pi_{i,t}} - \sigma_{i,t-1}^2\right) \tag{2.118}$$

  donde $N$ es el número de fotogramas dedicados a la etapa de entrenamiento.

[1] http://opencv.org/

Con este enfoque se posibilita que el modelo se adapte rápidamente al principio y más lentamente en la etapa de convergencia. Dado que no utiliza una constante de aprendizaje para todo el procesamiento, requiere que se ajuste adecuadamente la cantidad de fotogramas dedicados al entrenamiento, ya que de esto dependerá la velocidad con la que se irá adaptando el modelo en la etapa de convergencia.

- **ZivkovicGMM**

  En [127, 128] se propone mejorar el método GrimsonGMM utilizando únicamente las distribuciones que realmente estén soportadas por las intensidades actuales de los píxeles. Es decir, si una distribución lleva un periodo de tiempo relativamente grande sin ser emparejada, de modo que su peso es pequeño, se eliminará. Así se pretende optimizar los requisitos de espacio y tiempo al reducir el número de componentes de la distribución de mixtura. Por tanto en este método, que denominaremos ZivkovicGMM, el número de componentes que modelan la intensidad puede ser diferente en cada píxel. Denotaremos como $K_t^p$ la cantidad de componentes de la distribución de mixtura que modela la intensidad del píxel $p$ en el instante $t$, donde $1 \leq K_t^p \leq K$. Al igual que KaewGMM, también es un método para la detección de movimiento estándar en la biblioteca OpenCV.

  Inicialmente se crea una única distribución en cada píxel centrada en la intensidad del primer fotograma. Cada nueva intensidad se compara con el modelo actual de manera similar a Ecuación 2.105 en busca de emparejamientos. Si no se producen emparejamientos, pueden darse dos situaciones:

  - $K_t^p < K$, en este caso se crea una nueva distribución centrada en la intensidad actual, con una alta varianza y una probabilidad a priori baja.
  - $K_t^p = K$, se elimina la distribución con peso más pequeño y se procede igual que en el caso anterior.

  En cualquier caso, la ecuación de actualización de pesos es la siguiente:

  $$\pi_{i,t} = \pi_{i,t-1} + \alpha \left(M_{i,t} - \pi_{i,t-1}\right) - \alpha c_T \tag{2.119}$$

  donde $c_T > 0$ es una constante que permite que las distribuciones con un peso demasiado pequeño sean eliminadas. Para este fin, si al actualizar los pesos existe alguno que es negativo, se elimina la distribución correspondiente.

  El funcionamiento de este método es semejante al de GrimsonGMM tanto para actualizar el resto de variables como para decidir si un píxel forma parte o no del primer plano.

  Por otro lado, en [127, 128] también se estudia la cantidad de fotogramas que se necesitan para que el modelo propuesto en GrimsonGMM considere como parte del fondo a una intensidad nueva, es decir, una que actualmente no

produce ningún emparejamiento. Esto es útil para saber lo que tardará en ser asumido como parte del fondo un objeto que estaba en movimiento y que se ha quedado estático. De forma contraria, ese número también será la cantidad de fotogramas que son necesarios para que un modelo que no había considerado la intensidad real del fondo, comience a modelar correctamente la escena. Este error es común, por ejemplo, cuando se realiza una inicialización deficiente a causa de que hay algún objeto del primer plano durante los primeros fotogramas. Para averiguar este número que denotaremos como $t_C$, podemos emplear la Ecuación 2.110 de donde obtenemos que el peso de la nueva distribución $\pi_{i,t_C}$, deberá satisfacer la siguiente ecuación para asegurar que se considera como fondo:

$$\pi_{i,t_C} > 1 - T \tag{2.120}$$

De esta forma, partiendo del primer fotograma en el que no se produce emparejamiento y siguiendo la Ecuación 2.106 obtenemos las siguientes ecuaciones:

$$\begin{aligned} \pi_{i,1} &= \alpha \\ \pi_{i,2} &= (1-\alpha)\,\alpha + \alpha \\ \pi_{i,3} &= (1-\alpha)\,((1-\alpha)\,\alpha + \alpha) + \alpha = (1-\alpha)^2\,\alpha + (1-\alpha)\,\alpha + \alpha \\ &\dots \\ \pi_{i,n} &= (1-\alpha)^{n-1}\,\alpha + (1-\alpha)^{n-2}\,\alpha + \ldots + \alpha \end{aligned} \tag{2.121}$$

quedando la siguiente serie geométrica:

$$\pi_{i,n} = \sum_{k=0}^{n-1} (1-\alpha)^k\,\alpha \tag{2.122}$$

y resolviendo la ecuación anterior obtenemos el siguiente resultado:

$$\pi_{i,n} = \alpha \frac{1-(1-\alpha)^n}{1-(1-\alpha)} = 1-(1-\alpha)^n \tag{2.123}$$

Por tanto al sustituir la igualdad anterior en la Ecuación 2.120, se tiene que $\pi_{i,t_C}$ debe valer:

$$\pi_{i,t_C} = 1-(1-\alpha)^{t_C} < 1-T \tag{2.124}$$

y tomando logaritmos:

$$t_C > \frac{\log(T)}{\log(1-\alpha)} \tag{2.125}$$

Con este resultado podemos saber cuántos fotogramas serán necesarios aproximadamente para que una intensidad sea tomada como parte del fondo. Si por ejemplo utilizamos un umbral $T = 0.75$ y $\alpha = 0.01$, necesitaremos 29 fotogramas; si usamos el mismo umbral, pero reducimos la tasa de aprendizaje $\alpha = 0.001$, necesitaremos 288 fotogramas. Cabe destacar que si en la escena hay muchos objetos en movimiento o si existe algo de ruido, se necesitarán aún más fotogramas para corregir el modelo.

En cualquier caso, ajustar la cantidad de fotogramas necesarios para tomar una intensidad como fondo puede ser fundamental dependiendo de la secuencia. Si es habitual que los objetos en movimiento se queden estacionarios o al revés, que objetos estacionarios comiencen a moverse, entonces es importante ajustar bien estos parámetros para que el modelo se adapte bien a la dinámica del fondo. También es importante realizar un buen ajuste en los casos en los que la inicialización se realiza con fotogramas que contienen objetos del primer plano, aunque si la secuencia es muy larga, los efectos negativos de una mala inicialización quedan relegados a un segundo plano ya que el modelo suele tener tiempo de adaptarse y los malos resultados solo afectarán a una proporción pequeña de la secuencia.

Al igual que ocurría en los modelos basados en la media, si un objeto relativamente grande y de color uniforme se mueve despacio por la escena, el modelo puede llegar a corromperse, ya que ganarían peso las componentes que modelan el primer plano y finalmente pasarían a considerarse fondo. Para tratar de evitar este inconveniente, en Zivkovic et al. proponen utilizar una tasa de aprendizaje adaptativa basada en $1/t$ de manera que los cambios en el modelo sean progresivamente de menor intensidad conforme pasa el tiempo. Al principio de la secuencia la tasa tendrá un valor elevado, permitiendo ajustar el modelo y solventar los problemas derivados de una mala inicialización, mientras que conforme más avanza el tiempo se necesitarán más fotogramas para realizar un cambio significativo en el modelo, asumiendo que cada vez se ajusta mejor a la realidad. Este enfoque no funcionará bien si el fondo sufre cambios bruscos que no habían ocurrido en la inicialización, lo cual dependerá del tipo de secuencia.

- **Algoritmo EM**

  El algoritmo EM (*Expectation Maximization*) puede ser aplicado a la detección de movimiento. Si bien no proporciona un modelo de fondo, sí que permite aprender los parámetros de un modelo basado en distribuciones de mixtura.

  Dado un dato observado $\boldsymbol{x}_t$, se asume la existencia de unos datos ocultos $\boldsymbol{y}_t$, que conjuntamente forman los datos completos $(\boldsymbol{x}_t, \boldsymbol{y}_t)$. Denotando como $\boldsymbol{\theta}$ el vector de parámetros que debemos aprender, el algoritmo consta de dos pasos que se repiten en cada iteración y que consisten en lo siguiente:

  - Paso E. Calcular la esperanza del logaritmo de la verosimilitud de los datos completos empleando los valores de los parámetros actuales $\boldsymbol{\theta}(t)$:

$$\begin{aligned} Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}\left(t\right)\right) &= E_{\boldsymbol{y}}\left[\log P\left(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}\right) \mid \boldsymbol{x}, \boldsymbol{\theta}\left(t\right)\right] \\ &= \int P\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}\left(t\right)\right) \log P\left(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}\right) d\boldsymbol{y} \end{aligned} \tag{2.126}$$

- Paso M. Maximizar $Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}\left(t\right)\right)$ respecto de $\boldsymbol{\theta}$ para obtener el nuevo vector de parámetros:

$$\boldsymbol{\theta}\left(t+1\right) = \arg\max_{\boldsymbol{\theta}} Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}\left(t\right)\right) \tag{2.127}$$

En el caso del modelado de fondo, $\boldsymbol{x}_t$ puede ser el color actual de un píxel $\boldsymbol{x}_t = \boldsymbol{p}_t$, mientras que $y_t$ revela la componente que ha generado $\boldsymbol{p}_t$. En particular para la mixtura de gaussianas el vector de parámetros sería $\theta = \left(\pi_i, \mu_i, \Sigma_i\right),\ i \in \{1, ..., K\}$ y los pasos quedarían como se muestra a continuación:

- Paso E. Para cada píxel hay que hallar las matrices de responsabilidades de cada componente:

$$\forall i \in \{1, ..., K\},\ R_{i,t} = P\left(i|\boldsymbol{p}_t\right) = \frac{\pi_i \mathrm{p}\left(\boldsymbol{p}_t|i\right)}{\sum_{k=1}^{K} \pi_k \mathrm{p}\left(\boldsymbol{p}_t|k\right)} \tag{2.128}$$

  La responsabilidad $R_{i,t}$ es la probabilidad a posteriori de que un píxel corresponda a la i-ésima componente de la mixtura en el instante $t$.

- Paso M. Actualizar los parámetros de la mixtura tal y como sigue:

$$\pi_{i,t+1} = \frac{1}{t} \sum_{j=1}^{t} R_{i,j} \tag{2.129}$$

$$\mu_{i,t+1} = \frac{\sum_{j=1}^{t} R_{i,j} \boldsymbol{p}_j}{\sum_{j=1}^{t} R_{i,j}} \tag{2.130}$$

$$\boldsymbol{\Sigma}_{i,t+1} = \frac{\sum_{j=1}^{t} R_{i,j} \left(\boldsymbol{p}_j - \mu_{i,t+1}\right) \left(\boldsymbol{p}_j - \mu_{i,t+1}\right)^T}{\sum_{j=1}^{t} R_{i,j}} \tag{2.131}$$

Como puede observarse, al tratarse de un proceso por lotes, los requisitos de tiempo y espacio dependen de lo larga que sea la secuencia. Para obtener la verosimilitud $P\left(\boldsymbol{p}_t|i\right)$ es necesario calcular exponenciales, lo cual es computacionalmente costoso. Por todo ello la aplicación práctica del algoritmo EM es limitada, si bien se usa como referencia.

- **Distribuciones de mixtura no gaussianas**

  A pesar de que la gran mayoría de las publicaciones basadas en distribuciones de mixtura utilizan gaussianas como componentes, existen propuestas que utilizan otras distribuciones para modelar la escena. A continuación expondremos dos casos a título ilustrativo.

En [129] se propone utilizar una mixtura de distribuciones t de Student para modelar la escena tal y como se muestra a continuación:

$$\mathrm{p}(\boldsymbol{p}_t) = \sum_{i=1}^{K} \pi_{i,t} \mathcal{T}\left(\boldsymbol{p}_t, \boldsymbol{\mu}_{i,t}(p), \boldsymbol{\Sigma}_{i,t}(p), v_{i,t}\,(p)\right) \tag{2.132}$$

donde $\mathcal{T}$ es la función de densidad de una distribución t de Student.

En este caso determinar si $\boldsymbol{p}_t$ sigue una distribución t de Student, es equivalente a determinar si sigue una distribución gaussiana de media $\boldsymbol{\mu}_{i,t}(p)$ y covarianza $\boldsymbol{\Sigma}_{i,t}(p)/u_{i,t}\,(p)$, donde $u_{i,t}\,(p)$ sigue una distribución gamma parametrizada con $v_{i,t}\,(p)$.

La dinámica del método, tanto para actualizar los parámetros como para elegir qué componentes forman parte del fondo es similar al caso de la mixtura de gaussianas. De hecho se propone utilizar los mismos criterios y ecuaciones que en el modelo GrimsonGMM. No obstante, para actualizar los parámetros propios de la distribución t de Student se utilizan las siguientes ecuaciones:

$$u_{i,t+1}\,(p) = \frac{v_{i,t}\,(p) + D}{v_{i,t}\,(p) + \delta^m_{i,t}\,(p)} \tag{2.133}$$

$$\xi_{i,t} = \begin{cases} -1 & |\log\left(u_{i,t}\,(p)\right)| > |\log\left(u_{i,t-1}\,(p)\right)| \\ 1 & en\,otro\,caso \end{cases} \tag{2.134}$$

$$v_{i,t+1}\,(p) = v_{i,t}\,(p) + f\rho\xi_{i,t} \tag{2.135}$$

donde $f$ es una constante y $\delta^m_{i,t}\,(p)$ es la distancia de Mahalanobis entre la intensidad actual del píxel y la componente i-ésima de la mixtura.

El método de sustracción de fondo propuesto en [130] está basado el uso de la mixtura Dirichlet. Según el teorema de Bayes, la probabilidad a posteriori de que un píxel pertenezca al fondo es:

$$P\,(B \mid \boldsymbol{p}_t) = \frac{\mathrm{p}\,(\boldsymbol{p}_t \mid B)\,\mathrm{p}\,(B)}{\mathrm{p}\,(\boldsymbol{p}_t \mid B) + \mathrm{p}\,(\boldsymbol{p}_t \mid F)} \tag{2.136}$$

En este método la probabilidad a priori $\mathrm{p}\,(B)$ se considera un valor que es proporcionado como parámetro y $\mathrm{p}\,(\boldsymbol{p}_t \mid F)$ se asume que sigue una distribución uniforme, por lo que en cada fotograma hay que determinar el valor de la función de densidad de probabilidad $\mathrm{p}\,(\boldsymbol{p}_t \mid B)$, la cual que viene dada por la siguiente expresión:

$$\mathrm{p}\,(\boldsymbol{p}_t \mid B) = \sum_{c \epsilon C} \mathrm{p}\,(\boldsymbol{p}_t \mid c, B) \tag{2.137}$$

$$\mathrm{p}\left(\boldsymbol{p}_t \mid c, B\right) = \frac{s_c}{\sum_{i \epsilon C} s_i} \mathcal{P}\left(\boldsymbol{p}_t \mid n_t, k_t, \mu_t, \sigma_t^2\right) \tag{2.138}$$

$$\mathcal{P}\left(\boldsymbol{p}_t \mid n_t, k_t, \mu_t, \sigma_t^2\right) = \prod_{i \epsilon D} \mathcal{T}\left(p_t^i \mid n_{i,t}, \mu_{i,t}, \frac{k_{i,t}+1}{k_{i,t} n_{i,t}} \sigma_{i,t}^2\right) \tag{2.139}$$

donde $p_t^i$ denota la i-ésima componente de color del píxel $p$ en el instante $t$, $C$ es el conjunto de componentes de la mixtura y $s_i$ es una medida de las muestras asignadas a la i-ésima componente de la mixtura que se incrementa en $P\left(B \mid \boldsymbol{p}_t\right)$ con cada muestra añadida a dicha componente. En cada iteración cada muestra se asigna a una componente de la mixtura. Los parámetros de dicha componente se actualizan tal y como sigue:

$$n_{i,t+1} = n_{i,t} + w \tag{2.140}$$

$$k_{i,t+1} = k_{i,t} + w \tag{2.141}$$

$$\mu_{i,t+1} = \frac{k_{i,t}\mu_{i,t} + w p_t^i}{k_{i,t} + w} \tag{2.142}$$

$$\sigma_{i,t+1}^2 = \sigma_{i,t}^2 + \frac{k_{i,t} w}{k_{i,t} + w}\left(p_t^i - \mu_{i,t}\right)^2 \tag{2.143}$$

donde $w = P\left(B \mid \boldsymbol{p}_t\right)$.

#### 2.3.2.3. Modelos basados en máquinas de vectores de soporte

En este tipo de modelos la detección de fondo es tratada como un problema de clasificación. Son enfoques basados en aprendizaje supervisado. Se parte de un conjunto de muestras de entrenamiento $S = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_N, y_N)\}$, donde $\boldsymbol{x}_i$ es un elemento perteneciente a uno de los fotogramas de entrenamiento, como por ejemplo la intensidad de color; $y_i \in \{-1, 1\}$ vale 1 si dicho elemento es primer plano ó $-1$ en caso contrario. Para conocer qué píxeles pertenecen al fondo se utiliza un clasificador óptimo, $f(\boldsymbol{x})$ que es calculado empleando $A$ en base a unas condiciones determinadas.

En la propuesta de [131] se emplea el método de aprendizaje SVM (*Support Vector Machine*) para obtener un clasificador que permita determinar qué muestras pertenecen al fondo y así inicializar el modelo de fondo. $f(\boldsymbol{x})$ es calculado mediante un conjunto de entrenamiento en donde los píxeles se agrupan en bloques de 4x4 y en lugar de utilizar la intensidad del color, se utilizan rasgos. El clasificador empleado en SVM es el siguiente:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x}) \tag{2.144}$$

donde $\phi(\boldsymbol{x})$ es una función que asigna rasgos a los bloques de píxeles. De esta manera el conjunto de entrenamiento tiene la siguiente forma:

$$S = \{(\phi(\boldsymbol{x}_1), y_1), ..., (\phi(\boldsymbol{x}_N), y_N)\} \tag{2.145}$$

En esta propuesta se usan como rasgos característicos de los bloques de entrada tanto el flujo óptico como la diferencia entre fotogramas.

Para calcular $\boldsymbol{w}$ en SVM se debe resolver un problema de optimización como el que se indica a continuación:

$$min_{\boldsymbol{w},\xi_i} \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_i \xi_i \tag{2.146}$$

sujeto a las siguientes condiciones:

$$y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i \tag{2.147}$$

$$\xi_i \geq 0 \tag{2.148}$$

Finalmente, se utiliza la siguiente expresión para definir la función de densidad de probabilidad del fondo asociada a una muestra:

$$\mathrm{p}(\boldsymbol{x} \mid B,\ f(\boldsymbol{x})) = \frac{1}{1 + exp(A_1 f(\boldsymbol{x}) + A_2)} \tag{2.149}$$

donde $A_1$ y $A_2$ son parámetros ajustados empíricamente.

En [132] se usa SVR [133] (*Support Vector Regression*) para determinar si un píxel pertenece o no al fondo. A diferencia de la propuesta anterior, existe un clasificador SVR para cada píxel. Se usan directamente las intensidades de cada píxel para inicializar $f(\boldsymbol{x})$ y se actualiza utilizando un algoritmo de aprendizaje como el descrito en [134] con las nuevas intensidades que surjan a lo largo de la secuencia.

El método de detección de primer plano utilizado consiste en calcular la puntuación $f(\boldsymbol{p}_t)$ para la intensidad actual del píxel y comprobar si supera un cierto umbral:

$$\mathcal{M}(p,t) = \begin{cases} 1 & f(\boldsymbol{p}_t) > T \\ 0 & en\ otro\ caso \end{cases} \tag{2.150}$$

$T$ puede ser ajustado para aumentar el rendimiento en función de la secuencia considerada.

#### 2.3.2.4. Modelos KDE

Una manera de modelar la escena consiste en estimar la intensidad de los píxeles utilizando un método no paramétrico basado en funciones núcleo o KDE (*Kernel denstiy estimation*). De esta manera la función de densidad de probabilidad de cada píxel de la imagen sería la siguiente:

$$\mathrm{p}(\boldsymbol{p}_t|B) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{K}\left(\boldsymbol{p}_t - \boldsymbol{p}_i\right) \tag{2.151}$$

donde $\mathcal{K}$ es una función núcleo, $N$ es el número de muestras usadas para realizar la estimación y el conjunto de estas muestras lo denotamos con $S$.

El modelo propuesto en [135], que denominaremos **ElgammalKDE**, emplea un enfoque basado en KDE donde se utilizan funciones núcleo gaussianas. El conjunto $S$ contiene intensidades recientes seleccionadas dentro de una ventana de tiempo de tamaño $W$, tal que $W \geq N$. Se asume que los canales de color son independientes entre sí, por lo que la función de densidad de probabilidad quedaría como se indica a continuación:

$$\mathrm{p}(\boldsymbol{p}_t|B) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{D}\frac{1}{\sqrt{2\pi\sigma_j^2}}\exp\left(-\frac{1}{2}\frac{\left(p_t^j - p_i^j\right)^2}{\sigma_j^2}\right) \tag{2.152}$$

$$\sigma_j = \frac{m_j}{0.68\sqrt{2}} \tag{2.153}$$

donde $m_j$ es la mediana del canal j-ésimo calculada utilizando todos los pares consecutivos de intensidades $\left|p_i^j - p_{i+1}^j\right|$, tal que $\boldsymbol{p}_i, \boldsymbol{p}_{i+1} \in S$.

Al igual que en algunos modelos ya descritos, en este trabajo se emplea la idea de que un píxel pertenecerá al fondo si $\mathrm{p}(\boldsymbol{p}_t|B) \geq T$, donde $T$ es un umbral fijo que debe ser ajustado antes de iniciar la segmentación. Sin embargo durante la segmentación se mantienen dos modelos: modelo a largo plazo (*Long-term model)* y el modelo a corto plazo (*Short-term model*). Como resultado, se definen dos funciones de densidad de probabilidad asociadas a cada uno de estos modelos: $\mathrm{p}_L(\boldsymbol{p}_t|B)$ y $\mathrm{p}_C(\boldsymbol{p}_t|B)$ para el modelo a largo y corto plazo, respectivamente. De esta manera un píxel pertenecerá al fondo si en ambos modelos se supera el umbral $T$ o bien si se supera solo en el modelo a corto plazo, pero es vecino de un píxel cuyos modelos superan el umbral. La función $\mathcal{M}\left(p,t\right)$ quedaría de la siguiente manera:

$$\mathcal{M}\left(p,t\right) = \begin{cases} 0 & \left(\mathrm{p}_C(\boldsymbol{p}_t|B) \geq T\right) \wedge \exists r,\, vecino\left(p,q\right) \wedge \\ & \left(\mathrm{p}_C(\boldsymbol{q}_t|B) \geq T\right) \wedge \left(\mathrm{p}_L(\boldsymbol{q}_t|B) \geq T\right) \\ 1 & en\, otro\, caso \end{cases} \tag{2.154}$$

donde la función *vecino* determina si un píxel es adyacente a otro en la imagen, es decir, si es uno de los ocho píxeles que le rodea o él mismo:

$$vecino\,(p,q) = \begin{cases} verdadero & p = q \\ verdadero & p\,y\,q\,son\,adyacentes \\ falso & en\,otro\,caso \end{cases} \tag{2.155}$$

Al existir dos modelos, se definen dos conjuntos de muestras para cada uno de ellos: $S_L$ y $S_C$. Para actualizar los conjuntos de muestras se deben mantener en cada uno las últimas $N$ muestras que estén dentro de la ventana de tiempo y que cumplan ciertos criterios. Para el modelo a largo plazo se eligen las últimas $N$ muestras sin ninguna consideración adicional. En el modelo a corto plazo solo se incluyen las últimas muestras que hayan sido marcadas como fondo.

Este método propone no utilizar directamente las componentes RGB con el fin de disminuir los errores producidos por las sombras. Para ello convierte la imagen a un espacio de tres componentes $[r, g, s]$. De esta manera $\boldsymbol{p}_t = [r_t, g_t, s_t]$, donde la componente $s$ pretende proporcionar una medida de la iluminación. Para convertir de RGB a este espacio de color deben usarse las siguientes ecuaciones:

$$r = \frac{R}{R+G+B} \tag{2.156}$$

$$g = \frac{G}{R+G+B} \tag{2.157}$$

$$s = R+G+B \tag{2.158}$$

Utilizando este espacio de color se puede saber si el color de un píxel es una versión iluminada u oscurecida de otro. Para comparar el color de un píxel $i$ con el de $j$, se puede utilizar como medida: $\frac{s_i}{s_j}$. De esta forma si el color de $i$ es igual que el de $j$ pero oscurecido, entonces $\beta_1 \leq \frac{s_i}{s_j} \leq 1$, donde $\beta_1$ es un umbral definido arbitrariamente a partir del cual ya no se considera que el color de $i$ pueda ser debido a una sombra que afecta al color de $j$. De forma contraria, si el color de $i$ es una versión iluminada de $j$, podemos concluir que $\beta_2 \geq \frac{s_i}{s_j} \geq 0$.

Para tratar de que el modelo sea inmune a los cambios de iluminación se propone calcular las probabilidades $\mathrm{p}_L(\boldsymbol{p}_t|B)$ y $\mathrm{p}_C(\boldsymbol{p}_t|B)$ mediante la Ecuación 2.152 considerando únicamente las componentes $[r, g]$ de aquellos píxeles relevantes, es decir, aquellos que si son iluminados u oscurecidos, podrían tener el mismo nivel de iluminación. Por lo que los elementos que son considerados para calcular $P(\boldsymbol{p}_t|B)$, denotado como $S_R \subseteq S$, vienen dados por la siguiente ecuación:

$$S_R = \left\{ \boldsymbol{p}_i \mid \boldsymbol{p}_i \in S,\, \beta_1 \leq \frac{s_t}{s_i} \leq \beta_2 \right\} \tag{2.159}$$

donde $\beta_1$ y $\beta_2$ son constantes ajustadas antes de la ejecución del método para reducir los errores que pueden producirse por cambios de iluminación, como en el caso de las sombras.

#### 2.3.2.5. Modelos PCA

El análisis de componentes principales o PCA (*Principal Component Analysis*) permite reducir la dimensionalidad de un conjunto de datos $\mathcal{S} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ donde la dimensión de $\boldsymbol{x}_i$ es $\mathcal{D}$, buscando y extrayendo aquellas $\mathcal{P}$ componentes que presentan mayor variabilidad. Asumiendo que a menor variabilidad menor relevancia, es posible eliminar las $\mathcal{D}-\mathcal{P}$ componentes que menos influyen a la hora de caracterizar los datos, considerándolas como ruido.

Si calculamos la matriz de covarianzas $\boldsymbol{\Sigma}$ del conjunto de datos $\mathcal{S}$, podemos descomponerla como se indica a continuación:

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \tag{2.160}$$

siendo $\boldsymbol{U}$ una matriz ortogonal formada por los autovectores y $\boldsymbol{\Lambda}$ una matriz diagonal con $\mathcal{D}$ autovalores, uno por autovector. Estas matrices se pueden utilizar para transformar los datos del espacio original a la base ortonormal definida por las componentes principales:

$$\boldsymbol{z} = \boldsymbol{U}^T (\boldsymbol{x} - \boldsymbol{\mu}) \tag{2.161}$$

donde $\boldsymbol{\mu}$ es el vector de medias del conjunto de datos.

Al usar PCA se determinan $\boldsymbol{U}_\mathcal{P}$ y $\boldsymbol{U}_{\mathcal{D}-\mathcal{P}}$ que son, respectivamente, las matrices compuestas por los $\mathcal{P}$ autovectores principales y los $\mathcal{D} - \mathcal{P}$ autovectores considerados como ruido. De esta forma se puede descomponer $\boldsymbol{z}$ obteniendo la siguiente expresión:

$$\boldsymbol{x} - \boldsymbol{\mu} = \boldsymbol{U}_\mathcal{P}\boldsymbol{z}_\mathcal{P} + \boldsymbol{U}_{\mathcal{D}-\mathcal{P}}\boldsymbol{z}_{\mathcal{D}-\mathcal{P}} \tag{2.162}$$

Por lo que si asumimos el error de proyección $\boldsymbol{U}_{\mathcal{D}-\mathcal{P}}\boldsymbol{z}_{\mathcal{D}-\mathcal{P}}$, podemos aproximar $\boldsymbol{x}$ y $\boldsymbol{z}_\mathcal{P}$ utilizando únicamente las $\mathcal{P}$ dimensiones principales mediante las siguientes ecuaciones:

$$\boldsymbol{x} \approx \boldsymbol{\mu} + \boldsymbol{U}_\mathcal{P}\boldsymbol{z}_\mathcal{P} \tag{2.163}$$

$$\boldsymbol{z}_\mathcal{P} \approx \boldsymbol{U}_\mathcal{P} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{2.164}$$

En [99] se propone utilizar PCA para segmentar la escena. Se toman $N$ imágenes de entrenamiento para formar el conjunto de datos $\boldsymbol{\mathcal{S}}$ y calcular tanto $\boldsymbol{U}$ como $\boldsymbol{\mu}$. La dimensión $\mathcal{D}$ de cada muestra es igual al número de píxeles de cada imagen multiplicado por el número de canales de color. Y el número de componentes principales $\mathcal{P}$, es asignado al principio de la simulación. A este propuesta la denominaremos **PCAOliver**.

Para realizar la segmentación en primer lugar se debe convertir la imagen de entrada $\boldsymbol{I}_t$, utilizando la Ecuación 2.164:

$$\boldsymbol{z}_t \approx \boldsymbol{U}_M \left(\boldsymbol{I}_t - \boldsymbol{\mu}\right) \tag{2.165}$$

A continuación se marcarán como primer plano aquellos píxeles cuya distancia euclídea entre la imagen de entrada y su proyección sea mayor que un cierto umbral $T$:

$$\delta^e \left(\boldsymbol{\mathcal{I}}_t, \boldsymbol{z}_t\right) > T \tag{2.166}$$

La idea intuitiva que se sigue es que el nuevo espacio debe recoger las características más relevantes del fondo de la escena, así que si los píxeles de la imagen de entrada original presentan una diferencia significativa con sus proyecciones, significa que pertenecen al primer plano.

#### 2.3.2.6. Modelos basados en redes neuronales

Las redes neuronales artificiales se inspiran en las neuronas biológicas. De este modo una red neuronal artificial está compuesta por un conjunto de neuronas en las que cada una tiene un número determinado de entradas y una salida conectada o no con otra neurona. En general la topología de la red y el tipo y regla de aprendizaje son las características básicas de una red neuronal. Existen diferentes propuestas basadas en redes neuronales para realizar la detección de fondo. Entre otras en [136] se utiliza una red neuronal competitiva basada en dipolos y [137] utiliza una red neuronal para corregir la salida de un algoritmo basado en MoG.

Dentro del campo de las redes neuronales, los mapas autoorganizados (SOM) también son empleados para la detección de fondo. Al igual que el resto de redes neuronales tienen una inspiración biológica, en este caso en el funcionamiento de la corteza visual del cerebro. La topología de esta red consiste en una malla de neuronas. Las neuronas cercanas responderán ante estímulos similares. La dinámica de la red hace que para una entrada dada $\boldsymbol{x}_t$, exista una única neurona ganadora.

La neurona i-ésima de un mapa autoorganizado almacena un prototipo $\boldsymbol{c}_i$ con $D$ componentes. El estado de las neuronas que forman la red en el instante $t$ se puede denotar como $\boldsymbol{A}_t = \{\boldsymbol{c}_{1,t}, ..., \boldsymbol{c}_{n,t}\}$. Para una entrada $\boldsymbol{x}_t$ la siguiente función determina si la neurona i-ésima es la ganadora:

$$ganadora\left(i,\boldsymbol{x}_t\right)=\begin{cases}1 & i=\arg\min_j\left\|\boldsymbol{x}_t-\boldsymbol{c}_{j,t}\right\| \\ 0 & en\,otro\,caso\end{cases} \tag{2.167}$$

La regla de aprendizaje que mostramos a continuación hace que el prototipo de la neurona ganadora se aproxime a la entrada actual y, en menor medida, sus vecinas hacen lo propio:

$$\boldsymbol{c}_{j,t+1}=\left(1-\alpha\left(t\right)\varphi\left(i,j\right)\right)\boldsymbol{c}_{j,t}+\alpha\left(t\right)\varphi\left(i,j\right)\boldsymbol{x}_t \tag{2.168}$$

siendo $i$ el índice de la neurona ganadora y $j$ el de una neurona cualquiera. $\alpha\left(t\right)$ es la tasa de aprendizaje y debe tener una caída lineal o exponencial. Así la red se adapta rápidamente al principio y va convergiendo conforme avanza el tiempo. $\varphi\left(i,j\right)$ es la función de cercanía y es decreciente con la distancia entre dos neuronas, es decir, a mayor distancia, menor valor. Suele usarse un núcleo gaussiano con un parámetro $\rho>0$:

$$\varphi\left(i,j\right)=\exp\left(\frac{-\delta\left(i,j\right)}{\rho}\right) \tag{2.169}$$

En [138] se propone un método denominado **SOBS** (*Self-Organizing Background Subtraction*) que utiliza una mapa autoorganizado para realizar la extracción del fondo. El mapa consta de $3\times fil$ filas y $3\times col$ columnas, existiendo un conjunto $C_t(p)=\{\boldsymbol{c}_{1,t}(p),...,\boldsymbol{c}_{9,t}(p)\}$ de neuronas asociadas a cada píxel $p$. Cada neurona inicializa su prototipo con el color del primer fotograma de entrada. El espacio de color empleado es HSV. La dinámica es muy similar a la que se acaba de describir para los mapas autoorganizados en general.

Para saber si un color $\boldsymbol{p}_t$ pertenece al fondo se debe determinar si existe alguna neurona que empareje con el color de entrada. Un emparejamiento se produce cuando la distancia entre el color y la neurona ganadora es menor que un cierto umbral:

$$M_{i,t}(p)=\begin{cases}1 & ganadora\left(i,\boldsymbol{p}_t\right)\wedge\delta^e\left(\boldsymbol{p}_t,\boldsymbol{c}_{i,t}(p)\right)\leq\epsilon\left(t\right) \\ 0 & en\,otro\,caso\end{cases} \tag{2.170}$$

$$\mathcal{M}\left(p,t\right)=\begin{cases}0 & \exists\boldsymbol{c}_{i,t}(p)\in C_t(p),\ M_{i,t}(p)=1 \\ 1 & en\,otro\,caso\end{cases} \tag{2.171}$$

$\epsilon\left(t\right)$ es un umbral cuyo valor depende de si estamos en la etapa de ordenación o en la de convergencia:

$$\epsilon\left(t\right)=\begin{cases}\epsilon_1, & t\leq N \\ \epsilon_2, & t>N\end{cases} \tag{2.172}$$

donde $\epsilon_1 \geq \epsilon_2$ y $N$ es el número de fotogramas dedicados a la etapa de ordenación.

Para actualizar los prototipos de cada neurona se utiliza la Ecuación 2.168. El valor de la tasa de aprendizaje viene dado por la siguiente expresión:

$$\alpha(t) = \begin{cases} \alpha_1 - t\frac{\alpha_1 - \alpha_2}{N}, & t \leq N \\ \alpha_2, & t > N \end{cases} \tag{2.173}$$

que como podemos observar también depende de la etapa en la que se encuentre el método. $\alpha_1$ y $\alpha_2$ son constantes preestablecidas que satisfacen $\alpha_1 \geq \alpha_2$.

En [139] también se utilizan mapas autoorganizados para realizar la segmentación del fondo, pero a diferencia de SOBS permite obtener la probabilidad de que un píxel pertenezca al fondo y también proporciona un modelo de los objetos del primer plano. En este caso se usa una red autoorganizada probabilística en la que:

$$\mathrm{p}(\boldsymbol{p}_t \mid B) = \frac{1}{H}\sum_{i=1}^{H} \mathrm{p}(\boldsymbol{p}_t \mid i) \tag{2.174}$$

donde $H$ es el número de componentes (neuronas) en la mixtura del mapa autoorganizado y las probabilidades a priori son iguales.

#### 2.3.2.7. Modelos basados en lógica difusa

Los conceptos de lógica difusa pueden ser aplicados al campo de la detección de fondo utilizando diferentes enfoques. Se puede utilizar prácticamente en cualquier etapa del procesado. Existen propuestas orientadas a: realizar el modelado del fondo [140, 141], mantener el modelo [142], detectar el primer plano [26] y también en el post-procesado [143].

En [26] se propone extraer y agregar rasgos del modelo de fondo y de la imagen actual con el fin de buscar diferencias que indiquen la existencia de objetos del primer plano. La agregación de estas características se realiza utilizando lógica difusa, en concreto la integral de Choquet. A este método lo denominaremos **FuzzyElBaf**.

El modelo inicial de fondo se realiza promediando el color de cada píxel durante los $N$ primeros fotogramas. Se consideran dos tipos de rasgos a extraer: basados en el color y basados en texturas. Para obtener los primeros se convierte de RGB a Ohta, HSV o a YCrCb según el caso. Para los rasgos basados en texturas se utiliza el operador LBP (*Local Binary Pattern*) propuesto en [144]. Originalmente este operador está definido para los vecinos $3 \times 3$ de un píxel, sin embargo puede ser generalizado para $d \times d$ píxeles:

$$L(\boldsymbol{p}_t) = \sum_{i=0}^{d-1} s(g_i(t) - g_p(t))\, 2^i \tag{2.175}$$

$$s(y) = \begin{cases} 1 & y \geq 0 \\ 0 & y < 0 \end{cases} \tag{2.176}$$

donde $g_i(t)$ corresponde al valor en escala de grises del i-ésimo píxel vecino de $p$ y $g_p(t)$ es el valor en escala de grises de $\boldsymbol{p}_t$ en el instante $t$.

Los operadores $S^T$ y $S_k^C$ se definen para medir el grado de similitud entre la imagen de entrada y el modelo de fondo atendiendo a las texturas y al espacio de color, respectivamente. $S_k^C$ se define para cada uno de los espacio de color usados, donde $k = \{1, 2, 3\}$ denota cada uno de los canales de color. La definición de estos dos operadores es la siguiente:

$$S_k^C(p) = \begin{cases} \frac{I_k(p)}{B_k(p)} & if\ I_k(p) < B_k(p) \\ 1 & if\ I_k(p) = B_k(p) \\ \frac{B_k(p)}{I_k(p)} & if\ I_k(p) > B_k(p) \end{cases} \tag{2.177}$$

$$S^T(p) = \begin{cases} \frac{L^I(p)}{L^B(p)} & if\ L^I(p) < L^B(p) \\ 1 & if\ L^I(p) = L^B(p) \\ \frac{L^B(p)}{L^I(p)} & if\ L^I(p) > L^B(p) \end{cases} \tag{2.178}$$

donde $I_k$ y $B_k$ son, respectivamente, las intensidades de la componente de color k-ésima del píxel de entrada y de su modelo de fondo. $L^I(p)$ y $L^B(p)$ son, respectivamente, el valor del operador LPB del píxel de entrada y de su modelo de fondo.

Una vez se tienen las medidas de similitud entre el píxel de entrada y el modelo de fondo, se agregan utilizando la integral de Choquet. Para definir la integral de Choquet podemos considerar $n$ rasgos para cada píxel. Denotamos como $(x_1, ..., x_n)$ al vector de rasgos y $(h(x_1), ..., h(x_n))$ a los valores del vector de rasgos. De este modo la integral de Choquet puede ser calculada mediante la siguiente expresión:

$$C_\mu(x_1, ..., x_n) = \sum_{i=1}^{n} h\left(x_{(i)}\right) \mu\left(A_{(i)}\right) - \mu\left(A_{(i+1)}\right) \tag{2.179}$$

donde $\left(h(x_{(1)}), ..., h(x_{(n)})\right)$ es una permutación no decreciente de los valores del vector de rasgos, $A_{(i)} = \left\{x_{(i)}, ..., x_{(n)}\right\}$ y $\mu$ satisface las siguientes condiciones:

$$\begin{gathered} \mu(\emptyset) = 0 \\ \mu\left(\left\{x_{(1)}, ..., x_{(n)}\right\}\right) = 1 \\ A \subseteq B \Rightarrow \mu(A) \leq \mu(B) \end{gathered} \tag{2.180}$$

En este método se utilizan 3 rasgos ($n = 3$), dos para color y uno para el operador LBP. Por lo tanto $h(x_i) = S_k^C(p)$ en el caso del color y $h(x_i) = S^T(p)$ para las texturas.

Para determinar qué píxeles pertenecen al primer plano se compara el resultado de la integral con un cierto umbral $T$, tal y como se muestra a continuación:

$$\mathcal{M}(p,t) = \begin{cases} 1 & C_\mu(x_1, x_2, x_3) < T \\ 0 & en\, otro\, caso \end{cases} \tag{2.181}$$

donde el valor de la integral de Choquet se refiere a los rasgos del píxel $p$ en el instante $t$.

Para actualizar el modelo de fondo se utiliza un esquema selectivo con ecuación como la siguiente:

$$\begin{gathered} \boldsymbol{B}_{t+1}(p) = \mathcal{M}(p,t)\left((1-\beta)\,\boldsymbol{B}_i(p) + \beta \boldsymbol{p}_i\right) + \\ (1 - \mathcal{M}(p,t))\left((1-\alpha)\,\boldsymbol{B}_i(p) + \alpha \boldsymbol{p}_i\right) \end{gathered} \tag{2.182}$$

donde $\alpha$ y $\beta$ son constantes previamente establecidas.

En [142] se propone aprovechar el resultado de la integral de Choquet para actualizar el modelo de fondo. En lugar de utilizar un esquema selectivo como el anterior, se emplea uno difuso como el de la siguiente ecuación:

$$\boldsymbol{B}_{t+1}(p) = \beta_t \boldsymbol{B}_i(p) + (1-\beta_t)\left((1-\alpha)\,\boldsymbol{B}_i(p) + \alpha \boldsymbol{p}_t\right) \tag{2.183}$$

donde $\beta_t$ es función de $C_\mu$ tal que $\beta_t = 1$ para máx$(C_\mu)$ y $\beta_t = 0$ para mín$(C_\mu)$. De esta manera el modelo se adaptará al color actual del píxel en función del valor de $C_\mu$. Conforme más se parezca al fondo, más rápido asumirá el color de $\boldsymbol{p}_t$. El parámetro $\alpha$ limita lo bruscos que pueden llegar a ser estos cambios.

Por otro lado, en [145] se expone un modelo que combina mapas autoorganizados y lógica difusa. En adelante nos referiremos a él como **FASOM** (*Fuzzy Adaptative Self-Organizing Background Subtraction*). El modelo planteado es similar al propuesto en [138], consiste en un mapa autoorganizado de $3 \times fil$ filas y $3 \times col$ columnas, la diferencia estriba en que se ha aplicado enfoque difuso para calcular la tasa de aprendizaje, $\alpha(t)$. En lugar de usar una tasa de aprendizaje cuyo valor depende de la etapa en la que se encuentre algoritmo, se usa una cuyo valor depende de la neurona que se esté considerando y además usa un enfoque difuso en su definición:

$$\alpha_{i,j}(t) = F_1(\boldsymbol{p}_t)\, F_2(\boldsymbol{p}_t)\, \alpha(t) \tag{2.184}$$

donde el valor de $\alpha(t)$ se determina mediante la Ecuación 2.173, mientras que $F_1(\boldsymbol{p}_t)$ y $F_2(\boldsymbol{p}_t)$ son funciones basadas en lógica difusa cuyas definiciones son:

$$F_1(\boldsymbol{p}_t) = \begin{cases} 1 - \frac{\delta^e(\boldsymbol{p}_t, \boldsymbol{c}_{M,t})}{\epsilon} & \delta^e(\boldsymbol{p}_t, \boldsymbol{c}_{M,t}) < \epsilon \\ 0 & en\, otro\, caso \end{cases} \tag{2.185}$$

$$F_2(\boldsymbol{p}_t) = \begin{cases} 2NCF(\boldsymbol{p}_t) - 1 & NCF(\boldsymbol{p}_t) > 0.5 \\ 0 & en\,otro\,caso \end{cases} \tag{2.186}$$

donde $\boldsymbol{c}_{M,t}(p_t)$ es el peso de la neurona ganadora $M$ en la posición $p$, mientras que $NCF(\boldsymbol{p}_t)$ es un factor de coherencia entre vecinos dado por la siguiente expresión:

$$NCF(\boldsymbol{p}) = \frac{card(\Omega_p)}{card(vecinos(p))} \tag{2.187}$$

$$\Omega_p = \{q \in vecinos(p) \mid \|\boldsymbol{q}_t - \boldsymbol{c}_{M,t}(q_t)\| < \epsilon\} \tag{2.188}$$

siendo $vecinos(p)$ una función que devuelve el conjunto de píxeles vecinos de un píxel $p$ y $card$ denota la cardinalidad de un conjunto. Por lo tanto $NCF(\boldsymbol{p})$ proporciona una medida de la cantidad de píxeles en las inmediaciones de $p$ que están bien representados con el modelo de fondo actual.

Así pues, $F_1(\boldsymbol{p}_t)$ es una función que mide el grado de pertenencia al fondo del color de entrada actual, mientras que $F_2(\boldsymbol{p}_t)$ proporciona una medida de lo bien que se ajusta el modelo de fondo a los píxeles de entrada en la vecindad del píxel considerado.

### 2.3.3. Post-procesado

Con el fin de subsanar algunas de las limitaciones que tienen los métodos de detección de movimiento se puede emplear una etapa de post-procesado. Durante esta etapa se toma directamente la máscara del primer plano generada y se analiza en busca de píxeles que no hayan sido marcados correctamente. En general el post-procesado mejora los resultados si los parámetros son ajustados convenientemente [146]. A continuación mostramos algunas de las técnicas más habituales:

- **Operadores morfológicos**

  La erosión y dilatación son operadores morfológicos que se aplican a cada píxel de la máscara de salida. Estos operadores utilizan una máscara binaria denominada elemento estructurante. El tamaño y la forma de este elemento estructurante determinará qué píxeles vecinos son considerados en la operación. A continuación se muestran dos ejemplos de elemento estructurante:

$$\mathbf{E}_{cuadrado} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{2.189}$$

$$\mathbf{E}_{diamante} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{2.190}$$

donde los valores que son iguales a 1 serán los vecinos tenidos en cuenta para realizar la operación morfológica.

En la **erosión** se asigna al fondo el píxel en cuestión si alguno de los vecinos que están marcados en el elemento estructurante pertenece al fondo. De forma contraria, el operador **dilatación** asigna al primer plano el píxel considerado si alguno de los vecinos que están marcados en el elemento estructurante pertenece al primer plano.

Lo habitual es aplicar estos dos operadores uno después de otro para no degenerar la segmentación realizada. Realizar una dilatación y después una erosión permite rellenar los huecos que quedan en el interior de las regiones del primer plano. Al realizar la erosión y después la dilatación se busca eliminar los píxeles del primer plano aislados.

- **Filtros**

  De manera similar a los operadores morfológicos se pueden utilizar filtros, por ejemplo, la mediana o la moda que aplican dichas medidas estadísticas a los vecinos de cada píxel. Además de ser útiles para el post-procesado también pueden ser utilizados para mejorar la entrada del método de detección, es decir, en una etapa de pre-procesado. En este sentido también pueden ser útiles otros filtros como el gaussiano.

- **Procesamiento de blobs**

  En un análisis a mayor nivel se puede considerar el tamaño que deben tener los objetos de la escena para descartar aquellos que son de tamaño demasiado pequeño. En algunos trabajos se considera la información de las etapas posteriores a la detección de movimiento para mejorar la segmentación [32].

- **Test de relevancia**

  Algunos métodos proporcionan una medida de lo probable que es que un píxel pertenezca al primer plano. Estudiando estas probabilidades en todos los píxeles de la escena se puede determinar si una región que ha sido marcada como primer plano tiene una probabilidad demasiado pequeña en comparación con el resto de regiones.

- **Flujo óptico**

  En cada píxel se puede definir una medida del flujo óptico de manera que si no hay movimiento, el flujo óptico sea prácticamente nulo. Gracias a esto se puede eliminar el rastro (regiones fantasma) que dejan los objetos que hasta el momento habían pertenecido al fondo y han comenzado a moverse. Sin embargo, este enfoque puede ser perjudicial en el caso de que un objeto del primer plano se detenga ya que también sería eliminado del primer plano.

## 2.4. Evaluación del rendimiento

La evaluación objetiva de los resultados es una parte fundamental de cualquier proceso científico. Para llevar a cabo esta evaluación deben emplearse unas medidas que permitan conocer el grado de consecución de los objetos planteados. Además estas medidas deben facilitar la comparación de los resultados propios con los de otras propuestas, por tanto deben usarse aquellas que sean empleadas habitualmente en el ámbito en el que se esté trabajando.

Como hemos visto la videovigilancia es un proceso que se compone de distintas etapas, cada una de ellas con diferentes objetivos, por lo tanto en cada etapa se deben usar unas medidas de rendimiento distintas. La evaluación del rendimiento en la videovigilancia se lleva a cabo ejecutando los métodos propuestos en un conjunto de secuencias de prueba. En función de si la etapa es de mayor o menor nivel se miden unas características u otras. Así por ejemplo, para la etapa de reconocimiento de acciones se miden el número de acciones reconocidas de entre las que efectivamente hay en cada secuencia estudiada. En el caso del seguimiento de objetos se debe evaluar si el método propuesto es capaz de seguir a cada uno de los objetos durante toda la secuencia o al menos en qué porcentaje de fotogramas lo consigue. Para el reconocimiento de objetos se analiza si la propuesta es capaz de identificar los objetos que hay en la escena, sin que sea necesario que los siga a lo largo de la secuencia. Por último, en la detección de movimiento se analizan los resultados utilizando un nivel más bajo que los anteriores, se comprueba si se reconoce el movimiento debido a objetos del primer plano en cada uno de los píxeles que componen la escena.

Las secuencias utilizadas para evaluar el rendimiento de los métodos propuestos pueden condicionar la semejanza entre el rendimiento experimental y el real. Por ello es importante emplear un conjunto de secuencias suficientemente amplio de manera que represente la mayor variedad posible de entornos para los que está pensado el método propuesto.

Se pueden diferenciar dos tipos de secuencias en base a su origen: reales y sintéticas. Las secuencias reales son aquellas que se componen de fotogramas capturados directamente de la realidad. Las secuencias sintéticas difieren de las anteriores en que se les añaden objetos en movimiento utilizando técnicas digitales. Estás últimas tienen la ventaja de que es más sencillo y exacto determinar donde están los píxeles que registran movimiento. Esto permite una evaluación más exacta del rendimiento. Por contra, los objetos del primer plano no interaccionan con la escena, como suele ocurrir en la realidad. Por ejemplo, los objetos en movimiento no proyectan sombras ni se ven afectados por la iluminación. Esto hace que el rendimiento medido pueda no corresponderse con el que tendría en situaciones reales.

Para facilitar la reproducibilidad de los resultados obtenidos es deseable que las secuencias estén disponibles públicamente. Por todo ello en esta tesis se han empleado únicamente secuencias que ya se encuentran en Internet y que son públicamente accesibles. Además en la práctica totalidad de los casos se han utilizado secuencias que

forman parte de repositorios conocidos en el campo de la detección de movimiento. De este modo se pretende facilitar la comparación de los resultados con otras propuestas que también usen estos repositorios.

Dado que esta tesis versa sobre la detección de objetos en movimiento, a continuación expondremos los conceptos básicos (Subsección 2.4.1) necesarios para definir las medidas de rendimiento habituales en esta etapa de la videovigilancia y que serán utilizadas posteriormente en la Subsección 2.4.2 para definir medidas más sofisticadas.

### 2.4.1. Conceptos básicos

Para evaluar el rendimiento en términos de detección de movimiento es necesario comparar en cada secuencia la salida real del método de detección con una salida ideal. Esta salida ideal se compone de una serie de imágenes binarias que se corresponden con fotogramas de la secuencia de entrada. A estas imágenes se les denomina también ***ground truth***. Al igual que en la salida de un método de detección, el valor 1 en un píxel denota movimiento y el 0 la ausencia del mismo. En ocasiones también se utilizan valores intermedios para representar los píxeles en los que hay sombra.

Si bien en la mayoría de las secuencias no están disponibles los *ground truth* de todos los fotogramas, es interesante considerar al menos aquellos en los que se producen situaciones especialmente interesantes de cara a la detección de movimiento. Al emplear de forma mayoritaria secuencias que están incluidas en repositorios públicos para la detección de movimiento, los *ground truth* están ya disponibles. No obstante, en ocasiones esto no ocurre y es necesario realizar una segmentación manual de algunos fotogramas de entrada para utilizarlos de *ground truth*. En estos casos se ha prestado especial atención a los fotogramas elegidos para que los resultados obtenidos no estén sesgados a favor o en contra de los métodos propuestos. Se ha optado por utilizar una frecuencia de muestreo constante en cada secuencia que determine qué fotogramas tendrán *ground truth* asociado. Las frecuencias elegidas han sido aquellas que hacen que al muestrear se obtenga una cantidad similar de *ground truth* en todas las secuencias analizadas. Cuando ha sido necesario, se ha comenzado a muestrear en un fotograma en concreto para que los *ground truth* resultantes incidan en las características más interesantes de la secuencia.

Sea $\mathcal{A}$ el conjunto de píxeles que están marcados como primer plano en el *ground truth* y $\mathcal{B}$ el conjunto de píxeles marcados como primer plano por el método a evaluar:

$$\mathcal{A} = \left\{ p \mid \widehat{\mathcal{M}}\left(\boldsymbol{p}_t\right) = 1 \right\} \tag{2.191}$$

$$\mathcal{B} = \left\{ p \mid \mathcal{M}\left(\boldsymbol{p}_t\right) = 1 \right\} \tag{2.192}$$

Adicionalmente y según el caso se puede considerar un conjunto $Sh$ con los píxeles que están marcados como sombra en el *ground truth:*

$$Sh = \left\{ p \mid 0 < \widehat{\mathcal{M}}\left(\boldsymbol{p}_t\right) < 1 \right\} \tag{2.193}$$

donde $Sh = \emptyset$ si no se consideran las sombras en la secuencia analizada.

A la hora de evaluar la segmentación de un método pueden suceder cuatro situaciones distintas en cada píxel atendiendo al valor del mismo en el *ground truth* y en la salida generada por el método de detección de movimiento (ver Cuadro 2.2):

- Verdadero positivo (tp, *true positive*): El píxel pertenece al primer plano en el *ground truth* y se detecta como tal.
- Verdadero negativo (tn, *true negative*): Pertenece al fondo y se detecta como tal.
- Falso negativo (fn, *false negative*): Pertenece al primer plano y se detecta como fondo.
- Falso positivo (fp, *false negative*): Pertenece al fondo y se detecta como primer plano.

| | | $\mathcal{M}(\boldsymbol{p}_t)$ | |
|---|---|---|---|
| | | 0 | 1 |
| $\widehat{\mathcal{M}}(\boldsymbol{p}_t)$ | 0 | tn | fp |
| | 1 | fn | tp |

**Cuadro 2.2:** Matriz de confusión para la clasificación

En base a este tipo de situaciones se definen cuatro medidas básicas que cuentan la cantidad de casos de cada tipo: verdaderos positivos (TP, *true positive*s), verdaderos negativos (TN, *true negative*s), falsos positivos (FP, *false negative*s) y falsos negativos (FN, *false negative*s). Las definiciones de cada una de ellas son las siguientes:

$$TP = \text{card}\left(\mathcal{A} \cap \mathcal{B}\right) \tag{2.194}$$

$$TN = \text{card}\left(\left(\overline{\mathcal{A}} \cup Sh\right) \cap \overline{\mathcal{B}}\right) \tag{2.195}$$

$$FP = \text{card}\left(\left(\overline{\mathcal{A}} \cup Sh\right) \cap \mathcal{B}\right) \tag{2.196}$$

$$FN = \text{card}\left(\mathcal{A} \cap \overline{\mathcal{B}}\right) \tag{2.197}$$

donde card representa la cardinalidad o número de elementos del conjunto en cuestión.

Una medida básica asociada a la cantidad de errores producidos a causa de las sombras es SE y se define como:

$$SE = \text{card}\left(Sh \cap \mathcal{B}\right) \tag{2.198}$$

Como puede observarse, tanto TP como TN atienden a la cantidad de píxeles bien clasificados, mientras que FP y FN denotan los errores. Estas medidas son valores absolutos, por lo que al no estar ponderadas por la cantidad de píxeles que hay en la escena, no son válidas para comparar los resultados entre secuencias que utilizan un tamaño de fotograma diferente. Al centrarse solo en un aspecto del método tampoco podemos utilizarlas para comparar el grado de acierto o error del mismo. Por ejemplo, utilizando solo una de ellas no podemos saber si el método detecta bien los píxeles del primer plano, TP nos indica cuantos píxeles del primer plano están bien clasificados, pero no sabemos cuantos debería haber habido.

### 2.4.2. Medidas de rendimiento

Si bien las medidas básicas pueden ser utilizadas para estudiar el rendimiento de los métodos de detección de movimiento [147], en general es habitual utilizar unas medidas más sofisticadas que las anteriores. En particular es deseable que estas medidas sean independientes del tamaño del fotograma, proporcionen una idea clara de la bondad de un método y sean utilizadas por la mayoría de las propuestas ya existentes.

A continuación vamos a definir un conjunto de medidas estándar utilizadas para comparar el rendimiento de los métodos de detección de movimiento [148, 149]. Las dos primeras medidas son la **tasa de falsos positivos** (FPR) y la **tasa de falsos negativos** (FNR). Ambas pueden ser definidas mediante las ecuaciones siguientes:

$$FPR = \frac{\text{card}\left(\overline{\mathcal{A}} \cap \mathcal{B}\right)}{\text{card}\left(\overline{\mathcal{A}}\right)} = \frac{FP}{FP + TN} \tag{2.199}$$

$$FNR = \frac{\text{card}\left(\mathcal{A} \cap \overline{\mathcal{B}}\right)}{\text{card}\left(\mathcal{A}\right)} = \frac{FN}{FN + TP} \tag{2.200}$$

donde $FPR$, $FNR \in [0, 1]$.

Tanto en FPR como en FNR los valores cercanos a 0 son mejores que los cercanos a 1. Como puede observarse, FPR está asociado al fondo de la escena ya que FP es ponderado con la cantidad de píxeles que realmente son del fondo. De manera complementaria FNR se relaciona con el primer plano. Si un método obtiene un FPR elevado significa que tiende a clasificar mal los píxeles de fondo. Mientras que

si es FNR el que tiene un valor alto, quiere decir que tiende a fallar a la hora de detectar los objetos del primer plano.

Para usar una única medida que proporcione una idea de lo eficaz que es un método de detección de movimiento, podemos utilizar la medida estándar *accuracy* (AC) o **exactitud** y que se define como:

$$AC = \frac{\text{card}\left(\mathcal{A} \cap \mathcal{B}\right) + \text{card}\left(\overline{\mathcal{A}} \cap \overline{\mathcal{B}}\right)}{\text{card}\left(Universo\right)} = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.201}$$

donde $Universo$ denota el universo o conjunto universal, que en nuestro caso es igual a los píxeles que componen un fotograma, por lo que $\text{card}\left(Universo\right) = fil \cdot col$.

AC calcula la tasa de acierto del método independientemente de si estos son debidos al primer plano o al fondo, esto nos permite conocer la fiabilidad de las máscaras generadas. Una segmentación perfecta se daría cuando $AC = 1$, mientras que $AC = 0$ denotaría que el método ha generado una máscara que es justo la complementaria del *ground truth*. La desventaja de esta medida es que no podemos saber si el valor de esta medida para un método concreto es debido a su comportamiento con los píxeles del primer plano o a los del fondo. Lo cual puede ser útil para interpretar mejor la salida del mismo.

Relacionada con la medida AC se puede encontrar en la bibliografía la medida PWC (*Percentage of Wrong Classifications*) que no es más que el porcentaje de píxeles que no son segmentados correctamente respecto del total de píxeles. Por tanto esta medida se puede definir como:

$$PWC = 100\left(1 - AC\right) \tag{2.202}$$

Otras dos medidas estándar utilizadas habitualmente son la **precisión** (PR) y ***recall*** (RC), esta última también se conoce como tasa de verdaderos positivos (TPR) o sensibilidad. Ambas inciden en la cantidad de píxeles del primer plano reconocidos correctamente. La primera de ellas calcula la proporción respecto del total de píxeles que efectivamente han sido marcados como primer plano, mientras que la segunda indica la proporción respecto de la cantidad de píxeles que deberían haber sido reconocidos como primer plano:

$$PR = \frac{\text{card}\left(\mathcal{A} \cap \mathcal{B}\right)}{\text{card}\left(\mathcal{B}\right)} = \frac{TP}{TP + FP} \tag{2.203}$$

$$RC = TPR = \frac{\text{card}\left(\mathcal{A} \cap \mathcal{B}\right)}{\text{card}\left(\mathcal{A}\right)} = \frac{TP}{TP + FN} \tag{2.204}$$

en ambos casos el rendimiento es mejor conforme mayor sean las magnitudes.

La precisión permite saber si el método detecta los objetos en movimiento sin cometer demasiados falsos positivos. Por otro lado, un valor de RC bajo denota que no se detectan la mayoría de los píxeles que en el *ground truth* pertenecen al primer plano. Por tanto, estas medidas por si solas no son capaces de indicar si el método funciona correctamente. Por ejemplo, es un error común marcar la zona que rodea a los objetos en movimiento como primer plano (debido a los defectos de compresión o por problemas relacionados con la iluminación y sombras) en dicho caso la precisión se vería perjudicada, pero *recall* no tendría que tener un valor especialmente bajo. De forma contraria supongamos que se detecta el objeto en movimiento salvo los bordes del mismo, en dicho caso *recall* tendría un valor bajo, sin embargo la precisión sería elevada.

Representando los valores TPR y FPR de un método se obtiene la **curva ROC** (*Receiver Operating Characteristic*). Esta curva es utilizada en ocasiones para estudiar el rendimiento de los métodos de detección de movimiento [150]. Representa el rendimiento de un método de detección en base a estas dos medidas según varía el umbral de discriminación. Por ejemplo, un umbral de discriminación podría ser el valor de $T$ en los métodos basados en la sustracción del fondo.

Una combinación de PR y RC muy utilizada en el ámbito de la detección de movimiento consiste en calcular la media armónica de ambas medidas. Al resultado se le denomina **F-medida** (*F-measure*) y se define mediante la siguiente expresión:

$$FM_\beta = \frac{(\beta^2 + 1) \cdot PR \cdot RC}{\beta^2 \cdot PR + RC} \tag{2.205}$$

El parámetro $\beta$ ajusta la importancia de PR frente a RC. En esta tesis asumiremos $\beta = 1$ por ser lo habitual en el ámbito de la detección de movimiento, de modo que $FM = FM_1$. La ventaja principal de la F-medida es que con un único valor podemos comparar el rendimiento del método de detección de movimiento con el de otras propuestas. En esta tesis se utilizará principalmente la F-medida para representar el rendimiento de los métodos estudiados y de manera complementaria la exactitud.

Por último, puntualizar que existen dos opciones a la hora de calcular las medidas que acabamos de definir. Se pueden acumular los valores de TP, TN, FP y FN a lo largo de toda la secuencia y después calcular las medidas de rendimiento deseadas, o bien calcular las medidas deseadas en cada fotograma y al final de la secuencia promediar todos los valores obtenidos. En la práctica totalidad de esta tesis se ha optado por utilizar la segunda opción, no obstante marcaremos con un apostrofe aquellas medidas de rendimiento que hayan sido calculadas mediante la primera opción, por ejemplo: F-medida' o también FM'.

## 2.5. Modelado de fondo mediante aproximación estocástica

Buena parte de los modelos de fondo actuales utilizan distribuciones de mixtura para modelar la escena (ver Subsección 2.3.2). En general estos modelos asumen que todas las variables que modelan el color de un píxel tienen la misma varianza y además que son independientes entre sí. Esto se traduce en que los modelos basados en mixtura de gaussianas presentan matrices de covarianza esféricas, es decir, matrices de la forma $\sigma^2\mathbf{I}$. Es sabido que utilizar matrices de covarianza esféricas para modelar no se ajusta a las escenas reales. En [151] se determina que la estimación del vector de medias es fiable, pero la matriz de covarianza real no se ajusta a un modelo esférico. La distribución de los valores para un píxel del fondo en el espacio RGB estándar se asemeja más a un cilindro que a una esfera, lo que lleva a que el modelo esférico sea poco realista [152, 153]. Para solventar este problema se ha propuesto el modelo cilíndrico. En este modelo el eje mayor del cilindro llega hasta el color negro y la posición a lo largo de este eje se corresponde con la luminancia. La distancia de un punto al eje mayor determina el radio del cilindro, lo que se denomina distorsión del color. La desventaja del modelo cilíndrico es que no tiene asociada ninguna densidad de probabilidad, sólo se define una medida de lo cerca que está un color del modelo. De este modo no es posible determinar la probabilidad de que el píxel pertenezca al fondo. A diferencia de los modelos cilíndricos y de las gaussianas esféricas, una distribución gaussiana con la matriz de covarianza completa puede modelar una forma alargada con cualquier posición, grosor y orientación. Esto lleva a que sea una alternativa viable para reproducir la probabilidad de que un píxel pertenezca al fondo.

Como ya se indicó en la Subsección 2.3.2, una de las alternativas más habituales para modelar el fondo consiste en utilizar varias gaussianas esféricas en lugar de una. Estudios comparativos indican que utilizar más de una gaussiana esférica es mejor que utilizar una sola [154]. No obstante, la mixtura de gaussianas esféricas sólo puede conseguir una aproximación cruda al conjunto de datos que modela, salvo que se usen un número considerable de gaussianas. La mayor dificultad se encuentra en aquellos puntos que están en el espacio intermedio entre los vectores de medias de dos componentes gaussianas consecutivas, la densidad de probabilidad en esos puntos es menor que la de los cercanos a la media de una de las componentes. Al igual que antes, al utilizar una gaussiana con una matriz de covarianza completa se evita este problema ya que la densidad de probabilidad es suave a lo largo del conjunto de datos. Esto es confirmado en el estudio realizado en [155], donde se muestra que una gaussiana con una matriz de covarianza completa supera en bastantes situaciones a un modelo con varias gaussianas esféricas. Esto es debido a que las covarianzas son capaces de adaptarse bien a la inestabilidad del fondo.

En general la mixtura de gaussianas no puede modelar el primer plano correctamente a menos que se utilice una cantidad elevada de componentes. Esto se debe a que los

objetos del primer plano pueden tener cualquier aspecto, por tanto la distribución de probabilidad de cualquier mixtura utilizada para modelar el primer plano debe tener una forma plana, es decir, sin zonas prominentes. No obstante, la mayoría de los algoritmos de detección del primer plano ignoran este hecho y emplean una cantidad reducida de gaussianas esféricas para modelar el primer plano [126, 156]. Esta situación lleva a que las gaussianas que modelan el primer plano tengan una varianza muy elevada para tratar de adaptarse a la heterogeneidad de los objetos del primer plano. Además, una gran parte de la masa de probabilidad de estas gaussianas puede estar fuera del soporte de la distribución de entrada real, esto se debe a que las gaussianas deben adaptarse a las muestras de entrada que están cerca de la frontera del soporte. Por tanto existirán regiones que formen parte del modelo pero que no se correspondan con entradas posibles. Esto ocurre también en las gaussianas empleadas para modelar el fondo, sin embargo el efecto no es tan grave ya que las muestras del fondo suelen estar más concentradas, por lo que la dispersión de las gaussianas es menor y existe menos masa de probabilidad fuera del soporte real de la entrada. Una solución para este problema podría consistir en utilizar una componente plana para modelar el primer plano como por ejemplo una distribución uniforme [157, 158].

Dadas las limitaciones que acabamos de exponer de los modelos basados en gaussianas esféricas tanto para modelar el fondo como el primer plano, en [159] se propone un método para segmentar la escena donde confluyan las soluciones esbozadas previamente, dicho método lo denominaremos **AE**. En resumen, la propuesta tiene tres características esenciales:

- El primer plano se modela mediante una distribución plana, en concreto con una distribución uniforme.
- El fondo de la escena es modelado mediante una distribución gaussiana con matriz de covarianzas completa.
- El método de aprendizaje se basa en aproximación estocástica.

En las siguientes subsecciones explicaremos detenidamente cada una de las características de este modelo. En la Subsección 2.5.1 definiremos el modelo que se emplea para representar la escena. La Subsección 2.5.2 expone el algoritmo de aprendizaje utilizado para actualizar el modelo en cada fotograma. En la Subsección 2.5.3 se calcula el ruido de cuantización, mientras que la Subsección 2.5.4 versa sobre la detección de cambios repentinos en el fondo, en ambos casos el objetivo es mejorar el rendimiento del algoritmo. Por último, en la Subsección 2.5.5 se define el valor inicial de cada una de las variables que componen el modelo.

### 2.5.1. Modelo de fondo

El modelo propuesto consiste en una distribución de mixtura donde el primer plano se representa mediante una distribución uniforme y el fondo con una distribución

gaussiana. De esta forma la densidad de probabilidad $\mathrm{p}(\boldsymbol{p}_t)$ vendría dada por la siguiente expresión:

$$\mathrm{p}(\boldsymbol{p}_t) = \pi_{B,t} G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{B,t}(p), \boldsymbol{C}_{B,t}(p) + \boldsymbol{\Psi}\right) + \pi_{F,t} U\left(\boldsymbol{p}_t\right) \tag{2.206}$$

siendo $G\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ y $U\left(\boldsymbol{p}_t\right)$ las funciones de densidad de probabilidad gaussiana y uniforme, respectivamente:

$$G\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = (2\pi)^{-D/2} \det\left(\boldsymbol{\Sigma}\right) \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)\right) \tag{2.207}$$

$$U\left(\boldsymbol{p}_t\right) = \begin{cases} 1/Vol\left(\mathcal{S}\right) & \boldsymbol{p}_t \in H \\ 0 & \boldsymbol{p}_t \notin H \end{cases} \tag{2.208}$$

donde $\boldsymbol{\Psi}$ es una matriz diagonal constante que representa el ruido de cuantización y compresión (véase Subsección 2.5.3), $\boldsymbol{C}_{B,t}(p)$ es la matriz de covarianzas asociada al píxel $p$, $\mathcal{S}$ es el soporte de la función de densidad de probabilidad de la distribución uniforme y $Vol\left(\mathcal{S}\right)$ es el volumen $D$-dimensional de $\mathcal{S}$. $D$ representa la dimensión del píxel de entrada, es decir, $D = 3$ en el espacio RGB. Sean $d_i$ y $e_i$ respectivamente los valores mínimo y máximo de la i-ésima dimensión de entrada, por lo tanto $\mathcal{S}$ y $Vol\left(\mathcal{S}\right)$ se definen como:

$$H = [d_1, e_1] \times [d_2, e_2] \times [d_3, e_3] \tag{2.209}$$

$$Vol\left(\mathcal{S}\right) = \prod_{h=1}^{3} \left(d_h - e_h\right) \tag{2.210}$$

En el caso de que se utilicen 8 bits de precisión, que es lo habitual en el espacio RGB, tendríamos que $d_i = 0$ y $e_i = 255$. No obstante se pueden utilizar otros espacios de color, como por ejemplo los que se expusieron en la Sección 2.1.

La componente gaussiana de este modelo es capaz de modelar un conjunto amplio de fondos dinámicos gracias a que $\boldsymbol{C}_{B,t}(p)$ y por tanto $\boldsymbol{C}_{B,t}(p) + \boldsymbol{\Psi}$ no están restringidas a ser ser matrices diagonales. Además la componente uniforme es capaz de modelar cualquier tipo de objeto del primer plano.

Cabe destacar que en la práctica no se usa $\boldsymbol{\mu}_F$ para calcular las densidades de probabilidad. Sólo es necesario cuando se detecta un cambio repentino en el fondo de la imagen, lo cual será explicado en la Subsección 2.5.3.

Por último, las responsabilidades o probabilidad a posteriori de que un píxel corresponda al fondo o al primer plano se calcularían de la siguiente manera:

$$\forall i \in \{B, F\}\,,\ R_{i,t} = P\left(i|\boldsymbol{p}_t\right) =$$

$$\frac{\pi_i \mathrm{p}\left(\boldsymbol{p}_t|i\right)}{\pi_B \mathrm{p}\left(\boldsymbol{p}_t|B\right) + \pi_F \mathrm{p}\left(\boldsymbol{p}_t|F\right)} \tag{2.211}$$

### 2.5.2. Algoritmo de aprendizaje

Para adaptar el modelo a los cambios en la secuencia se propone utilizar un enfoque basado en la aproximación estocástica [160, 161, 162]. Existen varias propuestas para resolver el problema del modelado del fondo que se pueden relacionar con el ámbito de la aproximación estocástica. El esquema de actualización de los parámetros de los modelos descritos en [125, 126, 163] se pueden ver como algoritmos de aprendizaje basados en aproximación estocástica para la mixtura de gaussianas. Esto se debe a que la aproximación estocástica aplicada a la mixtura de gaussianas es un método recursivo para la maximización de la verosimilitud [164], lo que lleva al algoritmo propuesto en [128]. La aproximación estocástica se puede aplicar también a los mapas autoorganizados, como se demuestra en [139].

Al aplicar este enfoque al modelo definido en la subsección anterior, el algoritmo de aproximación estocástica de Robbins-Monro determina las siguientes ecuaciones de actualización:

$$\forall i \in \{B, F\}\,,\ \pi_i\left(t\right) = \left(1 - \epsilon\right)\pi_i\left(t-1\right) + \epsilon R_{i,t} \tag{2.212}$$

$$\forall i \in \{B, F\}\,,\ \mathbf{m}_i\left(t\right) = \left(1 - \epsilon\right)\mathbf{m}_i\left(t-1\right) + \epsilon R_{i,t}\boldsymbol{p}_t \tag{2.213}$$

$$\forall i \in \{B, F\}\,,\ \boldsymbol{\mu}_i\left(t\right) = \frac{\mathbf{m}_i\left(t\right)}{\pi_i\left(t\right)} \tag{2.214}$$

$$\mathbf{M}_B\left(t\right) = \left(1 - \epsilon\right)\mathbf{M}_B\left(t-1\right) +$$

$$\epsilon R_{B,t}\left(\boldsymbol{p}_t - \boldsymbol{\mu}_B\left(t\right)\right)\left(\boldsymbol{p}_t - \boldsymbol{\mu}_B\left(t\right)\right)^T \tag{2.215}$$

$$\boldsymbol{\Sigma}_{B,t}(p) = \frac{\mathbf{M}_B\left(t\right)}{\pi_B\left(t\right)} \tag{2.216}$$

donde $\epsilon$ es una constante que controla lo graduales que son los cambios en los parámetros del modelo: a mayor valor, mayores serán dichos cambios. $\mathbf{m}_i$ y $\mathbf{M}_B$ son variables auxiliares que almacenan promedios ponderados de los datos observados donde las probabilidades a posteriori son los pesos.

### 2.5.3. Estimación del ruido de cuantización

La matriz diagonal $\mathbf{\Psi}$ modela los efectos de la cuantización y la compresión que afectan a los píxeles de los fotogramas de entrada. La razón por la que se calcula de forma separada a $\mathbf{\Sigma}_B$ consiste en que estos efectos no son constantes, pueden suceder en un fotograma y desaparecer en el siguiente, por lo que el algoritmo de aprendizaje de aproximación estocástica no es capaz de modelarlos. La aproximación estocástica ignora estos cambios bruscos y no los incorpora al modelo. De este modo, la matriz de covarianzas modela los cambios que se producen en el color real del fondo de la escena, los cuales son suaves y pueden ser aprendidos mediante aproximación estocástica.

Se asume que los efectos de cuantización y de compresión son aproximadamente iguales para todos los píxeles y no varían a lo largo de una misma secuencia. Por lo tanto $\mathbf{\Psi}$ se calcula antes de comenzar el procesamiento del vídeo de entrada y es constante a lo largo del mismo, su valor viene dado por la siguiente ecuación:

$$\mathbf{\Psi} = \begin{pmatrix} \sigma_1^2 & 0 & ... & 0 \\ 0 & \sigma_2^2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & \sigma_D^2 \end{pmatrix} \tag{2.217}$$

donde los valores de $\sigma_i^2$ son estimados comparando el valor original de los píxeles sin comprimir $\boldsymbol{\tau}(p)$ con los observados en el vídeo de entrada $\boldsymbol{p}_t$:

$$\forall i \in \{1, ..., D\} \;\; \sigma_i^2 = E\left[\left(\tau_i(p) - p_{i,t}\right)^2\right] \tag{2.218}$$

donde la esperanza se calcula promediando el color de todos los píxeles y $p_{i,t}$ denota la i-ésima componente de color del píxel $p$. Nótese que en el modelo propuesto $D = 3$, por lo que $\mathbf{\Psi}$ será una matriz diagonal de orden 3.

En la práctica es habitual que los valores originales $\boldsymbol{\tau}(p)$ no estén disponibles, teniendo únicamente la versión comprimida. La solución elegida en este caso ha sido estimar el valor original aplicando una función de regresión tipo núcleo de primer orden sobre los valores observados:

$$\hat{\boldsymbol{\tau}}(p) = \frac{\sum_{q \in W(p)} K_p(q)\, q}{\sum_{q \in W(p)} K_p(q)} \tag{2.219}$$

donde $\hat{\boldsymbol{\tau}}(p)$ es el valor original estimado del píxel $p$, $W(p)$ es la ventana bidimensional centrada en la posición de $p$ y $K_p$ es una función de suavizado (para más detalles véase [159]). De este modo los valores $\sigma_i^2$ se calculan de la siguiente manera:

$$\forall i \in \{1, ..., D\} \;\; \sigma_i^2 \approx E\left[\left(\widehat{\tau}_i(p) - p_{i,t}\right)^2\right] \tag{2.220}$$

### 2.5.4. Detección de cambios repentinos en el fondo

Este método propone utilizar un procedimiento para atenuar los efectos negativos de tres tipos de dificultades: los objetos de fondo en movimiento, los objetos del primer plano parados y la inicialización deficiente (ver Subsección 2.3.1). Estos tres problemas tienen en común que en algún momento el modelo de fondo no se corresponde con lo que realmente existe en el fotograma actual y se producen falsos positivos en el área afectada. Por ejemplo, si el lugar que ocupaba un vehículo que sale de una plaza de aparcamiento es detectado como primer plano, estaríamos ante un fallo en la detección. De la misma manera si el vehículo aparca en una plaza y es detectado como primer plano durante varios fotogramas, podría considerarse un comportamiento erróneo. Como Kushner y Yin indican en [161], la aproximación estocástica en sistemas que varían con el tiempo es recomendable únicamente cuando los cambios de los parámetros estimados son suaves. De ahí que esta problemática deba ser manejada mediante un procedimiento específico que detecte cuando estamos ante una situación que no es posible tratar mediante aproximación estocástica.

En base a la detección realizada por el algoritmo básico (sin detección de cambios repentinos), se definen tres eventos aleatorios disjuntos: cambios de fondo no detectados (*Change*), detección correcta del primer plano (*Good*) y otras situaciones que no son relevantes (*Other*):

$$Change \equiv (\mathcal{M} = 1) \wedge \left(\widehat{\mathcal{M}} = 0\right) \tag{2.221}$$

$$Good \equiv (\mathcal{M} = 1) \wedge \left(\widehat{\mathcal{M}} = 1\right) \tag{2.222}$$

$$Other \equiv \mathcal{M} = 0 \tag{2.223}$$

En el fotograma $t$ se actualiza el valor de $z\,(p,t)$, el cual representa el número máximo de veces consecutivas que un píxel ha sido clasificado como fondo hasta el momento:

$$z\,(p,t) = \text{máx}\left\{N \mid \mathcal{M}\,(\boldsymbol{p}_t) = \mathcal{M}\left(\boldsymbol{p}_{t-1}\right) = ... = \mathcal{M}\left(\boldsymbol{p}_{t-N}\right) = 1\right\} \tag{2.224}$$

Básicamente $z\,(p,t)$ es un contador cuyo valor se incrementa en cada fotograma si $\mathcal{M} = 1$ y se reinicializa cuando $\mathcal{M} = 0$. Las probabilidades $P\,(Good \mid z,\, \mathcal{M} = 1)$ y $P\,(Change \mid z,\, \mathcal{M} = 1)$ se estiman de manera *offline*. La tarea consiste en contar el número de veces que se verifican *Change* y *Good* para un cierto valor de $z$, donde los valores de $z\,(p,t)$ se toman como entrada para todos los fotogramas $t$ y todos los píxeles en los que $\mathcal{M} = 1$. Los valores calculados para cada píxel son funciones de ruido que se suavizan mediante la regresión del núcleo antes de la estimación de probabilidades.

Para concluir el estudio *offline* se calcula el valor del umbral $Z$ como sigue:

$$Z = \text{mín}\left\{z \mid P\left(Change|z,\, \mathcal{M}=1\right) > \frac{1}{2}\right\} \tag{2.225}$$

Cuando el sistema está procesando el vídeo de entrada se mantiene el contador $z$. Si en el fotograma $t$ el valor de $z$ es mayor o igual que $Z$, es decir $z\left(p,t\right) \geq Z$, entonces el modelo de fondo asociado a ese píxel es reinicializado. La reinicialización consiste en asignar $\boldsymbol{C}_B = 0$, o lo que es lo mismo $\boldsymbol{\Sigma} = \boldsymbol{\Psi}$, donde $\boldsymbol{\Sigma}$ es la covarianza de la componente gaussiana de la mixtura. Nótese que $\boldsymbol{\Psi}$ nunca es cero, por lo que la inversión de la matriz $\boldsymbol{\Sigma} = \boldsymbol{C}_B + \boldsymbol{\Psi}$ será siempre posible. Como paso final de la reinicialización se asigna a $\boldsymbol{\mu}_B$ el valor de $\boldsymbol{\mu}_F$ y $\mathbf{m}_B$ se sustituye por $\mathbf{m}_F$.

### 2.5.5. Inicialización

La etapa de inicialización de este método consiste en asignar unos valores a cada una de las variables del modelo. Para llevar a cabo esta inicialización se utilizan los $L$ primeros fotogramas de cada vídeo a analizar. Las variables del modelo de fondo de cada píxel tomarán los siguientes valores:

$$\forall i \in \left\{B, F\right\},\ \pi_i\left(0\right) = \frac{1}{2} \tag{2.226}$$

$$\mathbf{m}_B\left(0\right) = \frac{\pi_B\left(0\right)}{L}\sum_{j=1}^{L}\boldsymbol{p}_j \tag{2.227}$$

$$\boldsymbol{\mu}_B\left(0\right) = \frac{\mathbf{m}_B\left(0\right)}{\pi_B\left(0\right)} \tag{2.228}$$

$$\mathbf{M}_B\left(0\right) = \frac{\pi_B\left(0\right)}{L}\sum_{j=1}^{L}\left(\boldsymbol{p}_j - \boldsymbol{\mu}_B\left(0\right)\right)\left(\boldsymbol{p}_j - \boldsymbol{\mu}_B\left(0\right)\right)^T \tag{2.229}$$

$$\mathbf{C}_B\left(0\right) = \frac{\mathbf{M}_B\left(0\right)}{\pi_B\left(0\right)} \tag{2.230}$$

La inicialización de los variables asociadas al modelo del primer plano no es crítica ya que no se utilizan al menos hasta que se procesen los $Z$ primeros fotogramas. El valor inicial de estas variables es el siguiente:

$$\boldsymbol{\mu}_F\left(0\right) = \left(\frac{1}{2}\left(d_1 + e_1\right), ..., \frac{1}{2}\left(d_D + e_D\right)\right)^T \tag{2.231}$$

$$\mathbf{m}_F\left(0\right) = \pi_F\left(0\right)\boldsymbol{\mu}_F\left(0\right) \tag{2.232}$$

Por último el contador $z$ de cada píxel es inicializado a cero $z\left(0\right) = 0$.

## 2.6. Biblioteca BGS

En esta sección vamos a hablar de la biblioteca BGS[2] [165]. Se trata de una biblioteca desarrollada por Andrews Sobral cuyo objetivo consiste en proporcionar un marco de trabajo para la detección fondo. Actualmente ofrece cerca de cuarenta métodos de detección de fondo, buena parte de ellos han sido desarrollados por diferentes autores. Es una biblioteca de código abierto y gratuita para fines académicos. Está escrita en C++ y se basa en OpenCV. Por todo ello constituye una herramienta muy útil para realizar estudios cualitativos y cuantitativos del funcionamiento de diversos métodos de detección de fondo actuales.

En esta tesis se ha utilizado la biblioteca BGS en aquellos métodos de detección de fondo cuyo código fuente o ejecutable no estaba disponible para realizar simulaciones. A continuación vamos a exponer las peculiaridades de aquellos métodos que hemos utilizado y cuya implementación tiene algunos matices que es conveniente conocer. En particular vamos a hablar de cinco métodos descritos en la Subsección 2.3.2 y uno de la Subsección 2.2.1.3.

### 2.6.1. Pfinder

El método Pfinder [57] disponible en la biblioteca BGS tiene tres particularidades a tener en cuenta. La primera de ellas consiste en que, a diferencia de la propuesta original, se considera únicamente una distribución gaussiana para modelar el fondo y no se distingue entre los distintos objetos que componen el primer plano. Esta modificación ya fue comentada en la Subsección 2.3.2. Dado que este método sólo está siendo utilizado para diferenciar entre el fondo y el primer plano, es razonable obviar el modelo de cada objeto del primer plano. De este modo se logra una simplificación del modelo que reduce el tiempo de cálculo para cada fotograma.

En segundo lugar, se utiliza un periodo de aprendizaje con una duración de $N$ fotogramas en el que idealmente se espera que no haya objetos del primer plano con el fin de que el modelo pueda adaptarse correctamente al fondo antes de que aparezcan dichos objetos. Este característica sí que es referida en el artículo original, si bien no se concreta claramente cómo implementarlo. En la biblioteca BGS se actualizarán todos los píxeles de la imagen durante este periodo, independientemente de si se ha detectado movimiento o no.

La tercera característica propia de esta implementación reside en que el valor de $\boldsymbol{\sigma}_t^2(p)$ se encuentra acotado. De este modo la Ecuación 2.98 se sustituye por la siguiente:

[2] https://github.com/andrewssobral/bgslibrary

$$\boldsymbol{\sigma}_t^2(p) = \begin{cases} 4 & sigma_t(p) < 4 \\ 180 & sigma_t(p) > 180 \\ sigma_t(p) & en\,otro\,caso \end{cases} \tag{2.233}$$

$$sigma_t(p) = \alpha \left(\boldsymbol{p}_t - \boldsymbol{\mu}_t\left(p\right)\right)^2 + (1-\alpha)\boldsymbol{\sigma}_{t-1}^2(p) \tag{2.234}$$

donde $sigma_t(p)$ es una variable auxiliar.

Para entender el objetivo de esta modificación debemos considerar la Ecuación 2.96, en ella se observa que una varianza excesivamente grande puede llevar a que el método considere como parte del fondo a prácticamente cualquier valor del píxel en cuestión. Por otro lado, una varianza demasiado pequeña haría que casi ningún color sea considerado parte del fondo. Por tanto utilizar unos valores máximo y mínimo para la varianza permite que estas dos situaciones extremas no se den o al menos que su efecto sea menor.

## 2.6.2. Modelos basados en distribuciones de mixtura

Las implementaciones de los métodos GrimsonGMM y ZivkovicGMM se ciñen en general a lo propuesto en [125] y [127, 128], respectivamente. No obstante, e reseñable que al igual que en el caso del método Pfinder, en ambos métodos se acota el valor posible de la varianza.

La siguiente diferencia modifica el método GrimsonGMM propuesto originalmente, se basa en lo publicado en [127, 128] y consiste en que el umbral utilizado para emparejar las componentes gaussianas es diferente del utilizado en la Ecuación 2.105. En concreto se utiliza un umbral ajustable $T_{\sigma^2}$, como se indica en la siguiente ecuación:

$$M_{i,t}(p) = \begin{cases} 1 & \left(\delta_{i,t}^e(p)\right)^2 < T_{\sigma^2}\sigma_{i,t}^2 \wedge \forall j \neq i \left(\left(\left(\delta_{j,t}^e(p)\right)^2 \geq T_{\sigma^2}\sigma_{j,t}^2\right) \vee \left(\pi_{j,t} < \pi_{i,t}\right)\right) \\ 0 & en\,otro\,caso \end{cases} \tag{2.235}$$

donde $\delta_{i,t}^e(p)$ es la distancia euclídea. Esta modificación no es más que una generalización de la ecuación propuesta en [125].

Además de las puntualizaciones anteriores hay que destacar los valores asignados a ciertos parámetros en la biblioteca BGS. El umbral utilizado en la Ecuación 2.110 para determinar si un píxel es fondo o primer plano es constante y su valor es $T = 0.75$. En el caso de que sea necesario crear una nueva componente para la mixtura, la varianza inicial es 36 y la probabilidad a priori es igual a la tasa de aprendizaje $\alpha$. En el caso de ZivkovicGMM, el valor de la constante, $c_T$, cuyo objetivo es eliminar las componentes menos relevantes es $c_T = 0.05$.

Por último, en el caso del método KaewGMM, la versión disponible en la biblioteca BGS es sencillamente un *wrapper* de la implementación llevada a cabo en la biblioteca OpenCV, como ocurre también con el método ZivkovicGMM. Es reseñable que en dicha implementación se establece una desviación típica mínima para cada componente, a la cual denotaremos $\sigma_{mín}$ y que no está contemplada de forma explícita en el manuscrito original [126].

### 2.6.3. Sakbot

El método Sakbot disponible en la biblioteca BGS es una versión simplificada del propuesto en [32, 33]. La diferencia estriba en que no utiliza los umbrales $T_L(\boldsymbol{p}_t)$ y $T_H(\boldsymbol{p}_t)$ cuyo valor es local a cada píxel, tal y como se expuso en la Subsección 2.3.2. En cambio usa dos constantes globales para toda la imagen, denotadas como $T_H$ y $T_L$. El valor de una está relacionado con la otra mediante la siguiente ecuación:

$$T_H = 2T_L \tag{2.236}$$

Si bien es una diferencia significativa, no altera la idea original de utilizar dos máscaras, una de ellas menos sensible a los cambios de intensidad que la otra.

Por último señalar que la cantidad de intensidades almacenadas en el *buffer* $S_t(p)$ es igual a 15, es decir, $m = 15$.

### 2.6.4. FuzzyElBaf

La implementación llevada a cabo del método FuzzyElBaf tiene varias peculiaridades que vamos a comentar a continuación. En primer lugar, se usa un esquema difuso para la actualización del modelo de fondo en lugar de uno selectivo como el que se usa en [26]. De este modo, la ecuación de actualización es la Ecuación 2.183, tal y como se propuso en [142]. Además, el valor de $\beta_t$ está determinado por la siguiente ecuación:

$$\beta_t = 1 - \left(C_\mu - \left(\frac{m_1}{m_1 - m_2} C_\mu - \frac{m_1 m_2}{m_1 - m_2}\right)\right) \tag{2.237}$$

donde $m_1$ y $m_2$ son, respectivamente, el valor mínimo y máximo de $C_\mu$ a lo largo de todos los píxeles de la imagen. Como puede observarse esta ecuación cumple con los valores máximo y mínimo definidos para $\beta_t$ (ver Subsección 2.3.2).

El modelo inicial consiste en la media de las intensidades de cada píxel de los $N$ primeros fotogramas. Se calcula de forma incremental, como recoge la siguiente ecuación:

$$\boldsymbol{B}_i(p) = \alpha_1 \boldsymbol{p}_i + (1 - \alpha_1)\, \boldsymbol{B}_{i-1}(p),\ i \in \{1, ..., N\} \tag{2.238}$$

donde $\alpha_1 \in [0, 1]$ es la tasa de aprendizaje para la fase de inicialización.

El trabajo original de Baf et al. usa tres espacios de color diferentes: Ohta, HSV and YCrCb. No obstante, la biblioteca BGS considera adicionalmente el espacio RGB. En nuestros experimentos se usaran todos estos espacios con el fin de ampliar el rango de configuraciones probadas.

Otra característica propia de la implementación llevada a cabo en BGS es que añade un tipo configuración diferente para los rasgos asociados a cada píxel. La propuesta original de Baf et al. usa dos componentes de color y una de textura para cada píxel. En la biblioteca BGS se añade la opción de usar únicamente tres componentes de color. En nuestro trabajo se probarán ambas opciones para aportar una mayor completitud.

# 3 Efectos del ruido

Los métodos de detección de fondo están expuestos a los efectos de diferentes tipos de ruido debido a las limitaciones de los dispositivos de adquisición de imágenes. El problema principal de este tipo de distorsión es que altera el color de los píxeles que componen el vídeo de entrada, lo que en principio puede empeorar la calidad de la detección. No obstante, hemos descubierto situaciones en donde añadir ruido a la entrada, en lugar de ser un problema, atenúa algunas limitaciones de los métodos.

En este capítulo se lleva a cabo un estudio sobre los efectos que produce el ruido a la hora de detectar el movimiento que, como hemos comentado anteriormente, no siempre es perjudicial. En particular hemos analizado el rendimiento de varios métodos frente a diferentes niveles de ruido gaussiano y uniforme que, como veremos en la Sección 3.1, simulan buena parte de las distorsiones que pueden suceder durante el proceso de digitalización de una imagen. Para lograr realizar un estudio suficientemente amplio se han elegido nueve métodos de detección actuales los cuales se citan en la Sección 3.2. En esta misma sección también se expondrán las características de los experimentos llevados a cabo y los resultados obtenidos. En la Sección 3.3 se realiza la discusión y por último en la Sección 3.4 se exponen las conclusiones.

## 3.1. Introducción

La segmentación de fondo es un proceso en el que una secuencia de imágenes capturada mediante un dispositivo de captación de imágenes es procesada por un método de segmentación. Por lo tanto básicamente, los problemas que dificultan la segmentación de fondo tienen su origen en uno o varios de los elementos implicados en dicho proceso, que son los siguientes: el método utilizado, la complejidad de la escena y el proceso de digitalización de la imagen.

Como vimos en la Subsección 2.3.1, en la realidad existen secuencias cuya dinámica puede afectar negativamente al resultado de la segmentación. Muchas de estas situaciones llegan a ser problemáticas en parte porque los modelos de fondo no son lo suficientemente robustos. Por ejemplo, los métodos que utilizan modelos basados en matrices de covarianza pueden sufrir problemas de sobreentrenamiento. Esta dificultad hace que el modelo de fondo se ajuste demasiado a los valores observados, de manera que los píxeles cuyas características difieren ligeramente de las del fondo son etiquetados erróneamente como primer plano. En el contexto del aprendizaje

automático este tipo de problemas se afrontan mediante técnicas de regularización de matrices de covarianza [166, 167, 168].

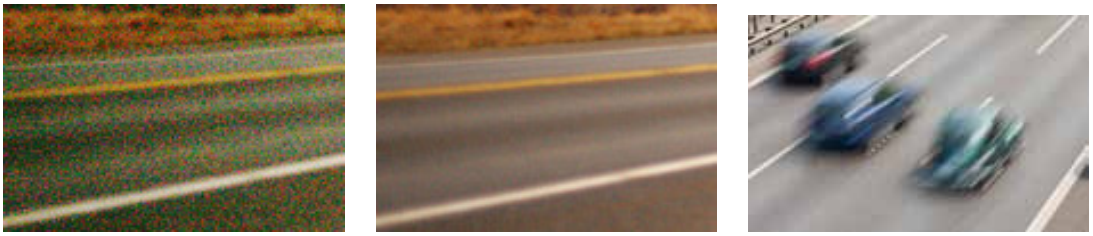


**Figura 3.1:** Dificultades en la adquisición de imágenes. La primera imagen es un caso en el que los valores de la intensidad del color no son correctos debido a un tiempo de exposición demasiado corto. La segunda imagen es la misma escena pero con un tiempo de exposición adecuado. La última imagen es un caso típico de desenfoque de los objetos en movimiento debido a un tiempo de exposición demasiado elevado. *Adaptado de www.clarkvision.com y Wikipedia.*

En este capítulo, más allá de los inconvenientes derivados de utilizar unos métodos inadecuados o de unas secuencias demasiado complejas, vamos a estudiar los problemas cuyo origen está en el proceso de digitalización de la imagen. En general este proceso puede dividirse en tres etapas, en cada una de las cuales pueden surgir diferentes complicaciones:

- **Adquisición**. En esta etapa la imagen real es captada por los sensores del dispositivo de entrada. Los problemas que pueden surgir están relacionados con procesos físicos o limitaciones del sensor [169]. Los sensores CCD (*Charge-Coupled Device*) son habituales en las cámaras digitales, su funcionamiento consiste básicamente en medir la cantidad de electrones que son elevados a un estado de energía alto debido a los fotones que inciden sobre él. Por ello, la distribución de estos fotones puede afectar al valor final del píxel. Este efecto se denomina ruido de recuento de fotones o *photon counting noise*. Una manera de aliviar este problema consiste en aumentar el tiempo de exposición de manera que la distribución de los fotones se corresponda mejor a la realidad. Sin embargo un tiempo de exposición elevado puede llevar a que la imagen se desenfoque si existe movimiento de la cámara o de los objetos de la escena (ver Figura 3.1). Los cambios de temperatura de los dispositivos de adquisición también pueden distorsionar las medidas tomadas. Esto se conoce como ruido térmico o *thermal noise*. La cantidad de electrones registrados por un sensor viene dado por la siguiente expresión:

$$n = n_I + n_{th} + n_{ro} \tag{3.1}$$

  donde $n_I$ es la cantidad de electrones debido al color de la imagen real, $n_{th}$ son los producidos por el ruido térmico y $n_{ro}$ los debidos al recuento de electrones. Cada una de estas tres variables sigue una distribución de probabilidad

**Figura 3.2:** Dificultades en el registro de imágenes. Es una imagen en escala de grises que emplea 4 bits por píxel, es decir, 16 intensidades de gris diferentes. En el cielo se observan unas franjas aproximadamente horizontales cuyo origen está en que la codificación del color ofrece una variedad de intensidades limitada, por lo que entre el color de una franja y la siguiente no existe un color intermedio, de ahí que sea perceptible por el ojo humano. *Fuente:* [169].

diferente [169]. En general las componentes $n_{th}$ y $n_{ro}$ son más significativas que $n_I$ cuando la escena presenta poca luz, por lo que una manera de atenuar estos problemas es ajustar la velocidad ISO de la cámara o bien mejorar las condiciones de iluminación.

- **Registro**. Durante la etapa de registro los estímulos detectados por los sensores son convertidos a valores discretos y almacenados en un formato manejable por los dispositivos digitales. Debido a la naturaleza continua de las variables que representan el color de los píxeles en la imagen real, no es posible evitar la cuantización de dichas variables para convertirlas en variables discretas. Los errores producidos en este proceso se conocen como ruido de cuantización. La Figura 3.2 muestra un ejemplo de esta situación.
- **Transmisión**. El ruido impulsivo está presente durante la transmisión de imágenes digitales a través de líneas con ruido [170]. La característica fundamental de este tipo de ruido es que los valores de los píxeles corruptos no guardan relación con los valores originales. No obstante la cantidad de píxeles afectados suele ser pequeña respecto de otros tipos de ruido. Existen dos tipos de ruido impulsivo: ruido sal y pimienta que distorsiona el píxel asignándole valores extremos dentro del rango de valores posibles y el ruido uniforme en el que los píxeles toman cualquier valor dentro de dicho rango. La Figura 3.3 ilustra ambos tipos de ruido. El ruido sal y pimienta se puede detectar y eliminar fácilmente, por tanto el ruido uniforme es el más interesante dentro de los ruidos impulsivos.

(a) (b) (c)

**Figura 3.3:** Dificultades en la transmisión de imágenes. (a) es una imagen en escala de grises sin distorsión, en (b) se ha aplicado ruido sal y pimienta y (c) es el resultado de aplicar ruido uniforme. *Fuente: imagen original Wikipedia e imágenes distorsionadas elaboración propia.*

Buena parte de estos problemas pueden ser corregidos o aliviados controlando la iluminación de la escena, la temperatura del sensor o la velocidad ISO. Sin embargo, en ocasiones esto no es posible o resulta demasiado costoso, por lo que en la práctica los métodos de detección de fondo deben ser capaces de enfrentarse a estas situaciones.

La mayoría de los procesos que producen ruido en una imagen digital pueden ser modelados mediante ruido aditivo gaussiano y ruido uniforme, especialmente aquellos procesos relacionados con las etapas de adquisición y transmisión [169]. El mayor problema del ruido uniforme respecto del procesamiento de la imagen consiste en que destruye completamente la información original del píxel al que afecta. Por otro lado, el ruido gaussiano preserva parte de la información original. En este sentido existen varios algoritmos que tratan de atenuar los efectos del ruido en las imágenes digitales [171, 172, 173, 174, 175, 176].

Existen trabajos como [148] donde se emplea un único nivel de ruido gaussiano para simular artificialmente el ruido del sensor en secuencias sintéticas. También hay estudios como [177] donde se llevan a cabo revisiones exhaustivas de métodos tradicionales y recientes para la detección del primer plano. En [178] no sólo se describen una gran variedad de algoritmos de detección, si no que además se realiza una evaluación de todos ellos utilizando vídeos reales y artificiales con ayuda de la biblioteca BGS. Sin embargo, el fin de este capítulo es diferente a todos ellos. Nuestro objetivo consiste en estudiar en profundidad los efectos del ruido gaussiano y uniforme en los métodos de detección del primer plano. Para lograrlo hemos puesto a prueba un conjunto de algoritmos actuales en secuencias distorsionadas con varios niveles de ruido gaussiano y uniforme. A priori los resultados pueden parecer sorprendentes, ya que hemos comprobado que añadir pequeñas cantidades de ruido puede ser beneficioso para algunos métodos. No obstante cabe señalar que este efecto ya se da en otras áreas de conocimiento. Un ejemplo de esta situación es el efecto de resonancia estocástica [179]. La resonancia estocástica tiene lugar cuando existe una señal que no es

lo suficientemente fuerte como para superar un cierto umbral de detección, pero que al añadir una pequeña cantidad de ruido, la señal resultante supera dicho umbral. Esto tiene múltiples aplicaciones, entre ellas la mejora de imágenes [180, 181].

## 3.2. Resultados experimentales

Como hemos visto en la sección anterior, existen diferentes circunstancias que hacen que las cámaras de vídeo sean susceptibles de sufrir ruido durante la adquisición, registro y transmisión de la imagen. La mayoría de los problemas asociados al proceso de digitalización pueden ser modelados mediante ruido aditivo gaussiano y uniforme [169]. Por lo que simularemos estos problemas añadiendo diferentes cantidades de ruido a secuencias reales. Para tener una visión amplia de cómo afecta el ruido a los métodos de segmentación actuales, usaremos varios métodos ampliamente conocidos.

La primera parte de esta sección describe los experimentos que se han llevado a cabo, mientras que la última parte está dedicada a los resultados obtenidos. En particular, la Subsección 3.2.1 expone las características del ruido utilizado en los experimentos. La Subsección 3.2.2 y la Subsección 3.2.3 versan, respectivamente, sobre los métodos de segmentación y los parámetros probados. En la Subsección 3.2.4 se describen brevemente las secuencias de entrada utilizadas. Por último la Subsección 3.2.5 analiza de forma cualitativa y cuantitativa los resultados obtenidos, en primer lugar se centra en el ruido gaussiano y después en el uniforme.

### 3.2.1. Tipos de ruido

El ruido aditivo gaussiano ha sido simulado añadiendo a cada canal de color una cantidad aleatoria de acuerdo con una distribución gaussiana. Se ha considerado que el ruido de cada canal es independiente de los demás. Hemos probado cuatro distribuciones gaussianas, todas ellas de media cero y con las siguientes desviaciones típicas $\sigma \in \{10, 20, 30, 40\}$ donde la intensidad de color de cada canal está en el intervalo $[0, 255]$. Por lo tanto, la intensidad del color vendrá dada por la siguiente expresión:

$$\bar{p}_{t,j} = p_{t,j} + \xi \tag{3.2}$$

$$\tilde{p}_{t,j}^{Gauss} = \begin{cases} 0 & \bar{p}_{t,j} < 0 \\ 255 & \bar{p}_{t,j} > 255 \\ \bar{p}_{t,j} & en\,otro\,caso \end{cases} \tag{3.3}$$

donde $p_{t,j}$ es el valor del píxel sin ruido en el instante $t$ para el $j$-ésimo canal de color y $\xi$ es un ruido gaussiano de media cero y desviación típica $\sigma$.

Para el caso del ruido uniforme hemos probado también cuatro escenarios diferentes donde en cada uno de ellos varía la probabilidad de que un píxel sea corrompido. Dicha probabilidad será denotada como $\varsigma$ y su valor podrá ser $\varsigma \in \{0.05, 0.1, 0.15, 0.2\}$. Al igual que en el caso gaussiano, el ruido es independiente en cada canal. Si un canal de un píxel es afectado por el ruido uniforme, el valor original es sustituido por otro de acuerdo con la distribución uniforme en el intervalo [0, 255], tal y como indica la siguiente expresión:

$$\tilde{p}_{t,j}^{Unif} = q p_{t,j} + (1 - q)\,\zeta \tag{3.4}$$

donde $\zeta$ está uniformemente distribuido en el intervalo [0, 255], $q \in \{0, 1\}$ indica si el píxel ha sido corrompido, $q = 0$ en caso de que sí y la probabilidad de que se corrompa es $\varsigma$.

Con el fin de ser justos, a la hora de corromper las secuencias de entrada, se han empleado diez semillas aleatorias diferentes para cada cantidad de ruido. Las mismas semillas han sido empleadas en todos los métodos y configuraciones, lo que garantiza tanto la equidad como la reproducibilidad.

### 3.2.2. Métodos

Hemos centrado nuestra atención en aquellos métodos de detección de fondo que tienen disponible una implementación pública y razonablemente buena, y además un alto impacto en la comunidad de visión por computador. De este modo, hemos elegido nueve métodos de segmentación, ocho de ellos están disponibles en la biblioteca BGS (ver Sección 2.6) y además de tener un nivel de impacto elevado, destacan también por su originalidad. Adicionalmente se ha utilizado el método AE para completar las pruebas ya que se trata de una propuesta publicada previamente por nuestro grupo de trabajo y también se utiliza en varias ocasiones a lo largo de esta tesis.

Dado que existe un gran número de publicaciones sobre detección de fondo que proponen métodos basados en distribuciones gaussianas, la mitad de los métodos elegidos son de este tipo, a saber: Pfinder, GrimsonGMM, ZivkovicGMM, ElgammalKDE y AE. Los cuatro primeros son descritos en detalle en la Subsección 2.3.2, mientras que AE es definido en la Sección 2.5. El método Pfinder utiliza una única distribución gaussiana para modelar el fondo y se trata de uno de los métodos de referencia básicos para la detección de fondo. El método GrimsonGMM fue pionero en utilizar varias distribuciones gaussianas para modelar el fondo. Tanto Pfinder como GrimsonGMM son habitualmente citados en otros trabajos dada su relevancia, de ahí que hayan sido considerados para analizar su comportamiento. ZivkovicGMM es un método muy conocido y de hecho es uno de los métodos de segmentación de fondo estándar en la biblioteca OpenCV. Existe una cantidad importante de métodos que utilizan estimadores basados en funciones núcleo, una de estas propuestas

es el método ElgammalKDE, el cual es además uno de los métodos más citados de entre los disponibles en la biblioteca BGS.

A parte de los métodos basados en distribuciones gaussianas, existen otras propuestas para resolver el problema de la segmentación de fondo. Por ello hemos considerado métodos que no usan este tipo de distribuciones y así realizar un estudio amplio. En concreto contemplamos cuatro métodos, que son: SOBS, FuzzyElBaf, PCAOliver y Sakbot. Los tres primeros son descritos en detalle en la Subsección 2.3.2 y Sakbot es definido en la Subsección 2.2.1.3. SOBS modela el fondo mediante una red neuronal artificial. Sakbot propone un modelo basado en la mediana de las intensidades de los píxeles. FuzzyElBaf extrae rasgos de color y de texturas de la imagen con el fin de detectar cambios en la secuencia que denoten movimiento de objetos del primer plano. PCAOliver es una propuesta que utiliza análisis de componentes principales para modelar el fondo de la escena. Estos cuatro métodos, además de utilizar un enfoque diferente al de la distribución de mixtura, son los más citados de entre los métodos no gaussianos disponibles en la biblioteca BGS.

En el Cuadro 3.1 se muestra un resumen con las características clave de cada uno de los modelos probados.

| Método | Características clave del modelo de fondo |
|---|---|
| Pfinder [57] | Una única distribución gaussiana |
| GrimsonGMM [125] | K distribuciones gaussianas |
| ZivkovicGMM [127, 128] | Número de distribuciones variable |
| AE [159] | Una distribución gaussiana usando aproximación estocástica de Robbins–Monro |
| ElgammalKDE [135] | Estimador basado en funciones núcleo |
| SOBS [138] | Redes neuronales artificiales |
| Sakbot [32, 33] | Mediana de las intensidades de los píxeles |
| FuzzyElBaf [26] | Rasgos de color y textura agregados mediante la integral de Choquet |
| PCAOliver [99] | Análisis de componentes principales |

**Cuadro 3.1:** Resumen de las características clave de cada modelo.

Los experimentos han sido implementados utilizando lenguaje C++ con la versión 1.9.0 de la biblioteca BGS y la versión 2.4.8 de OpenCV, salvo el método AE que usa *scripts* Matlab y ficheros MEX para aquellas partes más demandantes de tiempo de CPU, su código fuente está disponible en la Web[1]. La ejecución se ha llevado a cabo en un PC de sobremesa equipado con un procesador quad core 3.10 GHz, 6 GB RAM y hardware estándar. En ningún caso se realiza post-procesamiento adicional.

[1] http://www.lcc.uma.es/~ezeqlr/backsa/backsa.html

### 3.2.3. Selección de parámetros

Para cada método se han considerado los parámetros más relevantes tomando como punto de partida los manuscritos originales. El conjunto de valores probados para cada uno de estos parámetros ha sido elegido de acuerdo con los valores recomendados en la documentación y además se han probado los valores por defecto de la biblioteca BGS.

El Cuadro 3.2 muestra los valores probados para cada parámetro. La combinación de todos ellos conforma el conjunto de todas las configuraciones probadas. Se ha buscado que la cantidad de valores probados de cada parámetro sea similar para todos los métodos, salvo en AE que se usan más porque en el artículo original se recomiendan explícitamente esos valores.

| Método | Parámetros |
|---|---|
| SOBS | Fotogramas para ordenación, $N = \{55, 100\}$<br>Umbral durante la ordenación, $\epsilon_1 = \{100, 245\}$<br>Umbral durante la convergencia, $\epsilon_2 = \{30, 75\}$<br>Tasa de aprendizaje durante la convergencia, $\alpha_2 = \{50, 75\}$ |
| Sakbot | Umbral, $T_L = \{25, 30, 35\}$<br>Frecuencia de muestreo, $\nabla\tau = \{5, 10, 15\}$ |
| FuzzyElBaf | Espacio de color: RGB, Ohta, HSV, YCrCb |
| Pfinder | Umbral, $T = \{10, 12, 14, 16\}$<br>Tasa de aprendizaje, $\alpha = \{0.0001, 0.005, 0.01\}$ |
| ZivkovicGMM | Umbral de emparejamiento, $T_{\sigma^2} = \{20, 25, 30\}$<br>Tasa de aprendizaje, $\alpha = \{0.0005, 0.001\}$<br>Número máximo de componentes en la mixtura del modelo, $K = \{3, 5\}$ |
| GrimsonGMM | Umbral de emparejamiento, $T_{\sigma^2} = \{8, 9, 12\}$<br>Tasa de aprendizaje, $\alpha = \{0.0025, 0.01\}$<br>Número de componentes gaussianas en la mixtura del modelo, $K = \{3, 5\}$ |
| AE | Tamaño del paso,<br>$\epsilon = \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$ |
| ElgammalKDE | Umbral, $T = \{10^{-7}, 10^{-8}, 10^{-9}\}$<br>Umbral para detección de sombras, $\beta_T = \{0.2, 0.3, 0.5\}$<br>Cantidad de intensidades recientes de un píxel, $N = \{50, 75\}$ |
| PCAOliver | Umbral, $T = \{200, 225, 250\}$<br>Tamaño del conjunto inicial de fotogramas, $N = \{20, 100\}$<br>Número de autovectores elegidos, $M = \{10, 15\}$ |

**Cuadro 3.2:** Valores de los parámetros utilizados en los experimentos. La combinación de todos ellos conforma el conjunto de todas las configuraciones probadas para cada método.

### 3.2.4. Secuencias

Para tener una visión amplia del rendimiento de los métodos de segmentación de fondo hemos utilizado un conjunto de diez secuencias en las que se muestran situaciones muy variadas. Todas ellas están disponibles en el sitio web[2] del repositorio ChangeDetection.net (repositorio 2014) [149], el cual es ampliamente utilizado dentro del ámbito de la segmentación de fondo.

El Cuadro 3.3 muestra la categoría y las características básicas de los vídeos probados. Dentro de la categoría *baseline* las secuencias presentan situaciones de baja complejidad donde los métodos no deben tener problemas para realizar una segmentación correcta. Las secuencias Canoe y Fountain02 son más complejas debido a que presentan un fondo dinámico que hace que los métodos tiendan a producir una cantidad elevada de falsos positivos. De hecho, durante la mayor parte del tiempo no hay objetos del primer plano en la secuencia Fountain02, luego el movimiento detectado en esos periodos de tiempo se debe al fondo de la escena. Parking y Sofa son especialmente difíciles porque los objetos del primer plano permanecen estáticos durante un tiempo, para luego comenzar a moverse. Esta situación se conoce como *sleeping person* (ver Subsección 2.3.1) y hace que los métodos subsuman como fondo los píxeles implicados, lo cual es un error. En la categoría *shadow* el problema principal es que la sombra proyectada por los objetos del primer plano puede ser segmentada como parte del primer plano. Además en la secuencia PeopleInShade los objetos del primer plano son oscurecidos por el techo, lo que en ocasiones dificulta su detección. Finalmente, el vídeo Turnpike recoge la grabación de una cámara de tráfico que se caracteriza por una tasa de fotogramas por segundo baja, por tanto los objetos del primer plano no se desplazan de forma suave a largo de la escena y en la mayoría de los casos sólo aparecen durante un fotograma.

### 3.2.5. Resultados

El objetivo de esta subsección consiste en analizar el rendimiento de los métodos conforme se varía el nivel de ruido gaussiano y uniforme. Adicionalmente comparamos los resultados de cada método para señalar cuál es el que, utilizando las configuraciones mostradas en el Cuadro 3.2, obtiene mejor rendimiento en la mayoría de las situaciones.

Las Figuras 3.4-3.7 muestran el rendimiento de las mejores configuraciones para cada método y nivel de ruido en términos de F-medida. En la Figura 3.8 se representa la exactitud media en las diez secuencias, donde dicha medida se ha usado como criterio para elegir la mejor configuración en cada secuencia. La Figura 3.9 es análoga a la anterior, pero considerando la F-medida como criterio y medida de rendimiento. Las Figuras 3.10 y 3.11 muestran la FPR y FNR medias al utilizar las mejores configuraciones en términos de F-medida.

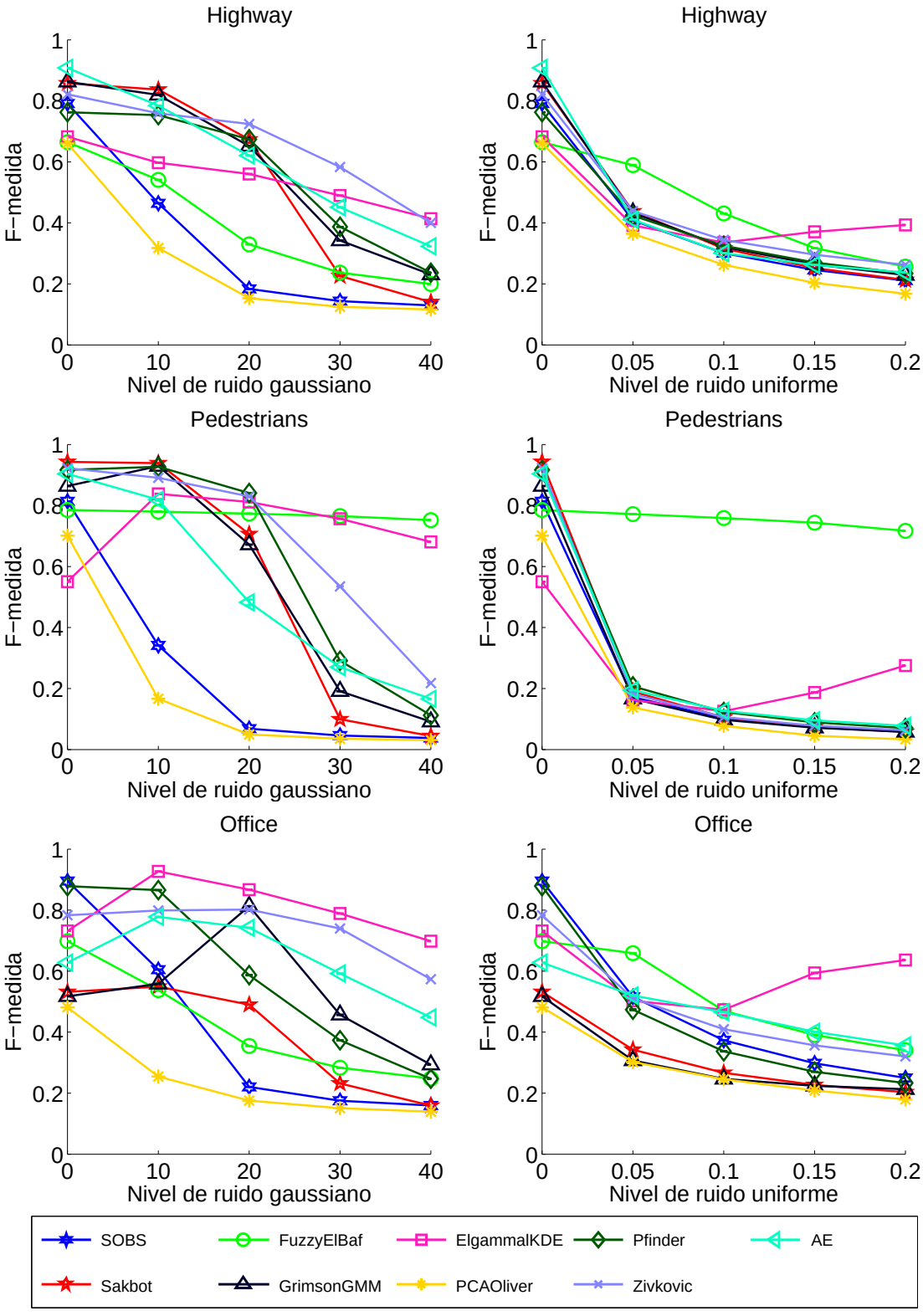



**Figura 3.4:** Rendimiento de la mejor configuración para cada método y nivel de ruido en términos de F-medida. La primera columna corresponde a los experimentos con ruido gaussiano, mientras que la segunda a los de ruido uniforme. De arriba abajo: Highway, Pedestrians y Office.

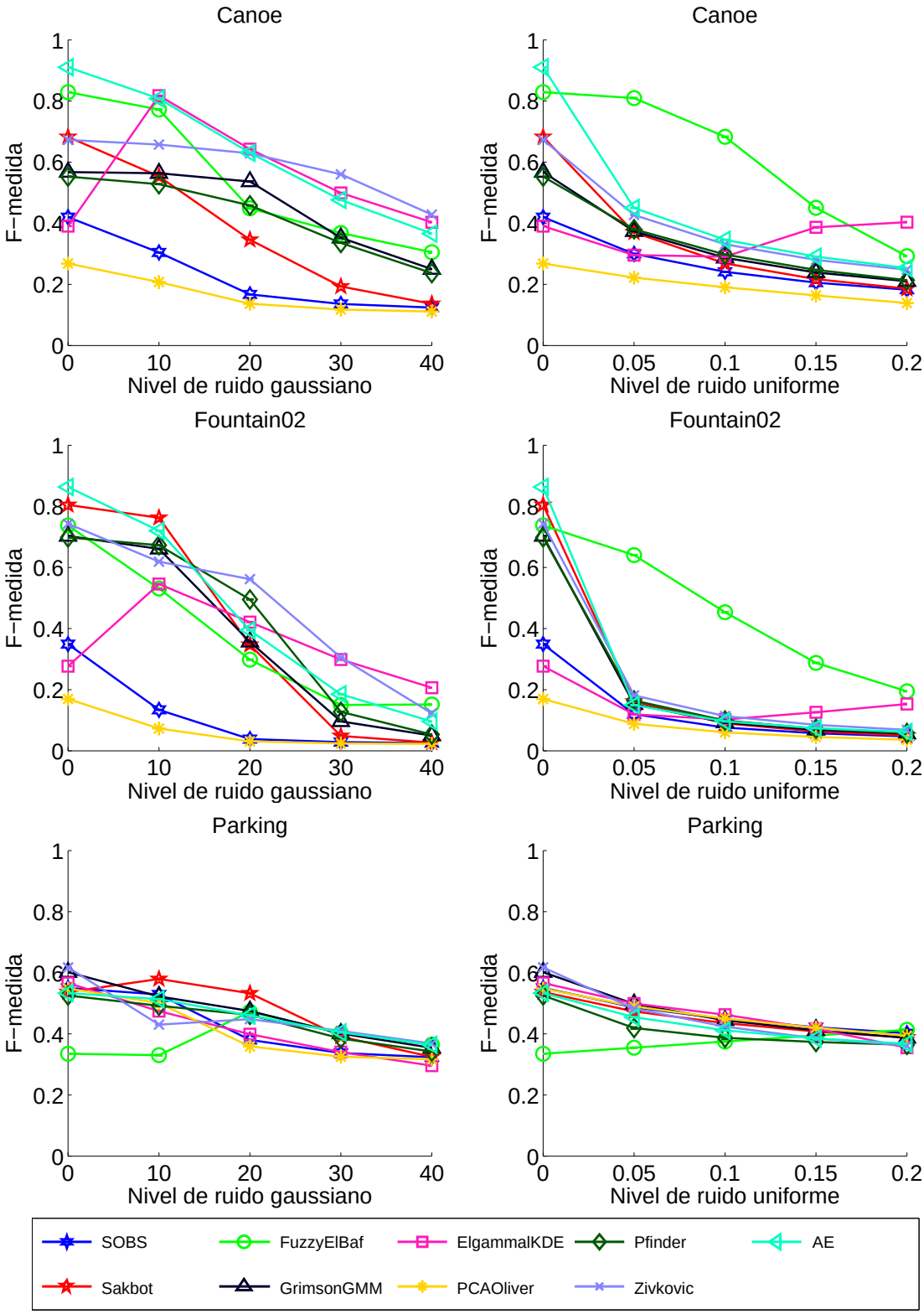



**Figura 3.5:** Rendimiento de la mejor configuración para cada método y nivel de ruido en términos de F-medida. La primera columna corresponde a los experimentos con ruido gaussiano, mientras que la segunda a los de ruido uniforme. De arriba abajo: Canoe, Fountain02 y Parking.

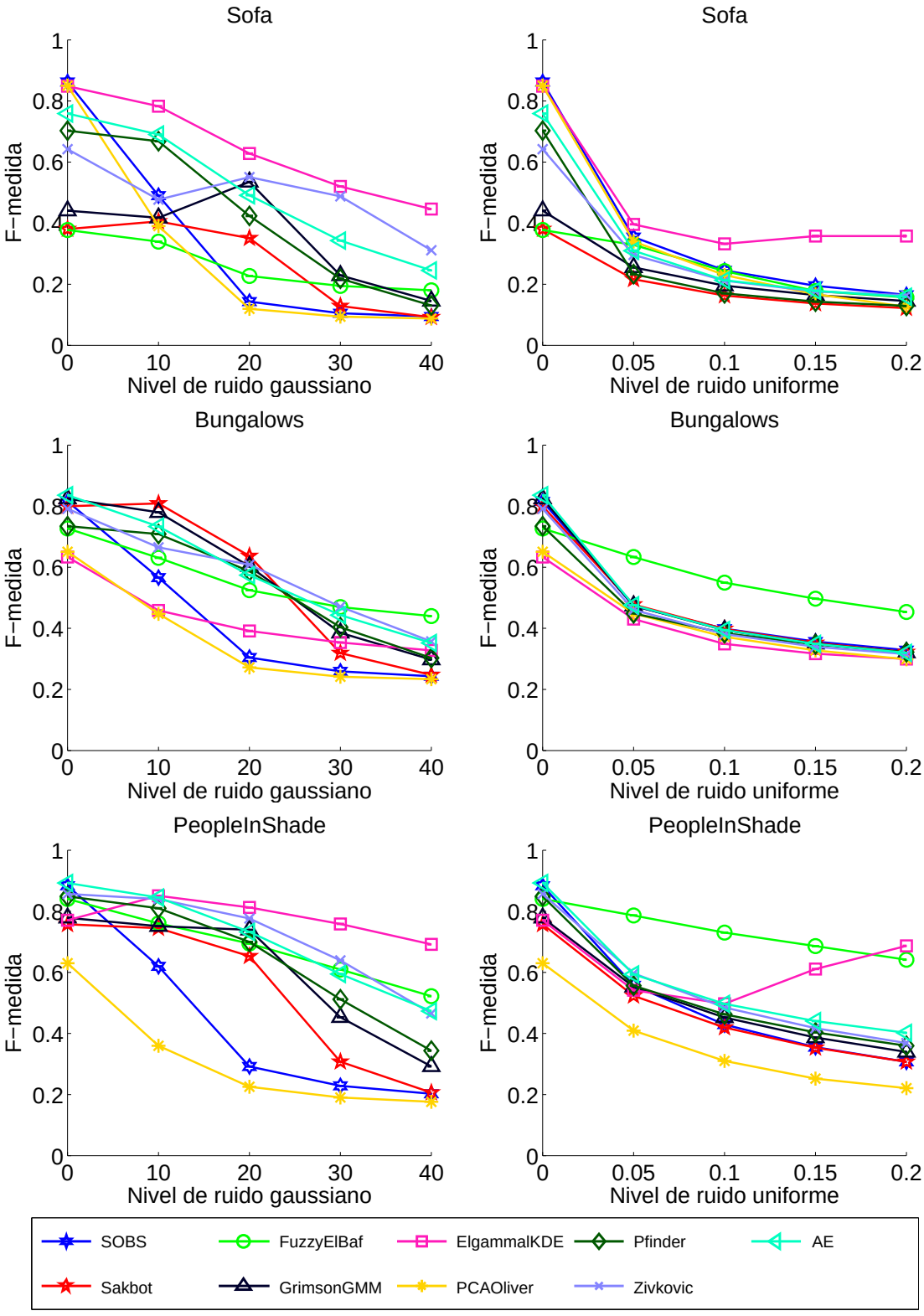



**Figura 3.6:** Rendimiento de la mejor configuración para cada método y nivel de ruido en términos de F-medida. La primera columna corresponde a los experimentos con ruido gaussiano, mientras que la segunda a los de ruido uniforme. De arriba abajo: Sofa, Bungalows y PeopleInShade.

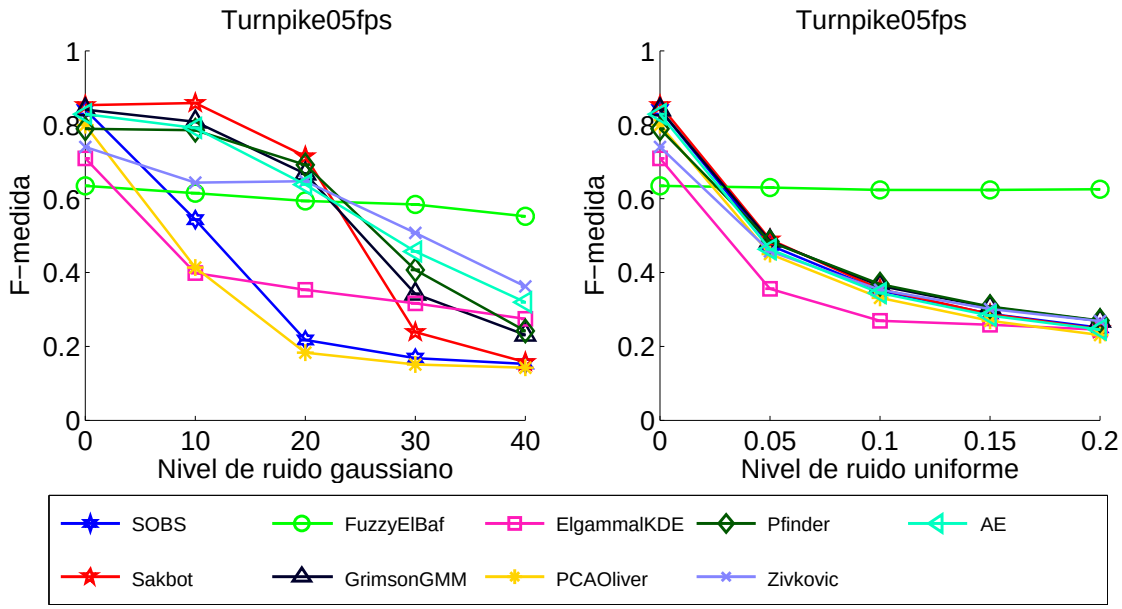



**Figura 3.7:** Rendimiento de la mejor configuración para cada método y nivel de ruido en términos de F-medida en la secuencia Turnpike_0_5fps. La primera columna corresponde a los experimentos con ruido gaussiano, mientras que la segunda a los de ruido uniforme.

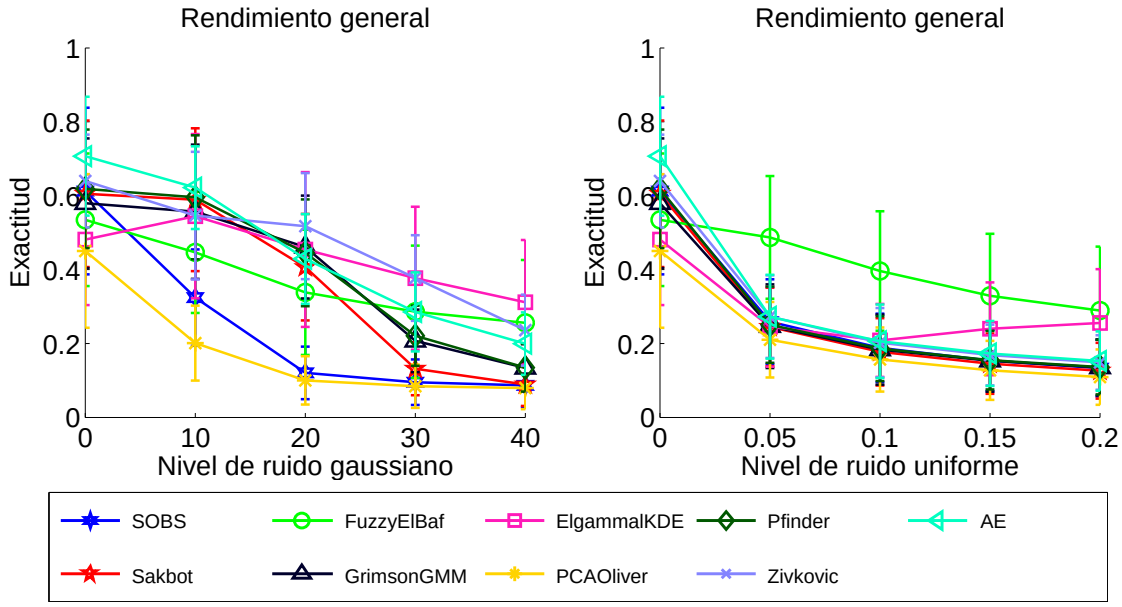



**Figura 3.8:** Rendimiento medio de los métodos en presencia de ruido gaussiano y uniforme en términos de exactitud. En cada caso se utiliza la mejor configuración para dicha medida de rendimiento.

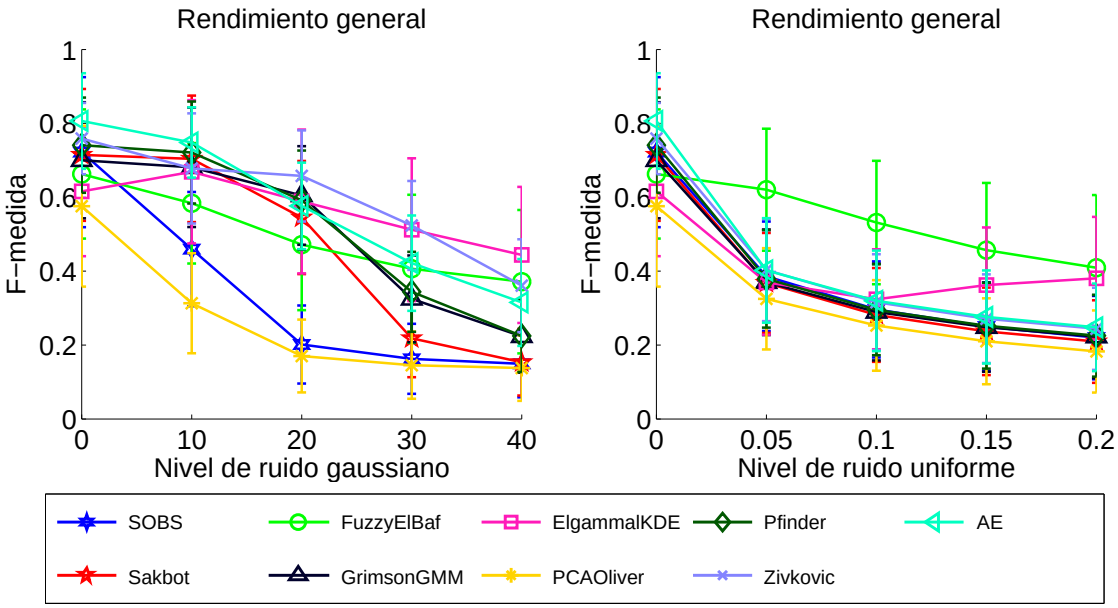



**Figura 3.9:** Rendimiento medio de los métodos en presencia de ruido gaussiano y uniforme en términos de F-medida. En cada caso se utiliza la mejor configuración para dicha medida de rendimiento.

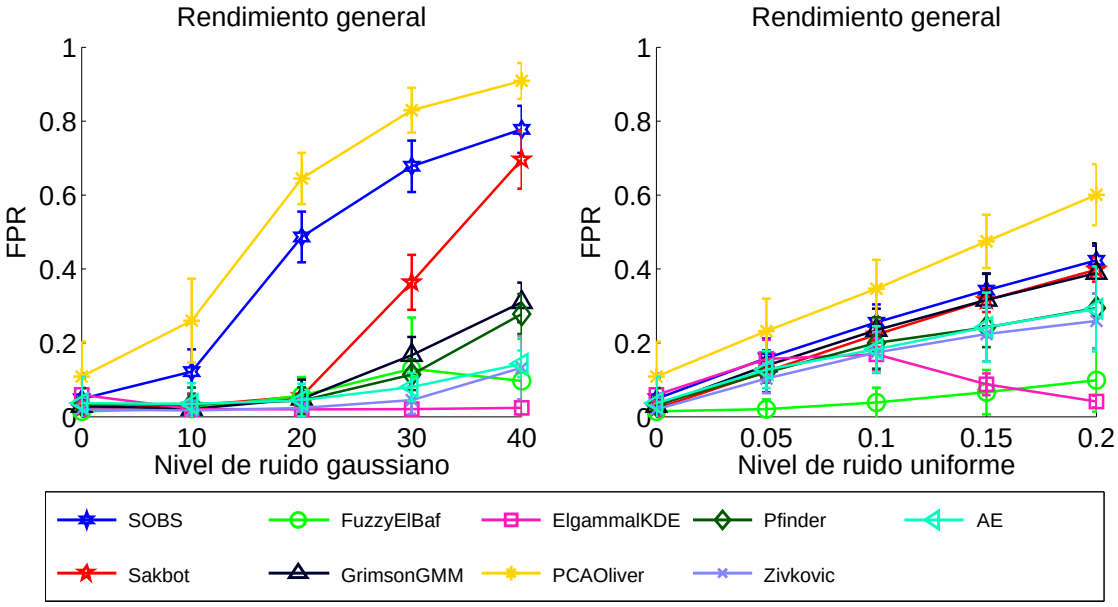



**Figura 3.10:** Rendimiento medio de los métodos en presencia de ruido gaussiano y uniforme en términos de FPR. En cada caso se utiliza la mejor configuración según F-medida.

| Nombre | Categoría | Tamaño | Duración |
|---|---|---|---|
| Highway | Baseline | 320x240 | 1700 |
| Pedestrians | Baseline | 360x240 | 1099 |
| Office | Baseline | 360x240 | 2050 |
| Canoe | Dynamic background | 320x240 | 1189 |
| Fountain02 | Dynamic background | 432x288 | 1499 |
| Parking | Intermittent object motion | 320x240 | 2500 |
| Sofa | Intermittent object motion | 320x240 | 2750 |
| Bungalows | Shadow | 360x240 | 1700 |
| PeopleInShade | Shadow | 380x244 | 1199 |
| Turnpike | Low frame rate | 320x240 | 1500 |

**Cuadro 3.3:** Características principales de las secuencias estudiadas. De izquierda a derecha: nombre de la secuencia, categoría dentro del repositorio, tamaño del fotograma en píxeles y duración de la secuencia en fotogramas.

El resto de figuras están dedicadas a mostrar las salidas de los métodos en diferentes situaciones. La Figura 3.12 compara las salidas de los diferentes métodos en una situación con alto nivel de ruido gaussiano. Las Figuras 3.13 y 3.14 comparan el ruido gaussiano y uniforme en algunas situaciones interesantes. La Figura 3.18 compara las salidas de los diferentes métodos en una situación con alto nivel de ruido uniforme. Las Figuras 3.19 y 3.20 están dedicadas a estudiar algunos efectos destacados del ruido uniforme.

Para complementar esta subsección, se han colgado en la Web[3] las secuencias de salida completas correspondientes a las Figuras 3.12 y 3.18.

#### 3.2.5.1. Ruido gaussiano

El rendimiento de los métodos en los experimentos con ruido gaussiano se muestra en la primera columna de las Figuras 3.4-3.7. En general los métodos probados obtienen buenos resultados ante un nivel de ruido $\sigma \leq 10$. Sin embargo, el comportamiento de cada método varía a partir de ese punto. La Figura 3.12 pone de manifiesto que la mayoría de los métodos tienen problemas en situaciones con alto nivel de ruido gaussiano. A continuación analizaremos las peculiaridades de cada método en cada situación.

Pfinder y GrimsonGMM muestran un comportamiento similar ante el ruido gaussiano. En ambos casos su rendimiento empeora especialmente cuando el nivel de ruido gaussiano está en el intervalo $30 \leq \sigma \leq 40$. Es destacable el aumento de rendimiento que experimenta GrimsonGMM en las secuencias Office y Sofa al añadir ruido gaussiano de desviación típica $\sigma = 20$. Como se muestra en la Figura 3.13, el

[2]http://www.changedetection.net/
[3]http://www.lcc.uma.es/ %7Eezeqlr/noise/noise.html

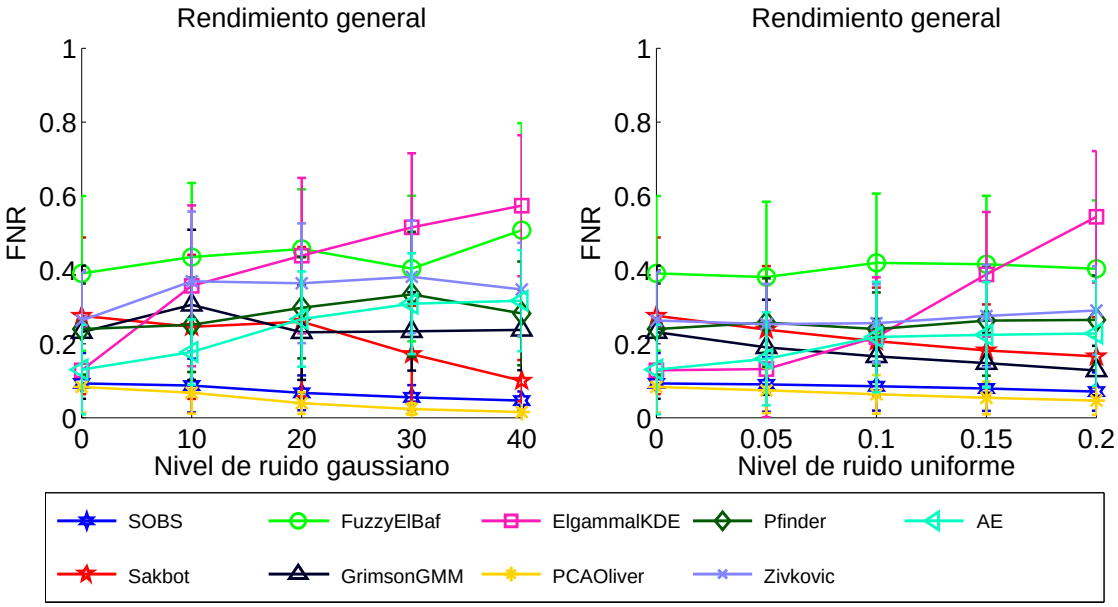



**Figura 3.11:** Rendimiento medio de los métodos en presencia de ruido gaussiano y uniforme en términos de FNR. En cada caso se utiliza la mejor configuración según F-medida.

método genera una cantidad importante de falsos negativos en situaciones sin ruido. Sin embargo, cuando $\sigma = 20$ este problema es atenuado a costa de un leve aumento del número de falsos positivos que en la práctica es despreciable.

Sakbot logra buenos resultados si el nivel de ruido es bajo, es decir, $10 \leq \sigma \leq 20$. Incluso mejora su rendimiento en términos de falsos negativos cuando se añade un poco de ruido gaussiano a la entrada. Esto se puede apreciar en la Figura 3.14. Aquí vemos que cuando no hay ruido, la segmentación es pobre porque no detecta bien los objetos del primer plano, pero al añadir ruido, la detección mejora. No obstante, el rendimiento cae de forma brusca cuando $\sigma \geq 30$ . Esto se debe a que a partir de ese nivel de ruido, el descenso de su FNR no es suficiente para compensar el gran aumento de su FPR (ver Figuras 3.10-3.11).

ElgammalKDE obtiene resultados pobres en situaciones sin ruido, sin embargo su rendimiento aumenta en seis de los diez vídeos cuando se corrompe la entrada con ruido gaussiano de desviación típica $\sigma = 10$. Aparentemente este método tiene una alta sensibilidad a la comprensión de vídeo, lo que redunda en un número significativo de falsos positivos. Una solución a este problema consiste en añadir una pequeña cantidad de ruido gaussiano a la entrada. En las columnas primera y segunda de la Figura 3.15 puede apreciarse esta mejoría. Por otro lado ElgammalKDE también tiene una buena tolerancia al ruido gaussiano, permitiéndole ser una de las mejores alternativas en situaciones con un alto grado de ruido gaussiano, tal y como se puede ver en las Figuras 3.4-3.7.

El método FuzzyElBaf presenta un comportamiento particularmente interesante.

En algunas secuencias logra muy buenos resultados gracias a su robustez frente al ruido, es decir, la caída del rendimiento a causa del aumento del ruido gaussiano es sensiblemente menor que en otros métodos. Esta situación puede observarse en las tres primeras columnas de la Figura 3.16, donde la salida de FuzzyElBaf es prácticamente independiente de la cantidad de ruido gaussiano que se aplique a la entrada. A pesar de ello, hay otras secuencias donde el rendimiento no es tan estable frente al aumento de ruido e incluso es peor que el de otros métodos. En este sentido destaca cómo degenera el rendimiento en algunas secuencias cuando se aumenta el

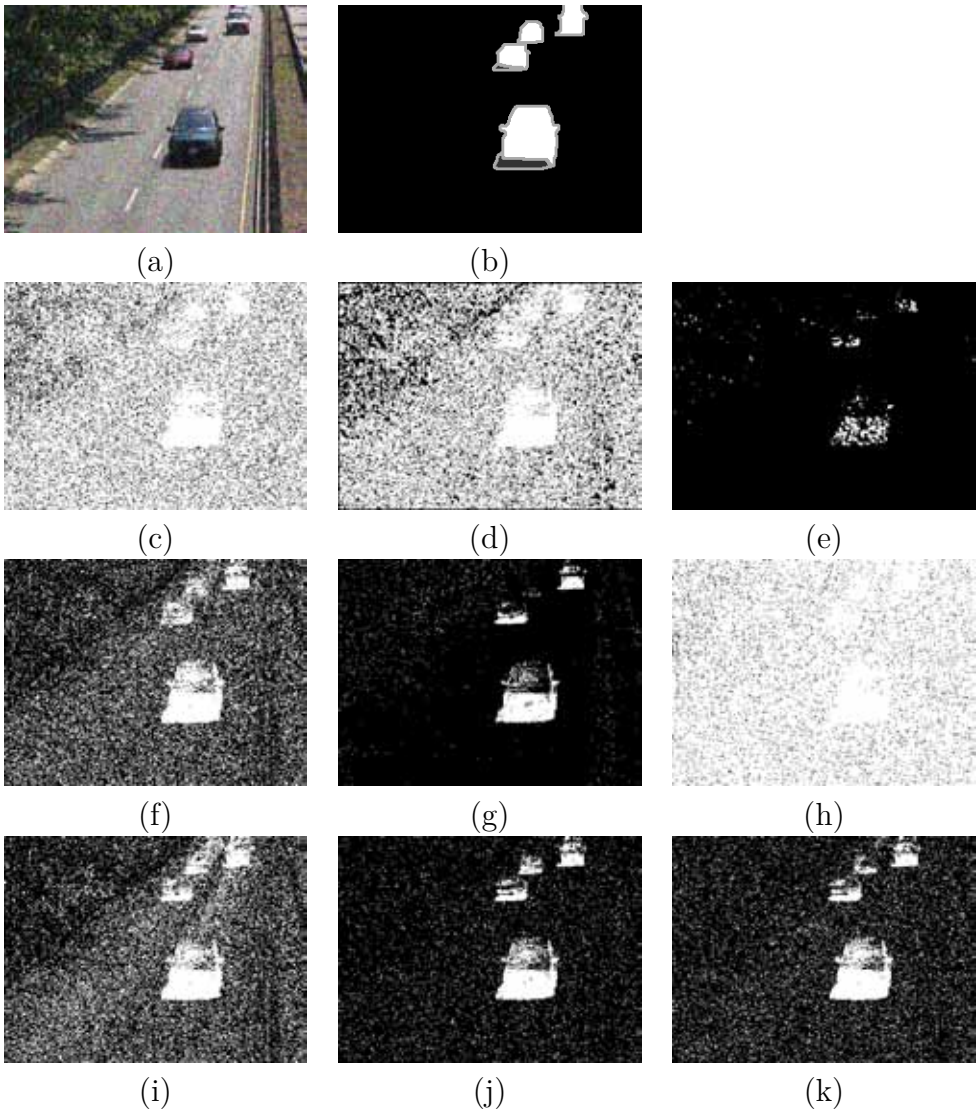



**Figura 3.12:** Salidas de los métodos en situaciones con altos niveles de ruido gaussiano. Fotograma 1400 de la secuencia highway. (a) entrada, (b) *ground truth*, (c) SOBS, (d) Sakbot, (e) FuzzyElBaf, (f) GrimsonGMM, (g) ElgammalKDE, (h) PCAOliver, (i) Pfinder, (j) ZivkovicGMM, (k) AE. Se añade un ruido gaussiano de $\sigma = 40$. En cada caso se utiliza la mejor configuración según F-medida.

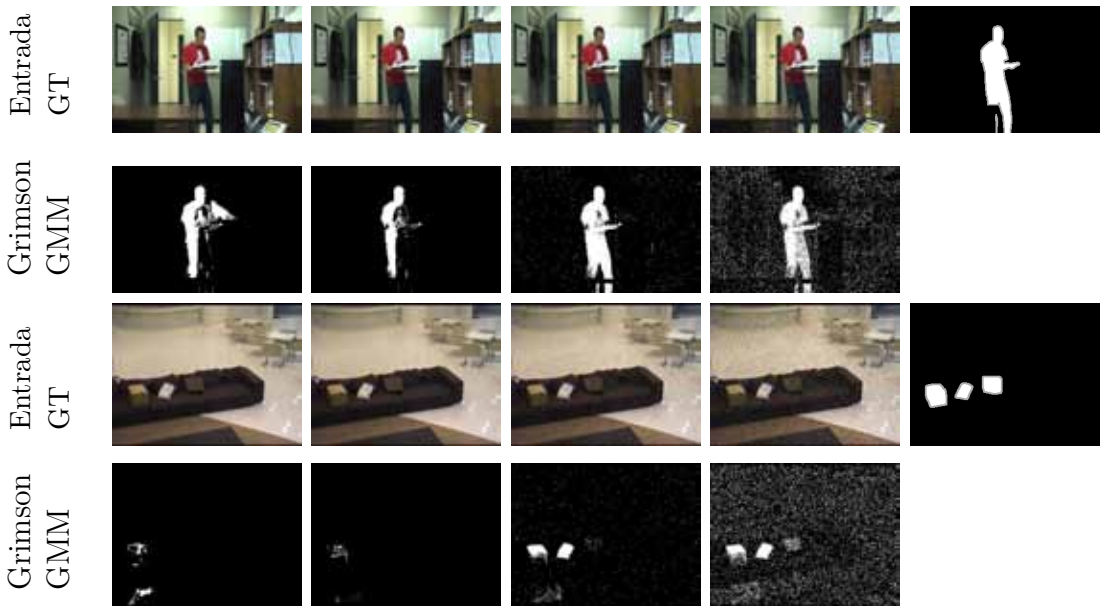



**Figura 3.13:** Salidas del método GrimsonGMM en situaciones con ruido gaussiano. La primera y segunda filas corresponden al fotograma 1300 de la secuencia Office, la tercera y cuarta filas al fotograma 1800 de Sofa. De izquierda a derecha: entrada sin ruido, ruido con desviación típica de 10, 20 y 30 y *ground truth*. Las salidas mostradas en las columnas uno, dos y tres corresponden a la configuración óptima cuando $\sigma = 20$, mientras que en la última columna se muestra la mejor configuración para ese nivel de ruido. Este método mejora su salida en términos de FNR tanto en Office como en Sofa cuando se añade ruido gaussiano con una desviación típica $\sigma \geq 20$. En concreto se consigue un buen equilibrio entre FPR y FNR cuando $\sigma = 20$.

nivel de ruido, por ejemplo en Highway, Office o en las secuencias pertenecientes a la categoría *dynamic background*. Aparentemente en estas situaciones el modelo no es capaz de modelar las partes más oscuras de la escena. Esta circunstancia se evidencia en la segunda columna de la Figura 3.17, donde el método FuzzyElBaf no modela correctamente la zona oscura de la escena y el espacio de color óptimo, en este caso YCrCb, produce una gran cantidad de falsos negativos. Como vemos en dicha figura, hay otros métodos como AE que rinden mejor en estas circunstancias, pero a costa de un número elevado de falsos positivos. Así pues, en general FuzzyElBaf es superado por otras propuestas en situaciones con poco o ningún ruido, pero frente a un nivel de ruido alto $\sigma = 40$ es una de las mejores alternativas.

ZivkovicGMM es uno de los mejores métodos en ausencia de ruido, si bien es cierto que reduce su rendimiento cuando $\sigma \geq 30$, principalmente debido a un incremento de FPR (ver la primera columna de la Figura 3.10). No obstante, la degradación no es suficientemente importante como para mermar su buen comportamiento, tanto es así que sigue logrando muy buenos resultados en situaciones con mucho ruido.

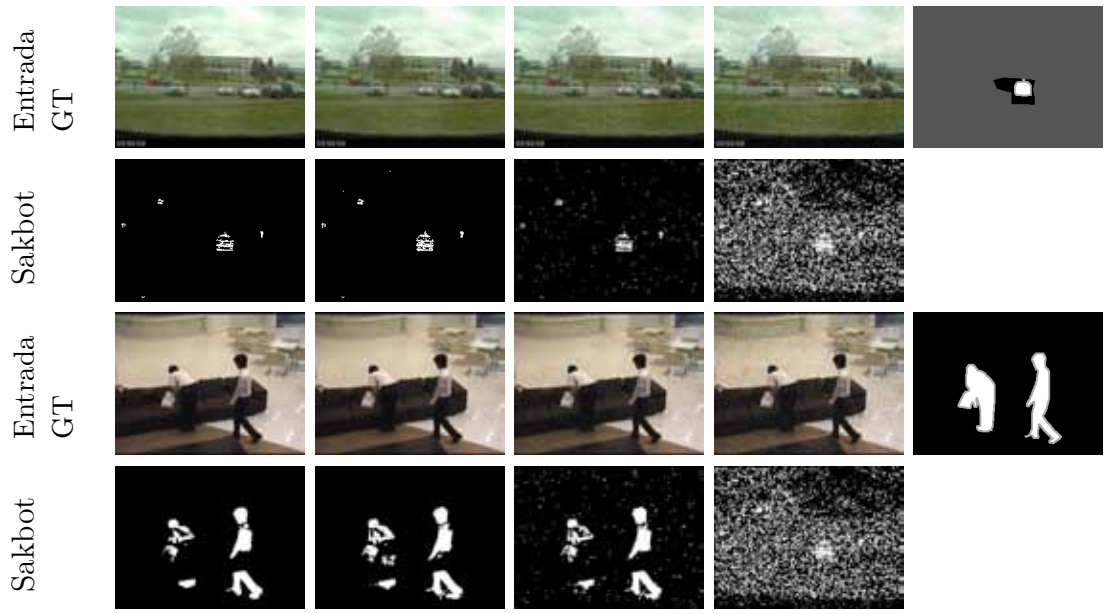



**Figura 3.14:** Salidas del método Sakbot en situaciones con ruido gaussiano. La primera y segunda filas corresponden al fotograma 1350 de la secuencia Parking, la tercera y cuarta filas al fotograma 2400 de Sofa. De izquierda a derecha: entrada sin ruido, ruido con desviación típica de 10, 20 y 30 y *ground truth*. Las salidas mostradas en las columnas uno y dos corresponden a la configuración óptima cuando $\sigma = 10$, mientras que la tercera y cuarta corresponden a las mejores configuraciones para esos niveles de ruido. En este caso Sakbot mejora su FNR al corromper la entrada con un ruido gaussiano de desviación típica $\sigma = 10$.

En ausencia de ruido AE es uno de los mejores métodos en la mayoría de las secuencias. También tiene una buena tolerancia al ruido, aunque menor que la de FuzzyElBaf o ElgammalKDE. El método AE sufre un aumento de su FPR al incrementar el nivel de ruido gaussiano. A pesar de ser de los mejores métodos en ausencia de ruido, incluso mejor que ZivkovicGMM, su tolerancia al ruido es menor que la de este ante pequeños niveles de ruido gaussiano. Su FNR es contenida, pero la FPR es mayor que el de los métodos mencionados (ver la primera columna de las Figuras 3.10 y 3.11). Como resultado, su rendimiento en situaciones con mucho ruido es peor que el de otras alternativas como ElgammalKDE, FuzzyElBaf o ZivkovicGMM.

PCAOliver y SOBS son las propuestas que han obtenido peores resultados. En la mayoría de las secuencias el método PCAOliver tiene los peores resultados independientemente del nivel de ruido. Por su parte SOBS logra unos resultados comparables al resto de competidores cuando no se corrompe la entrada, excepto en los vídeos de la categoría *dynamic background* donde obtiene unos resultados pobres. Ambas alternativas sufren un incremento muy intenso de su FPR cuando se añade ruido. Como resultado SOBS experimenta una caída brusca de su rendimiento y PCAOliver empeora aún más los pobres resultados que obtenía en ausencia de ruido.

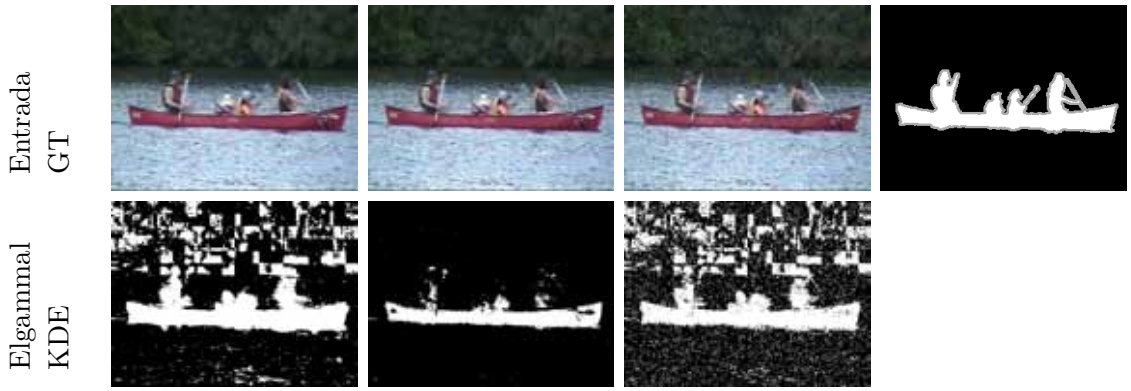



**Figura 3.15:** Salidas del método ElgammalKDE en situaciones con ruido. Las imágenes corresponden al fotograma 964 de la secuencia Canoe. De izquierda a derecha: entrada sin ruido, ruido gaussiano con una desviación típica de 10, ruido uniforme con probabilidad 0.05 y *ground truth*. Las salidas mostradas corresponden a la mejor configuración en cada situación. ElgammalKDE sufre problemas a causa de la compresión de vídeo, pero se pueden resolver añadiendo una pequeña cantidad de ruido gaussiano, sin embargo el ruido uniforme no funciona del mismo modo.

Por último hay que notar que no existen diferencias significativas entre el comportamiento de los métodos estudiados en términos de F-medida y el obtenido al utilizar exactitud como medida de rendimiento. Esto se comprueba al comparar la primera columna de las Figuras 3.8-3.9.

#### 3.2.5.2. Ruido uniforme

El ruido uniforme tiene dos características esenciales: no afecta a todos los píxeles y los píxeles distorsionados pueden sufrir un cambio abrupto en su color. Ambas características son opuestas a lo que ocurre en el caso del ruido gaussiano: afecta a todos los píxeles de la imagen y los cambios en el color son suaves. Esta realidad condiciona el rendimiento de los métodos estudiados en presencia de ruido uniforme.

En la mayoría de los casos un alto nivel de ruido uniforme produce un gran deterioro del rendimiento, tal y como ilustra la Figura 3.18. Incluso cuando la entrada es corrompida con un nivel bajo de ruido, el rendimiento decrece sensiblemente. Un ejemplo de este extremo lo vemos en la Figura 3.19 en donde se observa el bajo rendimiento del método AE en presencia de pequeñas cantidades de ruido uniforme, si bien el resto de métodos suelen tener un comportamiento similar en la mayoría de los casos. En este sentido, únicamente FuzzyElBaf y ElgammalKDE logran resultados notorios. La segunda columna de las Figuras 3.8 y 3.9 muestran los rendimientos medios utilizando la F-medida y la exactitud, respectivamente. A continuación vamos a analizar en qué situaciones son mejores los métodos FuzzyElBaf y ElgammalKDE.

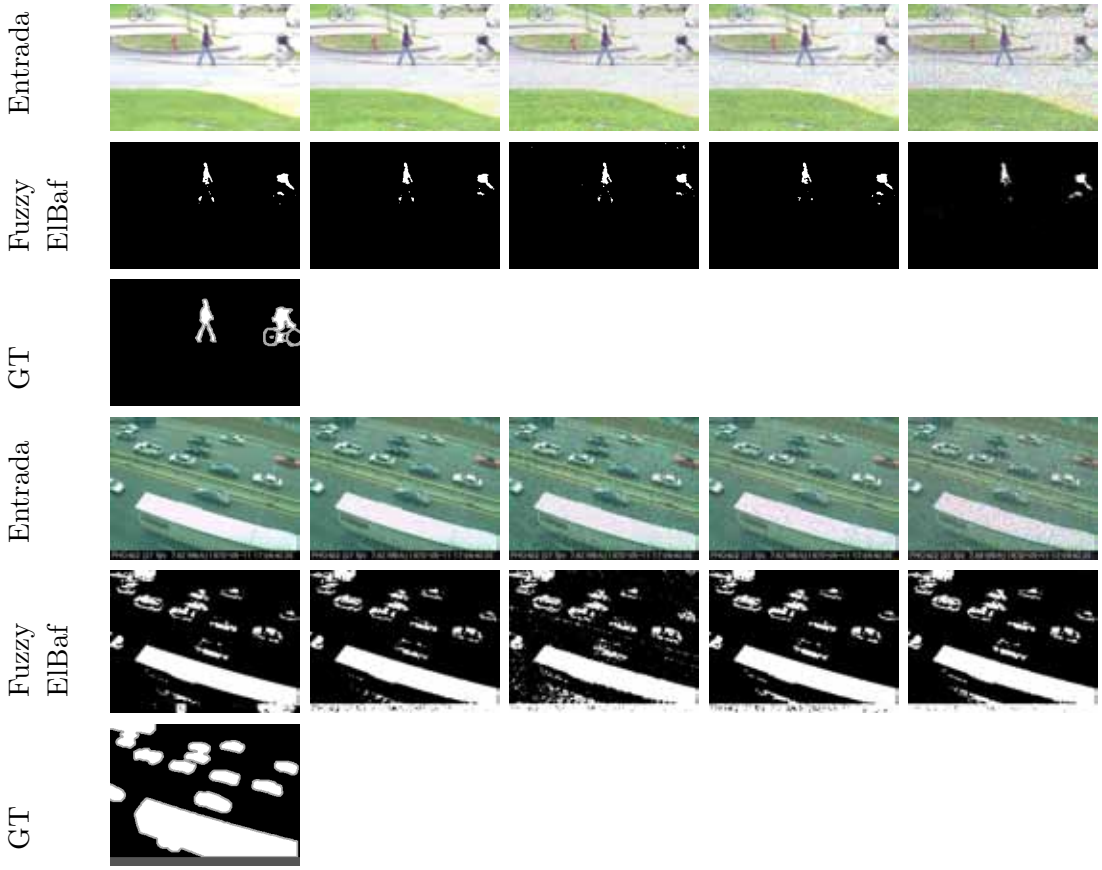



**Figura 3.16:** Salidas del método FuzzyElBaf en situaciones de ruido. La tres primeras filas corresponden al fotograma 500 de la secuencia Pedestrians, las tres últimas al fotograma 953 de Turnpike_0_5fps. La primera y cuarta filas son las entradas, la segunda y quinta las salidas, y la tercera y sexta el *ground truth*. De izquierda a derecha los niveles de ruido aplicados son: entrada sin ruido, ruido gaussiano con desviación típica de 20 y 40, ruido uniforme con probabilidades 0.10 y 0.20 y *ground truth*. Los resultados mostrados en esta figura corresponden a la configuración siguiente: espacio de color RGB utilizando las tres componentes de color; se trata de la mejor configuración en las situaciones mostradas. Como se observa, el método FuzzyElBaf es capaz de tolerar diferentes niveles de ruido tanto gaussiano como uniforme.

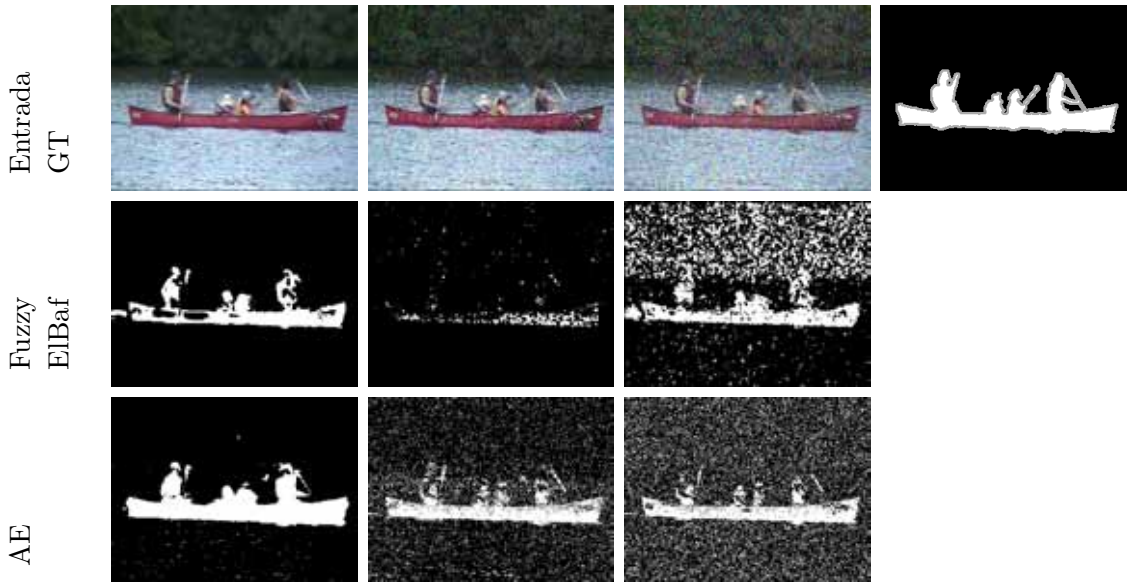



**Figura 3.17:** Salidas de los métodos FuzzyElBaf y AE en situaciones de ruido. Las imágenes corresponden al fotograma 964 de la secuencia Canoe. La segunda y tercera filas son las salidas de FuzzyElBaf y AE, respectivamente. De izquierda a derecha: entrada sin ruido, ruido gaussiano empleando una desviación típica de 40, ruido uniforme de probabilidad 0.20 y *ground truth*. Las salidas mostradas corresponden a la mejor configuración en cada situación. Es destacable que el método FuzzyElBaf tiene problemas para segmentar las zonas oscuras de la escena cuando se añade ruido a la entrada.

Como se comentó anteriormente, ElgammalKDE parece tener dificultades con la compresión de vídeo. La tercera columna de la Figura 3.15 muestra que corromper la entrada con ruido uniforme de baja probabilidad no solventa la situación (como sí que ocurría con el ruido gaussiano) y además causa una gran cantidad de falsos positivos. Esta es la razón por la que el rendimiento no es estable cuando $0.05 \leq \varsigma \leq 0.10$ e incluso es complicado diferenciarlo del obtenido por la mayoría de las demás propuestas. Sin embargo ElgammalKDE experimenta un mejoría de sus resultados en casi todas las secuencias cuando $0.15 \leq \varsigma \leq 0.20$ (ver Figura 3.20), mejorando incluso los resultados de FuzzyElBaf en algunos casos, por ejemplo en las secuencias Highway, Office o Sofa (ver Figuras 3.4 y 3.6). Un ruido uniforme de probabilidad $\varsigma \geq 0.10$ suele ser suficiente para solventar el problema con la compresión de vídeo. No obstante se producen demasiados falsos positivos cuando $0.05 \leq \varsigma \leq 0.10$ debido a que el modelo generado es demasiado rígido como para adaptarse a un ruido tan infrecuente. Por lo tanto, al corromper la entrada con un ruido con mayor probabilidad, es decir $0.15 \leq \varsigma \leq 0.20$, ElgammalKDE genera un modelo suficientemente flexible como para producir menor falsos positivos que en el caso de $0.05 \leq \varsigma \leq 0.10$. En otras palabras, el modelo se adapta a una situación en la que los colores de los píxeles son corrompidos frecuentemente.

FuzzyElBaf es quizá el método que mejor comportamiento exhibe en presencia de ruido uniforme. En primer lugar, no hay ningún método que supere al método FuzzyElBaf en el intervalo $0.05 \leq \varsigma \leq 0.10$. Por otro lado, cuando la probabilidad de ruido uniforme está en el rango $0.15 \leq \varsigma \leq 0.20$, ElgammalKDE es un buen competidor porque supera sus dificultades con la comprensión de vídeo, pero aún así FuzzyElBaf sigue siendo mejor. Al igual que en el caso del ruido gaussiano, FuzzyElBaf tiene problemas para segmentar las zonas oscuras de la escena, situación que se muestra en la tercera columna de la Figura 3.17 (el espacio HSV es el óptimo en este caso). A pesar de ello sigue obteniendo buenos resultados comparados con el resto de métodos, lo que hace de él una de las mejores alternativas tanto si la proba-

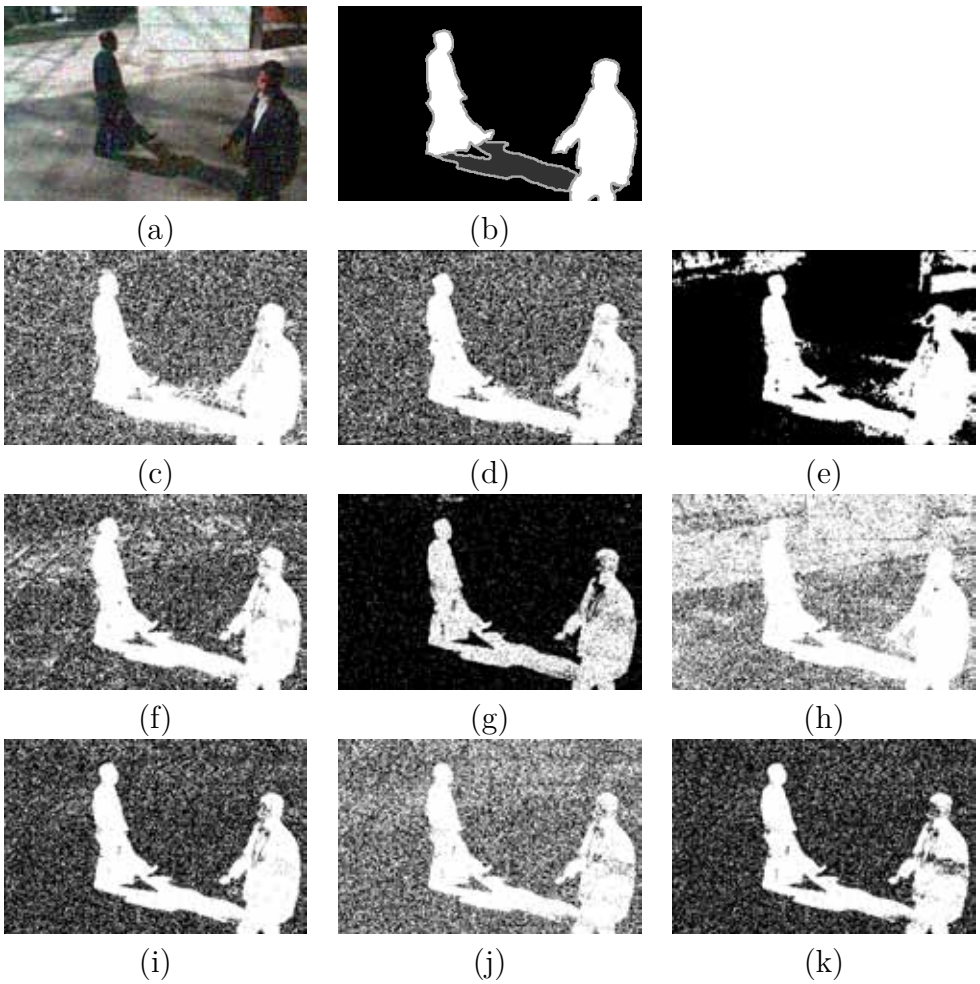



**Figura 3.18:** Salidas de los métodos en situaciones con altos niveles de ruido uniforme. Las imágenes corresponden al fotograma 310 de la secuencia peopleInShade. (a) entrada, (b) *ground truth*, (c) SOBS, (d) Sakbot, (e) FuzzyElBaf, (f) GrimsonGMM, (g) ElgammalKDE, (h) PCAOliver, (i) Pfinder, (j) ZivkovicGMM, (k) AE. La entrada ha sido corrompida con un ruido uniforme con probabilidad $\varsigma = 0.20$. En cada caso se utiliza la mejor configuración según F-medida.


**Figura 3.19:** Salidas del método AE en situaciones con ruido uniforme. Las imágenes corresponden al fotograma 310 de la secuencia peopleInShade. De izquierda a derecha: entrada sin ruido, ruido uniforme con probabilidades 0.05, 0.10 y *ground truth*. Las salidas mostradas pertenecen a la mejor configuración en cada situación. La mayoría de los métodos tienes dificultades a la hora de segmentar la escena incluso cuando el nivel de ruido uniforme es bajo.


**Figura 3.20:** Salidas del método ElgammalKDE en situaciones con ruido uniforme. Las imágenes corresponden al fotograma 1100 de la secuencia PeopleInShade. La primera fila son las entradas, la segunda las salidas del método y la tercera el *ground truth*. De izquierda a derecha los niveles de ruido aplicados son: entrada sin ruido, ruido uniforme con probabilidades 0.05, 0.10, 0.15, 0.20. Las salidas mostradas corresponden a la mejor configuración cuando $\varsigma = 0.20$. Como se puede observar, añadir una pequeña cantidad de ruido uniforme no solventa el problema de ElgammalKDE asociado a la compresión de vídeo. Es necesario usar un ruido uniforme de probabilidad $\varsigma \geq 0.15$ para que el problema se atenúe.

bilidad de ruido uniforme es baja como si es alta. En las columnas cuarta y quinta de la Figura 3.16 se puede comprobar que la salida del método es prácticamente independiente del nivel de ruido aplicado a la entrada. No hay otro método que logre tan buenos resultados. También es destacable que si comparamos las dos columnas de la Figuras 3.4-3.7, el comportamiento de FuzzyElBaf al aumentar el nivel de ruido en el caso uniforme es muy similar al que presenta ante el ruido gaussiano.

## 3.3. Discusión

En los experimentos realizados se han manifestado una serie de efectos que se producen en presencia de ruido, no todos los métodos tienen el mismo comportamiento en este tipo de situaciones. A continuación vamos a enunciar cada uno de los hechos más destacados.

El ruido distorsiona los píxeles observados y esto puede deteriorar el rendimiento de los métodos de segmentación de fondo en dos sentidos. En primer lugar, puede llevar a que el método clasifique erróneamente un píxel corrupto. En segundo lugar, puede corromper el modelo de fondo y por tanto perjudicar la segmentación de los siguientes fotogramas. El rendimiento empeora conforme el nivel de ruido uniforme aumenta, esto se debe a que los valores corrompidos por este tipo de ruido son completamente destruidos, es decir, no se conserva ninguna información acerca del valor original. Así pues, cualquier mejoría se debe atribuir a una fluctuación aleatoria. En el caso gaussiano la situación es diferente ya que se conserva alguna información acerca del valor original.

El método FuzzyElBaf es uno de los métodos más robustos frente a un alto nivel de ruido tanto gaussiano como uniforme. La explicación está en que el esquema de actualización que utiliza (ver Ecuación 2.183) protege el modelo de fondo de los entornos ruidosos gracias a que adapta la actualización al grado de disparidad global entre los píxeles de entrada y el modelo de fondo. El parámetro $\beta_t$ (ver Ecuación 2.237) controla cómo se va a actualizar el píxel. Cuando la escena es ruidosa (valor bajo de $m_1$) el píxel del modelo de fondo permanece prácticamente inalterado si la disparidad entre el píxel de entrada y del fondo es muy alta, previniendo en la mayoría de los casos que el modelo de fondo se actualice con valores corrompidos con ruido.

En la mayoría de los casos los mejores resultados de ElgammalKDE se obtienen al añadir una pequeña cantidad de ruido gaussiano. Esto puede atribuirse a la existencia de artefactos en los vídeos, como los efectos de la compresión de vídeo que varían los valores del píxel de forma abrupta e impredecible. Hay otros efectos como pequeñas variaciones no repetitivas del color (por ejemplo debido al viento o a los cambios de iluminación) que también pueden ser perjudiciales. Estas imperfecciones y efectos no pueden ser aprendidas correctamente por los modelos debido a que no se repiten lo suficiente o se repiten pero con un aspecto completamente distinto.

Bajo estas circunstancias, añadir un poco de ruido gaussiano, el cual no elimina completamente la información original, es beneficioso ya que habitúa al modelo a una mayor variabilidad de los valores de los píxeles, por tanto cuando surgen estos efectos e imperfecciones impredecibles, el modelo es más tendente a clasificarlos como fondo. En general la mejoría con poco ruido gaussiano puede verse como un tipo de regularización de matrices de covarianza, la cual es una técnica bien conocida para evitar los problemas de sobreentrenamiento [182, 166].

Otro efecto ampliamente estudiado es la resonancia estocástica. Sucede cuando una señal que no es lo suficientemente potente como para ser detectada por un sensor, puede ser intensificada añadiéndole ruido de manera que logra ser detectada por dicho sensor. Una situación análoga ocurre cuando un método de segmentación de fondo no es capaz de segmentar adecuadamente los objetos del primer plano debido a que los píxeles que los componen son muy similares a los del modelo de fondo. Los métodos GrimsonGMM y Sakbot exhiben este efecto en algunas secuencias, es decir, al procesar la entrada sin ruido sufren un número importante de falsos negativos, pero al añadir una pequeña cantidad de ruido gaussiano, se intensifican las diferencias entre los objetos del primer plano y el modelo de fondo, lo que facilita la detección. Esto reduce la cantidad de falsos negativos y redunda en un aumento del rendimiento.

## 3.4. Conclusiones

En este capítulo se presenta un estudio sobre los efectos producidos por el ruido gaussiano y uniforme en varios métodos de segmentación de fondo. Los resultados experimentales muestran que el ruido uniforme es muy perjudicial para estos métodos, produciendo una gran cantidad de falsos positivos y negativos en prácticamente todos los niveles de ruido probados. En general los métodos tienen mejor resistencia al ruido gaussiano debido a que no destruye por completo la información contenida en los píxeles originales. No obstante, niveles elevados de ruido gaussiano empeoran significativamente el rendimiento de los métodos. Sorprendentemente añadir pequeñas cantidades de ruido gaussiano a la entrada mitiga las limitaciones de algunos de los métodos estudiados en situaciones difíciles. Esto se puede atribuir tanto a efectos de regularización de matrices de covarianza como de resonancia estocástica.

# 4 Espacios de color

El espacio de color que utilizar para la segmentación de fondo es una decisión importante ya que condiciona los valores que tendrán cada una de las componentes de color de los píxeles de entrada y en su caso los del modelo de fondo. El espacio de color RGB es utilizado habitualmente para representar los colores en las imágenes digitales, sin embargo no siempre es la mejor opción de cara a la segmentación de fondo. Otros espacios de color como YCbCr o HSV han sido utilizados en la literatura para modelar el fondo, no obstante los mejores resultados han sido obtenidos utilizando diversos espacios de color en función del tipo de aplicación, escena, método de segmentación, etc.

En este capítulo se propone un proceso de selección de espacio de color y ponderación de componentes de color para detectar objetos del primer plano empleando mapas autoorganizados. En la Sección 4.1 se muestra una visión general sobre los métodos de segmentación de fondo que utilizan espacios de color distintos del RGB. La Sección 4.2 describe el modelo de fondo basado en mapas autoorganizados que se ha utilizado. La propuesta sobre selección de espacios de color y ponderación de componentes se expone en la Sección 4.3. La Sección 4.4 describe los experimentos llevados a cabo y los resultados obtenidos. Finalmente, la Sección 4.5 realiza la discusión sobre el trabajo realizado mientras que la Sección 4.6 está dedicada las conclusiones.

## 4.1. Introducción

La elección del espacio de color es esencial para el buen rendimiento de los métodos de segmentación de fondo. En [183] el espacio de color HSV se utiliza para modelar el fondo, el cual se combina con una segmentación basada en agrupación difusa para extraer los objetos de interés de una escena. Este modelo es capaz de extraer el fondo gracias a la descripción precisa que proporciona el espacio HSV. Posteriormente, la segmentación de los objetos en movimiento se utiliza para distinguir entre el primer plano y el ruido. La elección de un espacio de color adecuado también se investiga en [184], donde se determina un espacio de color híbrido constituido por tres componentes de color significativas. Esta aproximación es utilizada para la segmentación de imágenes en color de secuencias de fútbol con fondo dinámico. Otro trabajo [185] utiliza el espacio YCbCr en un algoritmo de segmentación de sombras para detectar vehículos en un sistema de monitorización de tráfico. Esta propuesta

emplea un método de segmentación de fondo basado en BDWT (*Binary Discrete Wavelet Transform*). Las BDWT se usan en conjunción con un algoritmo de detección de sombras para obtener el área de movimiento en la componente Y y después segmentar las sombras en el espacio YCbCr. En [186] se propone un modelo HC3 (Hybric Cone-Cylinder Codebook) el cual combina un modelo de fondo adaptativo con el espacio HSV para la eliminación de sombras. El problema de la segmentación de fondo cuando hay un fondo dinámico se aborda en [187], donde se usa un modelo basado en distribuciones de mixtura gaussianas para modelar el fondo. A diferencia de los espacios habituales se utiliza un espacio denominado Lab2000HL en el cual la componente de tonos es lineal.

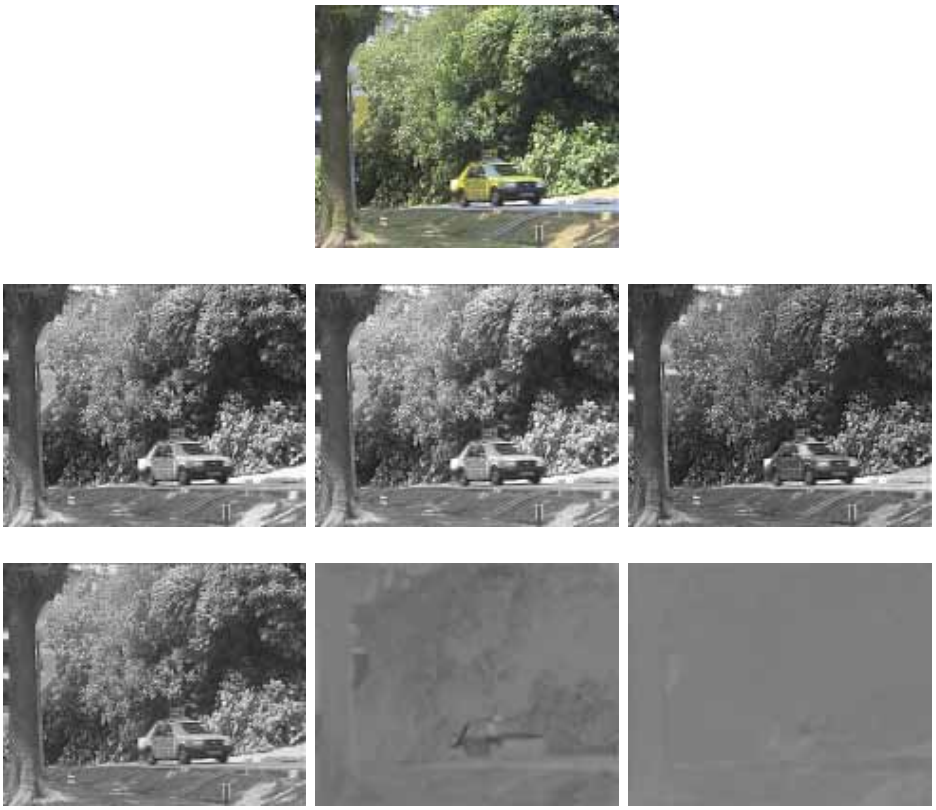


**Figura 4.1:** Ejemplo de imagen codificada con diferentes espacios de color. De arriba abajo: imagen original, canales $RGB$ y canales $Y'C_BC_R$.

En un trabajo previo [139] se propuso un modelo de fondo basado en mapas autoorganizados probabilísticos. Una limitación fundamental de esta aproximación es que considera matrices de covarianza esféricas, lo que lleva a que todas las dimensiones de entrada tengan el mismo peso. Además únicamente se consideraba el espacio de color RGB. A continuación abordaremos estas cuestiones, para ello seleccionaremos los mejores espacios de color y la mejor ponderación de las componentes de color para una escena dada.

## 4.2. Modelo

En esta sección se describe el sistema de detección del fondo que se ha utilizado como base. En primer lugar, en la Subsección 4.2.1 se define el modelo probabilístico del fondo, que se basa en un mapa autoorganizado. Posteriormente, en la Subsección 4.2.2 se explica el proceso de aprendizaje en línea basado en aproximación estocástica. Para obtener más detalles se puede consultar [139].

### 4.2.1. Definición

El modelo recibe como entrada los fotogramas del vídeo a analizar y procesa sus píxeles como muestras de entrenamiento. Esta orientado a construir una representación probabilística del fondo de la escena, que se emplea para determinar que píxeles pertenecen al primer plano en cada instante de tiempo. Como se verá más adelante, se pueden usar muchos espacios de color, pero en todos los casos la dimensión del espacio de entrada es $D = 3$, es decir, hay tres componentes de color. La distribución de probabilidad del píxel $p$ en el instante $t$ se modela mediante una mixtura con dos componentes. La función de densidad de probabilidad asociada viene dada por:

$$\mathrm{p}(\boldsymbol{p}_t) = \pi_{B,t}\mathrm{p}\left(\boldsymbol{p}_t|B\right) + \pi_{F,t}\mathrm{p}\left(\boldsymbol{p}_t|F\right) \tag{4.1}$$

como en otras ocasiones $B$ denota el fondo y $F$ el primer plano.

En términos generales se puede suponer que los objetos de primer plano pueden tener cualquier color. Al igual que se hace en la Subsección 2.5.1 para el modelado de fondo mediante aproximación estocástica, esto sugiere que para modelar el primer plano se use una distribución uniforme sobre el espacio de color:

$$\mathrm{p}\left(\boldsymbol{p}_t|F\right) = U\left(\boldsymbol{p}_t\right) \tag{4.2}$$

$$U\left(\boldsymbol{p}_t\right) = \begin{cases} 1/Vol\left(\mathcal{S}\right) & \boldsymbol{p}_t \in \mathcal{S} \\ 0 & \boldsymbol{p}_t \notin \mathcal{S} \end{cases} \tag{4.3}$$

donde $\mathcal{S}$ es el soporte de la función de densidad de probabilidad de la distribución uniforme y $Vol\left(\mathcal{S}\right)$ es volumen $D$-dimensional de $\mathcal{S}$. Esta manera de modelar el primer plano garantiza que todos los objetos entrantes se tratan de la misma manera independientemente de su color.

Dado que la distribución de los valores del color del fondo en una cierta posición $p$ depende de las características de la escena. Por ejemplo, árboles azotados por el viento y otros objetos de fondo dinámicos, se hace necesario el uso de distribuciones multimodales para modelar el fondo. Un mapa autoorganizado probabilístico puede

manejar este tipo de distribución, ya que cada neurona puede especializarse en un grupo del conjunto de datos de entrada:

$$\mathrm{p}\left(\boldsymbol{p}_t \mid B\right) = \frac{1}{H}\sum_{i=1}^{H}\mathrm{p}\left(\boldsymbol{p}_t \mid i\right) \tag{4.4}$$

donde $H$ es el número de componentes de mixtura (neuronas) del mapa autoorganizado y las probabilidades a priori se suponen iguales. A continuación es preciso definir una distancia topológica $\delta_T\left(i,j\right)$ entre cada par de neuronas $(i,j)$ del mapa. La elección habitual es la que se ha empleado aquí: una rejilla rectangular de unidades de proceso, junto con la distancia topológica euclídea:

$$\delta_T\left(i,j\right) = \left\|\boldsymbol{r}_i - \boldsymbol{r}_j\right\| \tag{4.5}$$

donde $\boldsymbol{r}_i$ y $\boldsymbol{r}_j$ son, respectivamente, las posiciones de las unidades de proceso $i$ y $j$ en la rejilla rectangular.

La carga de cálculos del modelo probabilístico debe ser tan pequeña como sea posible, dado que existe un mapa autoorganizado por cada posición $p$ del fotograma. La opción más sencilla es considerar una densidad de probabilidad gaussiana esférica para cada componente de mixtura $i \in \{1,\ldots,H\}$ del mapa [188, 189, 190]:

$$\mathrm{p}\left(\boldsymbol{p}_t \mid i\right) = \left(2\pi\right)^{-D/2}\left(\sigma_{i,t}^2\right)^{-D/2}\exp\left(-\frac{1}{2\sigma_{i,t}^2}\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right)\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right)^T\right) \tag{4.6}$$

donde $\boldsymbol{\mu}_{i,t}$ y $\sigma_{i,t}^2$ son, respectivamente, el vector de medias y la varianza de la componente de mixtura $i$:

$$\boldsymbol{\mu}_{i,t} = E\left[\boldsymbol{p}_t \mid i,t\right] \tag{4.7}$$

$$\sigma_{i,t}^2 = E\left[\frac{1}{D}\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right)^T\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right) \mid i,t\right] \tag{4.8}$$

A fin de decidir si un píxel observado pertenece al fondo, se lleva a cabo un procedimiento de clasificación bayesiano. La probabilidad de que la intensidad observada pertenezca al fondo viene dada por:

$$P_B\left(\boldsymbol{p}_t\right) = \frac{\pi_{B,t}\mathrm{p}\left(\boldsymbol{p}_t \mid B\right)}{\pi_{B,t}\mathrm{p}\left(\boldsymbol{p}_t \mid B\right) + \pi_{F,t}\mathrm{p}\left(\boldsymbol{p}_t \mid F\right)} \tag{4.9}$$

mientras que la probabilidad de pertenecer al primer plano es el suceso opuesto:

$$P_F\left(\boldsymbol{p}_t\right) = 1 - P_B\left(\boldsymbol{p}_t\right) \tag{4.10}$$

Hay muchos efectos indeseables que introducen ruido tanto en $P_B(\boldsymbol{p}_t)$ como en $P_F(\boldsymbol{p}_t)$. Por ejemplo, efectos de camuflaje (objetos de primer plano cuyo color es similar al del fondo), imperfecciones de la cámara y defectos de compresión de vídeo. A fin de aliviar este problema, se puede usar la información de los ocho vecinos de un píxel dado para reducir el ruido. Una estrategia sencilla podría ser dar el mismo peso a todos los vecinos (un filtro de paso baja), pero esto ignoraría el hecho de que muchos píxeles vecinos no están correlados. Por ejemplo, esto puede ocurrir en la orilla de un río: los píxeles fuera del agua (donde no hay ondas) son casi independientes de aquellos que están en el río (donde la corriente del agua cambia la superficie) a pesar de su proximidad. Para medir la correlación entre parejas de píxeles proponemos usar la correlación de Pearson [191] entre las variables aleatorias $P_F(\boldsymbol{p}_t)$ y $P_F(\boldsymbol{q}_t)$ correspondientes a cada par de píxeles 8-vecinos $p$ y $q$:

$$\rho_{\boldsymbol{p},\boldsymbol{q}} = \frac{\phi_{\boldsymbol{p},\boldsymbol{q}}}{\sqrt{\nu_{\boldsymbol{p}}}\sqrt{\nu_{\boldsymbol{q}}}} \tag{4.11}$$

$$\phi_{\boldsymbol{p},\boldsymbol{q}} = cov\left(P_F(\boldsymbol{p}_t), P_F(\boldsymbol{q}_t)\right) =$$

$$E\left[\left(P_F(\boldsymbol{p}_t) - E\left[P_F(\boldsymbol{p}_t)\right]\right)\left(P_F(\boldsymbol{q}_t) - E\left[P_F(\boldsymbol{q}_t)\right]\right)\right] \tag{4.12}$$

$$\nu_{\boldsymbol{p}} = var\left(P_F(\boldsymbol{p}_t)\right) = E\left[\left(P_F(\boldsymbol{p}_t) - E\left[P_F(\boldsymbol{p}_t)\right]\right)^2\right] \tag{4.13}$$

$$\nu_{\boldsymbol{q}} = var\left(P_F(\boldsymbol{q}_t)\right) = E\left[\left(P_F(\boldsymbol{q}_t) - E\left[P_F(\boldsymbol{q}_t)\right]\right)^2\right] \tag{4.14}$$

donde $E\left[P_F(\boldsymbol{p}_t)\right]$ coincide con la probabilidad a priori $\pi_{F,t}$ del píxel $p$ en el instante $t$.

La correlación de Pearson está acotada y es simétrica, es decir:

$$\rho_{\boldsymbol{p},\boldsymbol{q}} \in [-1, 1] \tag{4.15}$$

$$\rho_{\boldsymbol{p},\boldsymbol{q}} = \rho_{\boldsymbol{q},\boldsymbol{p}} \tag{4.16}$$

donde la última ecuación se puede usar para ahorrar la mitad de los cálculos.

Si dos píxeles vecinos suelen asignarse a la misma clase, es decir, ambos son de fondo o ambos son de primer plano, entonces la correlación entre ellos $\rho_{\boldsymbol{p},\boldsymbol{q}}$ es grande y positiva. Por otro lado, si ambos píxeles son independientes, entonces tenemos $\rho_{\boldsymbol{p},\boldsymbol{q}} = 0$. Una correlación negativa corresponde a un par de píxeles que habitualmente son de clases distintas. Esto es bastante infrecuente, pero hay algunos casos en los que aparecen pequeños valores negativos debidos al ruido.

El ruido en $P_F(\boldsymbol{p}_t)$ se puede reducir con la ayuda de las correlaciones $\rho_{\boldsymbol{p},\boldsymbol{q}}$, de manera que a los 8-vecinos de $p$ que tengan correlaciones positivas y altas se les da mayor importancia:

$$\widetilde{P}_F(\boldsymbol{p}_t) = trunc\left(\frac{1}{9}\sum_{q\in Neigh(p)} \rho_{\boldsymbol{p},\boldsymbol{q}} P_F(\boldsymbol{q}_t)\right) \tag{4.17}$$

donde $Neigh(p)$ contiene el píxel $p$ y sus 8-vecinos y además:

$$\rho_{\boldsymbol{p},\boldsymbol{p}} = 1 \tag{4.18}$$

$$trunc(x) = \begin{cases} x & x \geq 0 \\ 0 & en\, otro\, caso \end{cases} \tag{4.19}$$

$$\widetilde{P}_F(\boldsymbol{p}_t) \in [0,1] \tag{4.20}$$

El cometido de la función *trunc* en la ecuación 4.17 consiste en desactivar el aprendizaje de los parámetros estableciendo $\widetilde{P}_F(\boldsymbol{p}_t) = 0$ cuando ocurre un pico de correlaciones negativas ruidosas. Debe señalarse que el resultado final de este proceso es post-procesado rellenando huecos de tamaño igual a un píxel; tras esto los objetos con menos de 10 píxeles de tamaño son eliminados.

### 4.2.2. Aprendizaje

El proceso de actualización de los parámetros se obtiene por aplicación del algoritmo de aproximación estocástica de Robbins-Monro. Una de sus principales ventajas es que la complejidad computacional asociada $O(HD)$ es bastante baja, lo cual es de la máxima importancia para modelos del fondo basados en mapas autoorganizados [138]. Recordemos que $\widetilde{P}_F(\boldsymbol{p}_t)$ es una estimación mejorada (con menos ruido) de la probabilidad a posteriori de que la muestra de entrada $\boldsymbol{p}_t$ en la posición $p$ y en el instante de tiempo $t$ pertenezca al primer plano:

$$P(F \mid \boldsymbol{p}_t) = \widetilde{P}_F(\boldsymbol{p}_t) \tag{4.21}$$

$$P(B \mid \boldsymbol{p}_t) = \widetilde{P}_B(\boldsymbol{p}_t) = 1 - P(F \mid \boldsymbol{p}_t) \tag{4.22}$$

El sistema aprende en línea, es decir, en el instante de tiempo $t$ un nuevo fotograma de entrada es capturado y suministrado al sistema de manera que los parámetros del modelo para cada posición de píxel $p$ se modifican de acuerdo con el color $\boldsymbol{p}_t$.

La aplicación del algoritmo de Robbins-Monro lleva a las siguientes ecuaciones de actualización:

$$\pi_{F,t} = (1 - \epsilon(t))\,\pi_{F,t-1} + \epsilon(t)\,\widetilde{P}_F(\boldsymbol{p}_t) \tag{4.23}$$

$$\pi_{B,t} = 1 - \pi_{F,t} \tag{4.24}$$

$$\nu_{\boldsymbol{p}}(t) = (1 - \epsilon(t))\,\nu_{\boldsymbol{p}}(t-1) + \epsilon(t)\left(\widetilde{P}_F(\boldsymbol{p}_t) - \pi_{F,t}\right)^2 \tag{4.25}$$

$$\phi_{\boldsymbol{p},\boldsymbol{q}}(t) = (1 - \epsilon(t))\,\phi_{\boldsymbol{p},\boldsymbol{q}}(t-1) + \epsilon(t)\left(\widetilde{P}_F(\boldsymbol{p}_t) - \pi_{F,t}\right) \tag{4.26}$$

$$\rho_{\boldsymbol{p},\boldsymbol{q}}(t) = (1 - \epsilon(t))\,\rho_{\boldsymbol{p},\boldsymbol{q}}(t-1) + \epsilon(t)\,\frac{\phi_{\boldsymbol{p},\boldsymbol{q}}(t)}{\sqrt{\nu_{\boldsymbol{p}}(t)}\sqrt{\nu_{\boldsymbol{q}}(t)}} \tag{4.27}$$

$$\boldsymbol{\mu}_{i,t} = (1 - \xi(i,t))\,\mu_{i,t-1} + \xi(i,t)\,\boldsymbol{p}_t \tag{4.28}$$

$$\sigma^2_{i,t} = (1 - \xi(i,t))\,\sigma^2_{i,t-1} + \xi(i,t)\,\frac{1}{D}\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right)^T\left(\boldsymbol{p}_t - \boldsymbol{\mu}_{i,t}\right) \tag{4.29}$$

$$\xi(i,t) = \epsilon(t)\,\frac{\Lambda(i, Winner(p,t))}{\pi_{B,t}}\widetilde{P}_B(\boldsymbol{p}_t) \tag{4.30}$$

donde $Winner(p,t)$ es una función que determina la neurona ganadora y $\Lambda$ es una función de vecindad gaussiana, no confundir esta última con la función de densidad de probabilidad $\mathrm{p}(\boldsymbol{p}_t \mid i)$ de las componentes de mixtura. La función de vecindad varía con el instante de tiempo $t$ de acuerdo a un radio de vecindad decreciente $\Delta(t)$ y la distancia topológica $\delta_T(i,j)$ a la ganadora:

$$\Lambda(i, Winner(p,t)) = \exp\left(-\left(\frac{\delta_T(i, Winner(p,t))}{\Delta(t)}\right)^2\right) \tag{4.31}$$

$$\Delta(t+1) \leq \Delta(t) \tag{4.32}$$

Debe resaltarse que se usan $\widetilde{P}_F(\boldsymbol{p}_t)$ y $\widetilde{P}_B(\boldsymbol{p}_t)$ en lugar de $P_F(\boldsymbol{p}_t)$ y $P_B(\boldsymbol{p}_t)$ en las ecuaciones anteriores porque los dos primeros valores son versiones con menos ruido que los dos últimos. De este modo las probabilidades de fondo y primer plano de los píxeles vecinos son realimentadas al proceso de aprendizaje. El efecto global es que los píxeles que están en la misma región de la imagen cooperan entre sí, produciéndose resultados más coherentes.

Como en cualquier otra aplicación, el método de aprendizaje de mixturas probabilísticas que acabamos de presentar puede caer en una solución subóptima, es decir, una que no representa fielmente a la distribución de entrada. A fin de manejar este problema definimos dos umbrales: $\sigma^2_{min}$y $\sigma^2_{max}$. Y exigimos que toda unidad $i$ satisfaga la siguiente condición:

$$\sigma^2_{min} \leq \sigma^2_{i,t} \leq \sigma^2_{max} \tag{4.33}$$

Esto se consigue forzando el valor de $\sigma^2_{i,t}$ a $\sigma^2_{max}$ o $\sigma^2_{min}$ cada vez que la ecuación de actualización 4.29 produce un valor que no satisface la inecuación anterior.

## 4.3. Ponderación de las componentes de color

Hasta el momento no se ha especificado la naturaleza de las entradas tridimensionales $\boldsymbol{p}_t$. En esta sección se estudia el papel del espacio de color y del peso de las componentes de color, de tal manera que se consiga una elección adecuada de la información de entrada al modelo del fondo.

En la mayoría de videocámaras comerciales, los píxeles vienen dados en el espacio de color RGB con valores de 8 bits de precisión. Si usáramos la información de color sin procesar de este tipo de dispositivos, entonces tendríamos:

$$\mathcal{S}_{RGB} = \{(p_R, p_G, p_B) \mid p_R, p_G, p_B \in [0, 255]\} \tag{4.34}$$

$$Vol\left(\mathcal{S}_{RGB}\right) = 255^3 \tag{4.35}$$

Sin embargo, se sabe que el espacio de color RGB no refleja las verdaderas similitudes entre colores (ver Sección 2.1). Además, dependiendo de la escena, un color podría tener más información que los demás, por lo que deberíamos darle mayor importancia.

Consideremos un espacio de color genérico $\mathcal{S}$, con colores $\boldsymbol{p} = (p_1, p_2, p_3) \in \mathcal{S}$. La diferencia topológica entre dos colores $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{S}$, la cual se usa en la ecuación 4.30 para adaptar el modelo probabilístico, es la distancia euclídea al cuadrado en $\mathcal{S}$:

$$\delta_T\left(\boldsymbol{p}, \boldsymbol{q}\right) = \left(\delta^e\left(\boldsymbol{p}, \boldsymbol{q}\right)\right)^2 = \left\|\boldsymbol{p} - \boldsymbol{q}\right\|^2 \tag{4.36}$$

Bajo esta configuración básica, las tres componentes de color tienen la misma importancia para el cálculo de la distancia. Ahora bien, es posible que una ponderación diferente de las componentes logre una segmentación más ajustada a la escena real:

$$\boldsymbol{p}' = (\alpha p_1, \beta p_2, \gamma p_3) \tag{4.37}$$

$$\delta_T(\boldsymbol{p}', \boldsymbol{q}') = \alpha^2 (p_1 - q_1)^2 + \beta^2 (p_2 - q_2)^2 + \gamma^2 (p_3 - q_3)^2 \tag{4.38}$$

donde los factores de escala son no negativos:

$$\alpha, \beta, \gamma \geq 0 \tag{4.39}$$

A continuación, debemos tener en cuenta que las gaussianas de covarianzas esféricas son equivariantes con respecto a los escalados homogéneos, es decir, si todas las dimensiones se escalan con el mismo factor, entonces los parámetros del modelo aprendido también se escalan [121]. Para ver las consecuencias de este hecho, vamos a denotar $\mathcal{S}'$ al espacio de color transformado después del escalado homogéneo con un un factor de escalado común $\lambda$. Dicho escalado hace que el volumen del espacio de color transformado sea:

$$Vol(\mathcal{S}') = \lambda^3 Vol(\mathcal{S}) \tag{4.40}$$

lo que afectaría a la distribución uniforme utilizada para modelar el primer plano:

$$U(\boldsymbol{p}'_t) = \lambda^{-3} U(\boldsymbol{p}_t) \tag{4.41}$$

Si comparamos el modelo de fondo original con el transformado después del escalado homogéneo obtenemos que el vector de medias y de varianzas sufren las siguientes modificaciones:

$$\boldsymbol{\mu}'_{i,t} = \lambda \boldsymbol{\mu}_{i,t} \tag{4.42}$$

$$\left(\sigma^2_{i,t}\right)' = \lambda^2 \sigma^2_{i,t} \tag{4.43}$$

Por lo tanto las componentes transformadas de la distribución de mixtura son:

$$\begin{aligned} \mathrm{p}(\boldsymbol{p}_t \mid i) &= (2\pi)^{-3/2} \left(\left(\sigma^2_{i,t}\right)'\right)^{-3/2} \exp\left(-\frac{1}{2\left(\sigma^2_{i,t}\right)'} \left(\boldsymbol{p}'_t - \boldsymbol{\mu}'_{i,t}\right)\left(\boldsymbol{p}'_t - \boldsymbol{\mu}'_{i,t}\right)^T\right) \\ &= (2\pi)^{-3/2} \lambda^{-3} \left(\sigma^2_{i,t}\right)^{-3/2} \exp\left(-\frac{1}{2\sigma^2_{i,t}} \left(\boldsymbol{p}'_t - \boldsymbol{\mu}_{i,t}\right)\left(\boldsymbol{p}'_t - \boldsymbol{\mu}_{i,t}\right)^T\right) \\ &= \lambda^{-3} \mathrm{p}(\boldsymbol{p}'_t \mid i) \end{aligned} \tag{4.44}$$

De las ecuaciones 4.1,4.9, 4.41 y 4.44 se obtiene que:

$$P_B\left(\boldsymbol{p}'_t\right) = P_B\left(\boldsymbol{p}_t\right) \tag{4.45}$$

La ecuación anterior significa que las probabilidades de clasificación de los píxeles no se ven afectadas por los escalados homogéneos, los cuales multiplican todas las dimensiones de color por el mismo factor. Esto implica que únicamente hay dos grados de libertad a la hora de elegir los factores de escala, por tanto normalizaremos las componentes transformadas utilizando $\lambda = \alpha + \beta + \gamma$ tal que:

$$\alpha + \beta + \gamma = 1 \tag{4.46}$$

Considerando las ecuaciones 4.39 y 4.46 se obtiene que:

$$\alpha, \beta, \gamma \in [0, 1] \tag{4.47}$$

$$\gamma = 1 - \alpha - \beta \tag{4.48}$$

Por lo tanto se puede llevar a cabo un proceso de optimización sobre $\alpha$ y $\beta$ de acuerdo con estas dos últimas expresiones, a fin de elegir los mejores factores de escala para la tarea del modelado de fondo.

## 4.4. Resultados experimentales

En esta sección se explican las pruebas llevadas a cabo para justificar las conclusiones a las que se ha llegado. En primer lugar, las Subsecciones 4.4.1 y 4.4.2 describen los espacios de color y las secuencias utilizadas. La Subsección 4.4.3 está dedicada a mostrar los parámetros seleccionados y por último, las Subsecciones 4.4.4 y 4.4.5 exponen, respectivamente, los resultados cualitativos y cuantitativos.

### 4.4.1. Espacios de color

Se han elegido cinco espacios de color diferentes además del espacio RGB, a saber: los espacios CIELAB y CIELUV a los que denominamos Lab y Luv, respectivamente; los espacios cilíndricos HSV y HSL formados por las componentes matiz, saturación y brillo; y el espacio YCbCr, que incluye una componente lumínica y dos cromáticas. Las características de todos estos espacios de color así como las transformaciones de unos a otros son descritos en detalle en la Sección 2.1.

### 4.4.2. Secuencias

Con el objetivo de realizar las evaluaciones de la forma más justa posible, se ha utilizado un conjunto de secuencias que representan situaciones reales tanto de escenas interiores como exteriores. Por ello, utilizamos un total de 11 secuencias que han sido empleadas en otros estudios y además son accesibles de forma pública. En algunos casos ha sido necesario segmentar las secuencias manualmente para llevar a cabo el estudio cuantitativo. El Cuadro 4.1 es un resumen de las secuencias empleadas. A continuación vamos a describir brevemente las secuencias consideradas.

| Secuencias | Fuente | Descripción |
|---|---|---|
| Video2 y Video4 | VSSN 2006 | Objetos 3D insertados de forma artificial en una secuencia real para generar fácilmente el ground truth. |
| Campus, Meeting room, Subway station, Water surface, Lobby y Fountain | Institute for Infocomm Research | Situaciones complejas donde los métodos suelen sufrir los efectos del camuflaje, sombras proyectadas o cambios de iluminación. |
| One shop one wait 1cor | CAVIAR dataset | Representa una galería bulliciosa. |
| Level crossing | Sheikh & Shah[192] | Presenta texturas dinámicas. |
| Light switch | Laboratory for Image and Media Understanding | Cambios de iluminación en una secuencia de interior. |

**Cuadro 4.1:** Listado de secuencias empleadas en los experimentos.

| Espacio de color | $l$ | $\epsilon_0$ | $H$ |
|---|---|---|---|
| RGB | 0.05 | 0.05 | 18 |
| Lab | 0.001 | 0.01 | 18 |
| Luv | 0.005 | 0.001 | 6 |
| HSV | 0.01 | 0.05 | 6 |
| YCbCr | 0.005 | 0.01 | 12 |

**Cuadro 4.2:** Conjunto de parámetros utilizados por el modelo SOM en cada espacio de color.

Video2 y Video4 fueron diseñadas para la conferencia internacional VSSN'06[1] utilizando objetos 3D artificiales en movimiento sobre una secuencia con fondo real. La principal ventaja es que se proporciona la segmentación perfecta entre primer plano y fondo para todos los fotogramas gracias a que la propia plataforma generaba la secuencia artificial a la vez que el *ground truth*.

Hemos considerado también un conjunto de secuencias utilizadas en el trabajo de [121] y disponibles en su página web[2] denominadas: Campus, Meeting Room, Subway

[1] http://mmc36.informatik.uni-augsburg.de/VSSN06-OSAC
[2] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 0 |
| $\beta$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 |
| $\gamma$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.9 |
| $\alpha$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0 | 0.1 | 0.2 |
| $\beta$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 |
| $\gamma$ | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.8 | 0.7 | 0.6 |
| $\alpha$ | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| $\beta$ | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| $\gamma$ | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 |
| $\alpha$ | 0.6 | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0 | 0.1 | 0.2 | 0.3 |
| $\beta$ | 0.3 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.5 | 0.5 | 0.5 | 0.5 |
| $\gamma$ | 0.1 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.5 | 0.4 | 0.3 | 0.2 |
| $\alpha$ | 0.4 | 0 | 0.1 | 0.2 | 0.3 | 0 | 0.1 | 0.2 | 0 | 0.1 | 0 |
| $\beta$ | 0.5 | 0.6 | 0.6 | 0.6 | 0.6 | 0.7 | 0.7 | 0.7 | 0.8 | 0.8 | 0.9 |
| $\gamma$ | 0.1 | 0.4 | 0.3 | 0.2 | 0.1 | 0.3 | 0.2 | 0.1 | 0.2 | 0.1 | 0.1 |

**Cuadro 4.3:** Configuraciones usadas en los experimentos para ponderar cada componente de color.

Station, Water Surface, Fountain y Lobby. En estas escenas aparecen los típicos artefactos y elementos que complican la segmentación, como son el camuflaje (misma tonalidad del primer plano y el fondo), sombras de los objetos de primer plano proyectadas en el fondo o cambios de iluminación.

Se ha seleccionado un vídeo del repositorio del proyecto CAVIAR[3] denominado Corridor (también conocido como OneShopOneWait1cor[4]), que presenta un pasillo transitado por personas con ciertas dificultades de segmentación.

Por último, las secuencias LevelCrossing [192] y LightSwitch, muestran escenas exteriores en el primer caso y situaciones con cambios de iluminación abruptos en el segundo.

### 4.4.3. Selección de parámetros

En primer lugar es necesario ajustar el modelo SOM para cada espacio de color, es decir, determinar la tasa de aprendizaje $l$, el tamaño de paso $\epsilon_0$ y el número de neuronas en la capa oculta $H$. Para ello se han realizado pruebas para determinar la configuración óptima de cada espacio de color atendiendo a la F-medida.

Los rangos de valores considerados para los parámetros del modelo SOM son los siguientes: $l \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$, $\epsilon_0 \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$, $H \in \{3, 6, 12, 18\}$. Conforme mayor sea la tasa de aprendizaje $l$ y el tamaño del paso $\epsilon_0$,

[3]http://homepages.inf.ed.ac.uk/rbf/CAVIAR/
[4]http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

más rápido se adaptará el modelo SOM a las variaciones de color de los píxeles de la escena. Sin embargo, una adaptación demasiado rápida puede llevar a una detección deficiente si las variaciones en el color se deben a objetos del primer plano. Por otro lado, el número de neuronas $H$ debe ser suficientemente grande para representar de forma adecuada la variabilidad del color del fondo, pero si se usan demasiadas es posible que algunas neuronas se asocien erróneamente a los colores de los objetos del primer plano.

Los resultados de las pruebas realizadas han determinado que las configuraciones óptimas del modelo SOM para cada espacio de color son las que se muestran en el Cuadro 4.2.

Para ponderar cada una de las componentes de los vídeos de entrada se han 55 configuraciones diferentes de los parámetros $\alpha$, $\beta$ y $\gamma$. Estas configuraciones se listan en el Cuadro 4.3.

### 4.4.4. Resultados cualitativos

En esta subsección se muestran y analizan de forma cualitativa la salida correspondiente a un fotograma de cada secuencia y espacio de color. En las Figuras 4.2 y 4.3 se muestran dichas salidas, cada fila a partir de la tercera se corresponde con un espacio de color concreto. La segmentación se ha llevado a cabo utilizando una ponderación uniforme, es decir, $\alpha, \beta, \gamma = 1/3$.

Como vemos en la Figura 4.2, el fotograma elegido de Campus es segmentado correctamente en la mayoría de los casos, no se trata de una secuencia especialmente compleja, la dificultad estriba principalmente en que tiene un fondo dinámico y que los objetos en movimiento pueden camuflarse. Usando los espacios de color HSL o HSV se comete una pequeña cantidad de pocos falsos positivos. YCrCb sufre los efectos de camuflaje en mayor medida que el resto de alternativas. Lab, Luv y RGB rinden de forma similar, esto es, algunos falsos negativos en el borde de los objetos y muy pocos falsos positivos.

En la secuencia Meeting Room el espacio RGB es la mejor alternativa, logrando un resultado casi perfecto. En este caso, usar HSL y HSV produce falsos positivos debido al fondo dinámico de parte de la escena. YCrCb vuelve a cometer falsos negativos a causa del camuflaje. Lab y Luv también sufren los efectos del camuflaje aunque en menor medida y es el espacio RGB el que mejor segmenta este fotograma.

La secuencia Subway Station tiene un alto grado de complejidad tanto por tener un fondo dinámico como por el camuflaje de los objetos del primer plano. Es difícil saber qué método es mejor ya que los resultados son bastante pobres, en todo caso los únicos espacios que logran buenos resultados son RGB y Lab, el resto de métodos cometen demasiados falsos positivos y negativos.

Fountain y WaterSurface son secuencias parecidas, en ambos casos el fondo es dinámico, pero son más sencillas que Subway Station ya que la diferencia entre los

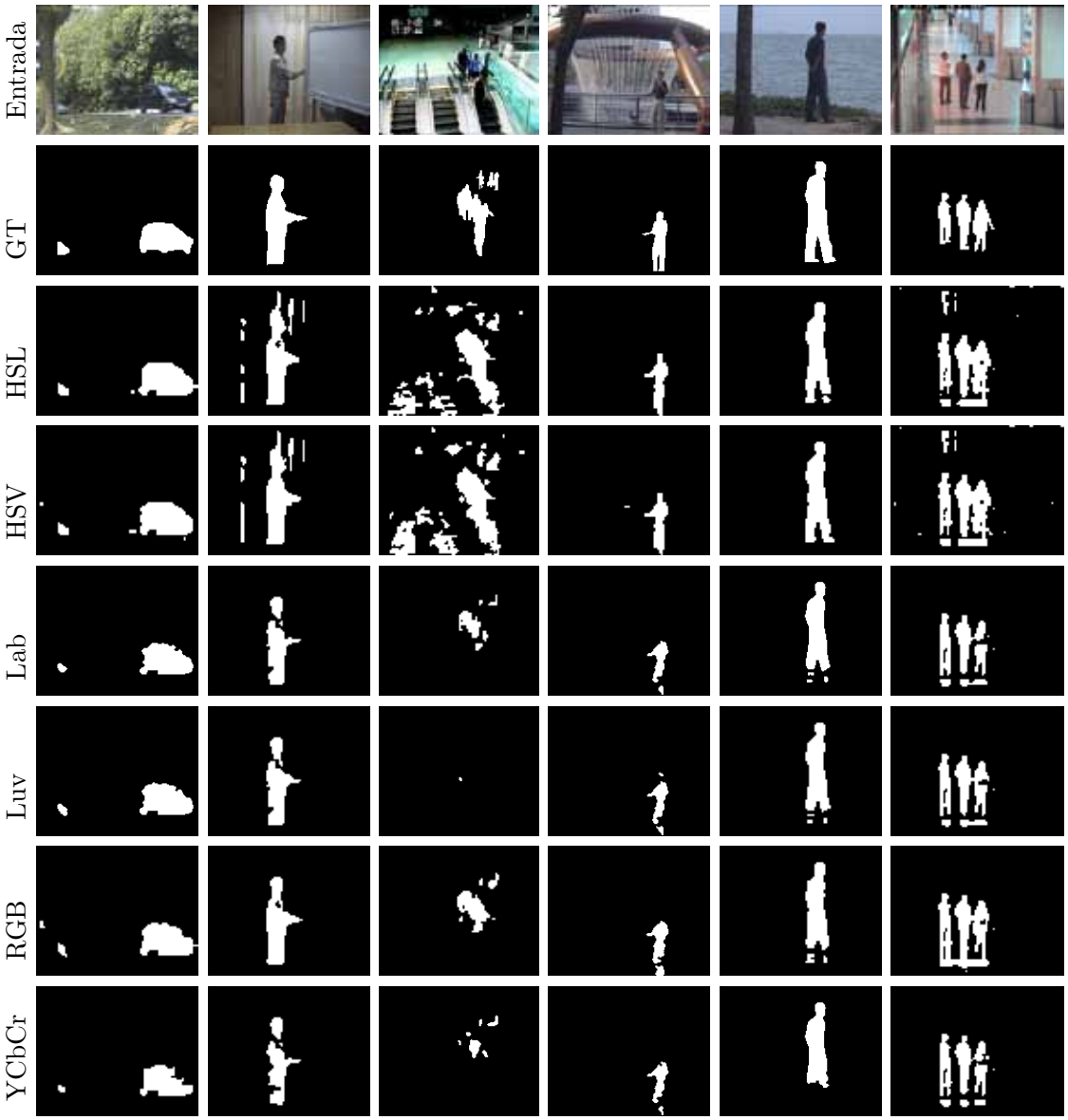



**Figura 4.2:** Ejemplos de segmentación utilizando la ponderación uniforme $\alpha, \beta, \gamma = 1/3$ . De izquierda a derecha: Campus (fotograma 2348), Meeting Room (23835), Subway Station (4787), Fountain (1489), WaterSurface (1559) y Corridor (370). De arriba a abajo: fotograma de entrada, ground truth, HSL, HSV, Lab, Luv, RGB y YCbCr.

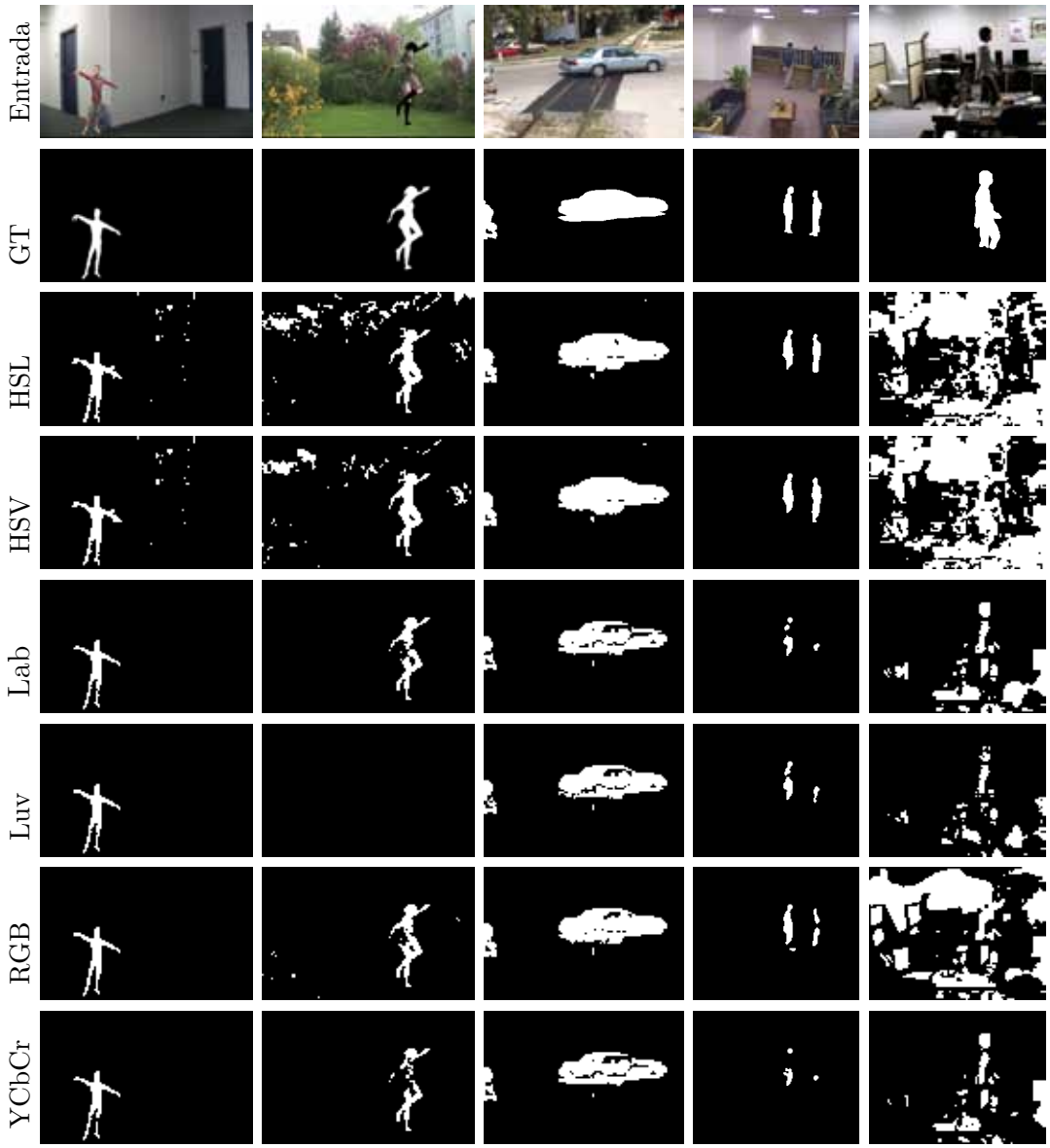



**Figura 4.3:** Ejemplos de segmentación utilizando la ponderación uniforme $\alpha, \beta, \gamma = 1/3$ . De izquierda a derecha: Video2 (fotograma 550), Video4 (690), LevelCrossing (440), Lobby (2446) y LightSwitch (1880). De arriba a abajo: fotograma de entrada, ground truth, HSL, HSV, Lab, Luv, RGB y YCbCr.

colores de los objetos del primer plano y del fondo es alta. En ambas secuencias al utilizar HSL o HSV los resultados son muy buenos, en particular en WaterSurface donde la segmentación es casi perfecta. Los demás espacios cometen algunos falsos negativos en la parte inferior de los objetos, pero aún así el resultado es bueno.

En el caso de la secuencia Corridor es destacable que todos los espacios producen un número significativo de falsos positivos debido a la sombra proyectada de los objetos del primer plano. Considerando todas las alternativas, Lab y Luv son quizá los mejores al tener un equilibrio relativamente alto entre falsos positivos y negativos. HSL y HSV sufren falsos positivos debido al movimiento de objetos distantes.

Al analizar la Figura 4.3 observamos que el fotograma seleccionado de la secuencia Video2 es segmentado de forma casi perfecta por Lab, Luv y RGB. YCbCr tiene algunos problemas de camuflaje, mientras que los espacios cilíndricos cometen algunos falsos positivos.

En Video4, HSL y HSV vuelven a tener los mismos problemas, incluso de manera más intensa, no obstante es destacable que son las únicas alternativas que no cometen falsos negativos. Por otro lado, Lab, RGB y YCbCr tienen problemas de camuflaje, mientras que Luv no detecta movimiento alguno.

El fotograma elegido de LevelCrossing no representa una dificultad para la mayoría de los espacios de color. HSL y HSV logran la mejor segmentación con apenas ningún fallo, las demás alternativas tienen algunos problemas a la hora de segmentar las ventanillas y bajos del vehículo, nuevamente se trata de un problema de camuflaje.

La secuencia Lobby y LighSwitch presentan escenas con cambios de iluminación, en la primera además los objetos del primer plano permanecen estáticos durante un número relativamente alto de fotogramas, mientras que en la segunda los cambios de iluminación son más frecuentes. En el caso de Lobby, los espacios de color Luv, Lab y YCbCr subsumen como parte del fondo algunas partes de los objetos de primer plano. RGB es algo más robusto en este sentido y comete menos falsos negativos. Sin embargo, son HSL y HSV los que mejores resultados obtienen. En cuanto a LighSwitch la situación es diferente a Lobby. HSL y HSV comenten un gran número de falsos positivos y es complicado apreciar el objeto del primer plano. RGB también sufre un gran número de falsos positivos. Lab, Luv y YCbCr tienen un rendimiento similar, siendo quizá las alternativas con mejor relación entre falsos positivos y negativos.

Por tanto, a modo de resumen podemos decir que en los fotogramas estudiados los espacios HSL y HSV son los que mejores resultados obtienen, si bien tienden a cometer más falsos positivos que el resto, esto se compensa gracias a que cometen muy pocos falsos negativos. Lab y YCrCb presentan un rendimiento similar, mientras que Luv es aparentemente la peor alternativa, especialmente en las secuencias Subway Station y Video4. El artefacto más frecuentemente observado en la mayoría de los casos es el camuflaje, no obstante cabe destacar que los espacios de color cilíndricos son prácticamente inmunes a él.

Dado que los fotogramas analizados representan una porción demasiado reducida del conjunto de fotogramas que componen las secuencias, en la subsección siguiente se realizará un estudio cuantitativo de los experimentos. De este modo podremos conocer con exactitud la influencia del espacio de color y la ponderación de los canales en el proceso de detección de fondo.

### 4.4.5. Resultados cuantitativos

Los resultados en términos de exactitud y F-medida para cada espacio de color y ponderación se muestran, respectivamente, en las Figuras 4.4 y 4.5. En estas figuras el eje X muestra el peso $\alpha$, el eje Y hace lo propio para el peso $\beta$ y el eje vertical muestra el valor medido en exactitud o en F-medida según el caso. De este modo cada punto se asocia a una ponderación de canales, representando su rendimiento medio en las 11 secuencias. Nótese que para espacio de color, los resultados medidos en exactitud y F-medida son similares, por lo tanto usaremos la medida exactitud para determinar la mejor ponderación para cada espacio de color.

Atendiendo a la forma de las superficies que componen los resultados mostrados en estas figuras podemos identificar tres grupos de espacios de color: Lab, Luv y YCbCr; HSV y HSL; y RGB. Los mejores resultados son obtenidos por los siguientes espacios de color en este orden: HSL, HSV, Lab, RGB, YCbCr y Luv. Esto confirma que en general el espacio de color RGB no es el más apropiado para la detección de fondo en secuencias de vídeo.

Para calcular la mejor ponderación para cada espacio de color se han probado todas las configuraciones de pesos para cada espacio de color en las 11 secuencias. Las mejores ponderaciones para cada espacio de color y secuencia se muestran en la Figura 4.4 junto con los resultados obtenidos en cada caso. El mejor resultado para cada secuencia se muestran en negrita. En esta figura se usa la medida exactitud como criterio para elegir la ponderación óptima. En general el mejor espacio de color depende en gran medida de la secuencia analizada. Los modelos cilíndricos (HSL y HSV) son la mejor alternativa para las secuencias con un fondo dinámico, como por ejemplo Campus y Video4, donde el movimiento de la vegetación debe ser modelado como parte del fondo o en Fountain, donde el agua que proyecta la fuente no debe ser considerada como parte del primer plano. Por otro lado, es destacable que los espacios de color HSL y HSV logran resultados particularmente mejores que las otras alternativas en la secuencia LightSwitch, probablemente se deba a que la forma cilíndrica de estos modelos ayuda a hacer frente a la estructura o variación del color. Tanto Lab como Luv también realizan buenas segmentaciones, especialmente en secuencias con baja variabilidad del fondo, como son las secuencias Video2, LevelCrossing o WaterSurface.

El Cuadro 4.5 nos permite analizar los resultados al emplear una ponderación uniforme donde todas las componentes del espacio de color tienen el mismo peso, es decir, $\alpha, \beta, \gamma = 1/3$. Como se observa, tanto F-medida como exactitud empeoran

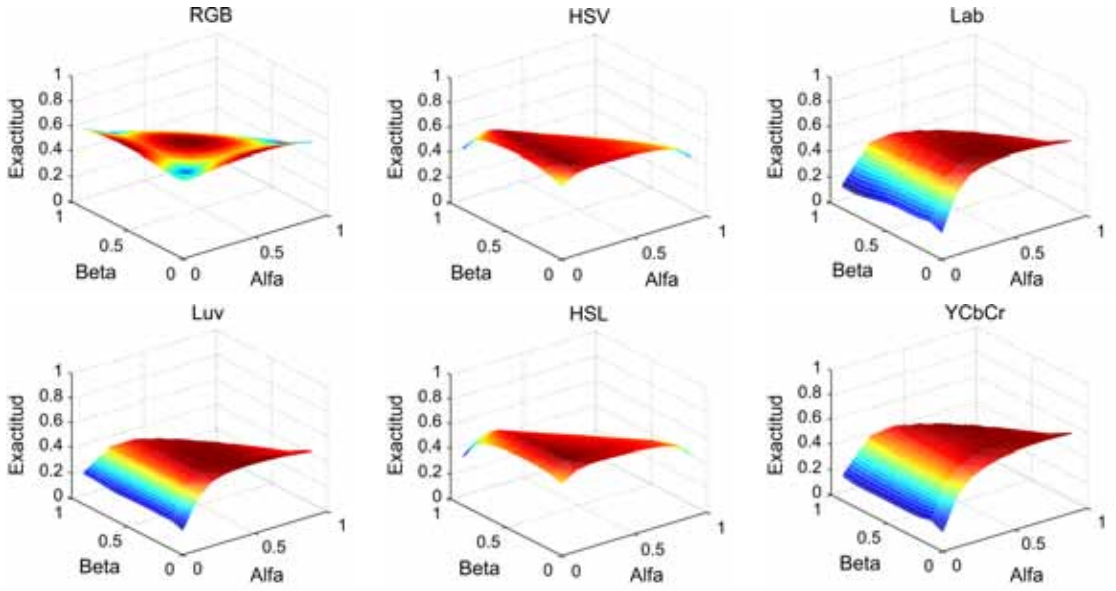



**Figura 4.4:** Rendimiento en términos de exactitud para cada espacio de color utilizando diferentes ponderaciones de los canales. El eje X e Y representan $\alpha$ y $\beta$, respectivamente. Nótese que $\gamma = 1 - \alpha - \beta$.

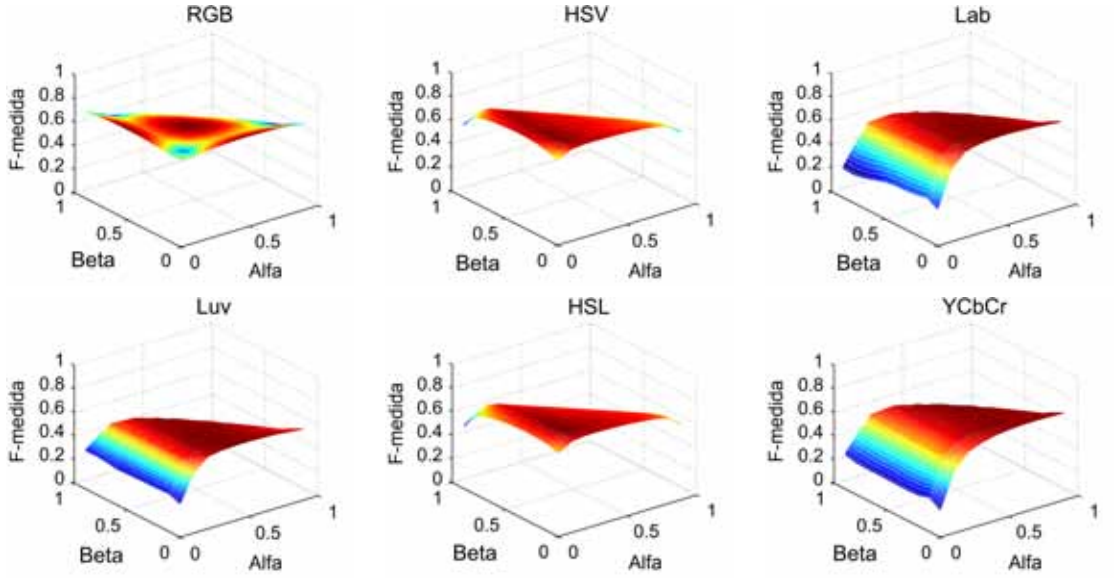



**Figura 4.5:** Rendimiento en términos de F-medida para cada espacio de color utilizando diferentes ponderaciones de los canales. El eje X e Y representan $\alpha$ y $\beta$, respectivamente.

| Secuencia | | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|---|
| Campus | Exactitud | **0.77±0.09** | **0.77±0.09** | 0.74±0.07 | 0.75±0.07 | 0.64±0.13 | 0.72±0.07 |
| | F-medida | **0.87±0.06** | 0.86±0.06 | 0.85±0.05 | 0.85±0.05 | 0.78±0.10 | 0.84±0.05 |
| | Ponderación | (0.2,0.2,0.6) | (0.2,0.1,0.7) | (0.2,0.3,0.5) | (0.1,0.4,0.5) | (0.4,0.0,0.6) | (0.2,0.7,0.1) |
| Meeting Room | Exactitud | 0.75±0.07 | 0.75±0.08 | 0.80±0.06 | **0.81±0.07** | **0.81±0.07** | 0.80±0.08 |
| | F-medida | 0.86±0.04 | 0.86±0.05 | 0.89±0.04 | **0.89±0.05** | **0.89±0.05** | 0.88±0.05 |
| | Ponderación | (0.1,0.1,0.8) | (0.2,0.1,0.7) | (0.4,0.1,0.5) | (0.4,0.1,0.5) | (0.6,0.0,0.4) | (0.5,0.1,0.4) |
| Subway station | Exactitud | 0.61±0.10 | 0.55±0.12 | 0.56±0.14 | 0.03±0.06 | **0.62±0.11** | 0.53±0.15 |
| | F-medida | 0.75±0.09 | 0.70±0.11 | 0.71±0.13 | 0.04±0.10 | **0.76±0.10** | 0.68±0.14 |
| | Ponderación | (0.1,0.1,0.8) | (0.0,0.3,0.7) | (0.4,0.5,0.1) | (0.5,0.4,0.1) | (0.2,0.5,0.3) | (0.7,0.1,0.2) |
| Fountain | Exactitud | **0.76±0.06** | **0.76±0.06** | 0.66±0.09 | 0.67±0.07 | 0.65±0.05 | 0.66±0.09 |
| | F-medida | **0.86±0.04** | **0.86±0.04** | 0.79±0.07 | 0.80±0.05 | 0.79±0.04 | 0.79±0.07 |
| | Ponderación | (0.5,0.1,0.4) | (0.4,0.2,0.4) | (0.5,0.0,0.5) | (0.3,0.6,0.1) | (0.5,0.0,0.5) | (0.6,0.0,0.4) |
| Level crossing | Exactitud | 0.89±0.04 | 0.89±0.04 | **0.91±0.03** | 0.89±0.03 | **0.91±0.03** | 0.90±0.03 |
| | F-medida | 0.94±0.02 | 0.94±0.02 | **0.95±0.02** | 0.94±0.02 | **0.95±0.02** | 0.95±0.02 |
| | Ponderación | (0.1,0.4,0.5) | (0.1,0.3,0.6) | (0.4,0.1,0.5) | (0.4,0.1,0.5) | (0.4,0.3,0.3) | (0.4,0.5,0.1) |
| Corridor | Exactitud | 0.77±0.03 | 0.80±0.05 | 0.82±0.04 | **0.84±0.04** | 0.63±0.02 | **0.84±0.04** |
| | F-medida | 0.87±0.02 | 0.89±0.03 | 0.90±0.03 | **0.92±0.03** | 0.77±0.02 | 0.91±0.03 |
| | Ponderación | (0.0,0.6,0.4) | (0.1,0.8,0.1) | (0.1,0.4,0.5) | (0.1,0.6,0.3) | (0.5,0.0,0.5) | (0.1,0.1,0.8) |
| Video2 | Exactitud | 0.89±0.05 | 0.90±0.03 | **0.94±0.02** | **0.94±0.02** | 0.93±0.02 | 0.94±0.03 |
| | F-medida | 0.94±0.03 | 0.94±0.02 | **0.97±0.01** | **0.97±0.01** | 0.96±0.01 | 0.97±0.01 |
| | Ponderación | (0.0,0.2,0.8) | (0.0,0.2,0.8) | (0.8,0.1,0.1) | (0.8,0.1,0.1) | (0.2,0.7,0.1) | (0.8,0.1,0.1) |
| Video4 | Exactitud | **0.79±0.05** | 0.79±0.10 | 0.76±0.11 | 0.00±0.01 | 0.75±0.07 | 0.73±0.11 |
| | F-medida | **0.88±0.03** | 0.88±0.07 | 0.86±0.08 | 0.01±0.01 | 0.85±0.05 | 0.84±0.07 |
| | Ponderación | (0.2,0.0,0.8) | (0.2,0.1,0.7) | (0.4,0.4,0.2) | (0.1,0.5,0.4) | (0.0,0.0,1.0) | (0.6,0.3,0.1) |
| Water surface | Exactitud | 0.89±0.03 | 0.91±0.03 | **0.91±0.02** | **0.91±0.02** | 0.90±0.02 | 0.90±0.02 |
| | F-medida | 0.94±0.02 | 0.95±0.02 | **0.95±0.01** | **0.95±0.01** | 0.95±0.01 | 0.95±0.01 |
| | Ponderación | (0.2,0.5,0.3) | (0.1,0.6,0.3) | (0.4,0.1,0.5) | (0.4,0.1,0.5) | (0.1,0.7,0.2) | (0.8,0.1,0.1) |
| Lobby | Exactitud | 0.50±0.15 | 0.54±0.16 | 0.61±0.26 | 0.62±0.27 | **0.64±0.27** | 0.56±0.24 |
| | F-medida | 0.65±0.13 | 0.69±0.14 | 0.71±0.29 | 0.72±0.30 | **0.73±0.30** | 0.67±0.28 |
| | Ponderación | (0.4,0.1,0.5) | (0.1,0.5,0.4) | (0.5,0.1,0.4) | (0.5,0.1,0.4) | (0.4,0.1,0.5) | (0.8,0.1,0.1) |
| Light switch | Exactitud | 0.51±0.17 | **0.55±0.16** | 0.27±0.17 | 0.19±0.12 | 0.12±0.21 | 0.23±0.13 |
| | F-medida | 0.66±0.18 | **0.70±0.17** | 0.39±0.22 | 0.31±0.16 | 0.18±0.23 | 0.35±0.17 |
| | Ponderación | (0.0,0.3,0.7) | (0.0,0.2,0.8) | (0.1,0.1,0.8) | (0.1,0.2,0.7) | (0.5,0.0,0.5) | (0.1,0.8,0.1) |

**Cuadro 4.4:** Mejores resultados en términos de exactitud y F-medida para cada espacio de color y secuencia. La tercera fila de cada secuencia muestra las configuraciones $(\alpha, \beta, \gamma)$ que obtienen los resultados de las dos filas superiores. Se han marcado en negrita los mejores resultados para cada secuencia.

cuando se utiliza una ponderación de este tipo. Por lo tanto con estos resultados la importancia de ponderar los canales queda probada para la detección de fondo.

A continuación calculamos la mejor ponderación para cada espacio de color independientemente de la secuencia, es decir, si tuviéramos que elegir una ponderación para todos los casos, cuál sería la mejor. En primer lugar, para cada ponderación y espacio de color se calcula la media de las exactitudes obtenidas en las 11 secuencias. Posteriormente, la ponderación que obtenga el valor máximo de entre las 55 medias resultantes se elige como la mejor ponderación para ese espacio de color. Dicho valor máximo se ha marcado en negrita en el Cuadro 4.6.

Este cuadro muestra una visión general de cómo de importante es cada uno de los canales que componen los espacios de color analizados de cara a la detección de fondo. Por ejemplo, la luminancia (primera componente de color) y los canales cromáticos $b^*$ y $v^*$ (tercera componente) son los más importantes en los espacios Lab y Luv. La relevancia de la tercera componente se debe al color del fondo de las

| Secuencia | | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|---|
| Campus | Exactitud | **0.76±0.10** | 0.74±0.14 | 0.71±0.11 | 0.73±0.09 | 0.62±0.16 | 0.60±0.15 |
| | F-medida | **0.86±0.07** | 0.84±0.11 | 0.82±0.08 | 0.84±0.06 | 0.76±0.13 | 0.74±0.12 |
| Meeting room | Exactitud | 0.61±0.16 | 0.68±0.14 | 0.77±0.08 | **0.79±0.07** | **0.79±0.07** | 0.76±0.08 |
| | F-medida | 0.75±0.12 | 0.80±0.10 | 0.87±0.05 | **0.88±0.05** | **0.88±0.05** | 0.86±0.05 |
| Subway station | Exactitud | 0.46±0.10 | 0.34±0.10 | 0.52±0.15 | 0.02±0.06 | **0.62±0.11** | 0.43±0.16 |
| | F-medida | 0.62±0.10 | 0.50±0.12 | 0.68±0.14 | 0.04±0.09 | **0.76±0.09** | 0.59±0.16 |
| Fountain | Exactitud | 0.73±0.07 | **0.75±0.06** | 0.64±0.10 | 0.65±0.09 | 0.61±0.07 | 0.61±0.12 |
| | F-medida | 0.84±0.05 | **0.86±0.04** | 0.78±0.08 | 0.79±0.07 | 0.76±0.05 | 0.75±0.10 |
| Level crossing | Exactitud | 0.88±0.05 | 0.88±0.05 | 0.89±0.03 | 0.89±0.03 | **0.91±0.03** | 0.88±0.03 |
| | F-medida | 0.93±0.03 | 0.94±0.03 | 0.94±0.02 | 0.94±0.02 | **0.95±0.02** | 0.94±0.02 |
| Corridor | Exactitud | **0.74±0.05** | 0.73±0.05 | 0.69±0.04 | 0.70±0.04 | 0.60±0.02 | 0.70±0.04 |
| | F-medida | **0.85±0.03** | 0.85±0.03 | 0.81±0.03 | 0.82±0.03 | 0.75±0.02 | 0.82±0.03 |
| Video2 | Exactitud | 0.80±0.07 | 0.79±0.08 | 0.92±0.03 | 0.91±0.02 | **0.92±0.02** | 0.91±0.03 |
| | F-medida | 0.89±0.05 | 0.88±0.05 | 0.96±0.01 | 0.96±0.01 | **0.96±0.01** | 0.95±0.02 |
| Video4 | Exactitud | 0.71±0.08 | 0.74±0.11 | **0.76±0.11** | 0.00±0.01 | 0.66±0.12 | 0.70±0.09 |
| | F-medida | 0.83±0.05 | 0.84±0.08 | **0.86±0.08** | 0.01±0.01 | 0.79±0.09 | 0.82±0.07 |
| Water surface | Exactitud | 0.88±0.03 | **0.90±0.02** | 0.89±0.03 | 0.90±0.03 | 0.89±0.03 | 0.85±0.04 |
| | F-medida | 0.93±0.01 | **0.95±0.01** | 0.94±0.02 | 0.94±0.02 | 0.94±0.02 | 0.92±0.02 |
| Lobby | Exactitud | 0.47±0.16 | 0.54±0.16 | 0.50±0.23 | 0.55±0.25 | **0.62±0.27** | 0.41±0.22 |
| | F-medida | 0.62±0.15 | 0.69±0.14 | 0.62±0.26 | 0.67±0.28 | **0.72±0.30** | 0.54±0.25 |
| Light switch | Exactitud | **0.32±0.15** | **0.32±0.15** | 0.12±0.20 | 0.10±0.15 | 0.12±0.21 | 0.12±0.19 |
| | F-medida | **0.46±0.18** | **0.47±0.18** | 0.18±0.22 | 0.15±0.19 | 0.18±0.23 | 0.18±0.21 |

**Cuadro 4.5:** Resultados de los métodos utilizando la ponderación uniforme $\alpha, \beta, \gamma = 1/3$ para cada espacio de color y secuencia. Se han marcado en negrita los mejores resultados para cada secuencia.

secuencias estudiadas, es decir, el color de fondo más común en estas secuencias varía entre el azul y el amarillo, lo cual se corresponde aproximadamente con lo que modela esta componente. La luminancia también es la componente más significativa en los modelos cilíndricos (HSL y HSV), donde los mejores resultados se obtienen cuando su peso es elevado. No obstante, las componentes de tono y saturación también son necesarias ya que el rendimiento merma cuando sus pesos son nulos (ver la primera configuración). Para terminar, observamos que los mejores resultados del espacio RGB se obtienen utilizando una ponderación aproximadamente uniforme, lo cual nos lleva a que todas las componentes tienen la misma importancia. Esto se debe a que la iluminación se distribuye uniformemente entre las tres componentes.

| Ponderación | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|
| (0.0,0.0,1.0) | 0.59 | 0.59 | 0.22 | 0.20 | 0.63 | 0.16 |
| (0.1,0.0,0.9) | 0.66 | 0.66 | 0.50 | 0.42 | 0.63 | 0.42 |
| (0.2,0.0,0.8) | 0.68 | 0.68 | 0.62 | 0.53 | 0.65 | 0.54 |
| (0.3,0.0,0.7) | 0.68 | 0.68 | 0.66 | 0.56 | 0.66 | 0.59 |
| (0.4,0.0,0.6) | 0.67 | 0.68 | 0.67 | 0.57 | 0.67 | 0.62 |
| (0.5,0.0,0.5) | 0.67 | 0.67 | 0.68 | 0.57 | 0.67 | 0.63 |
| (0.6,0.0,0.4) | 0.65 | 0.66 | 0.67 | 0.56 | 0.67 | 0.64 |
| (0.7,0.0,0.3) | 0.64 | 0.64 | 0.66 | 0.55 | 0.66 | 0.64 |
| (0.8,0.0,0.2) | 0.60 | 0.60 | 0.65 | 0.54 | 0.64 | 0.64 |
| (0.9,0.0,0.1) | 0.50 | 0.51 | 0.63 | 0.52 | 0.62 | 0.64 |
| (0.0,0.1,0.9) | 0.62 | 0.63 | 0.26 | 0.23 | 0.63 | 0.21 |
| (0.1,0.1,0.8) | **0.70** | **0.70** | 0.58 | 0.49 | 0.61 | 0.49 |
| (0.2,0.1,0.7) | 0.70 | 0.69 | 0.66 | 0.56 | 0.63 | 0.59 |
| (0.3,0.1,0.6) | 0.68 | 0.68 | 0.68 | 0.57 | 0.64 | 0.63 |
| (0.4,0.1,0.5) | 0.68 | 0.67 | **0.68** | 0.57 | 0.65 | 0.65 |
| (0.5,0.1,0.4) | 0.67 | 0.66 | 0.68 | 0.56 | 0.65 | 0.65 |
| (0.6,0.1,0.3) | 0.65 | 0.65 | 0.66 | 0.55 | 0.64 | 0.65 |
| (0.7,0.1,0.2) | 0.63 | 0.63 | 0.64 | 0.53 | 0.63 | 0.64 |
| (0.8,0.1,0.1) | 0.58 | 0.58 | 0.62 | 0.51 | 0.60 | 0.62 |
| (0.0,0.2,0.8) | 0.63 | 0.66 | 0.24 | 0.23 | 0.64 | 0.19 |
| (0.1,0.2,0.7) | 0.70 | 0.70 | 0.58 | 0.48 | 0.64 | 0.48 |
| (0.2,0.2,0.6) | 0.69 | 0.69 | 0.66 | 0.56 | 0.65 | 0.58 |
| (0.3,0.2,0.5) | 0.68 | 0.68 | 0.68 | **0.57** | 0.66 | 0.62 |
| (0.4,0.2,0.4) | 0.67 | 0.67 | 0.68 | 0.57 | 0.66 | 0.64 |
| (0.5,0.2,0.3) | 0.66 | 0.66 | 0.67 | 0.56 | 0.66 | 0.65 |
| (0.6,0.2,0.2) | 0.64 | 0.65 | 0.66 | 0.55 | 0.65 | 0.65 |
| (0.7,0.2,0.1) | 0.58 | 0.61 | 0.64 | 0.53 | 0.62 | 0.64 |
| (0.0,0.3,0.7) | 0.63 | 0.67 | 0.24 | 0.23 | 0.66 | 0.18 |
| (0.1,0.3,0.6) | 0.69 | 0.69 | 0.57 | 0.48 | 0.65 | 0.48 |
| (0.2,0.3,0.5) | 0.68 | 0.69 | 0.65 | 0.55 | 0.66 | 0.58 |
| (0.3,0.3,0.4) | 0.67 | 0.68 | 0.67 | 0.57 | **0.67** | 0.62 |
| (0.4,0.3,0.3) | 0.66 | 0.67 | 0.68 | 0.57 | 0.67 | 0.64 |
| (0.5,0.3,0.2) | 0.64 | 0.66 | 0.67 | 0.56 | 0.66 | 0.65 |
| (0.6,0.3,0.1) | 0.59 | 0.61 | 0.65 | 0.54 | 0.64 | 0.65 |
| (0.0,0.4,0.6) | 0.63 | 0.67 | 0.24 | 0.23 | 0.66 | 0.18 |
| (0.1,0.4,0.5) | 0.69 | 0.69 | 0.56 | 0.48 | 0.65 | 0.48 |
| (0.2,0.4,0.4) | 0.68 | 0.68 | 0.64 | 0.55 | 0.67 | 0.59 |
| (0.3,0.4,0.3) | 0.66 | 0.67 | 0.66 | 0.56 | 0.67 | 0.63 |
| (0.4,0.4,0.2) | 0.65 | 0.66 | 0.67 | 0.56 | 0.66 | 0.65 |
| (0.5,0.4,0.1) | 0.59 | 0.62 | 0.66 | 0.55 | 0.64 | 0.66 |
| (0.0,0.5,0.5) | 0.62 | 0.67 | 0.23 | 0.23 | 0.67 | 0.18 |
| (0.1,0.5,0.4) | 0.68 | 0.69 | 0.55 | 0.48 | 0.65 | 0.49 |
| (0.2,0.5,0.3) | 0.67 | 0.68 | 0.63 | 0.55 | 0.66 | 0.59 |
| (0.3,0.5,0.2) | 0.65 | 0.67 | 0.65 | 0.56 | 0.66 | 0.64 |
| (0.4,0.5,0.1) | 0.60 | 0.63 | 0.67 | 0.56 | 0.64 | **0.66** |
| (0.0,0.6,0.4) | 0.61 | 0.66 | 0.21 | 0.23 | 0.67 | 0.18 |
| (0.1,0.6,0.3) | 0.67 | 0.69 | 0.54 | 0.47 | 0.65 | 0.50 |
| (0.2,0.6,0.2) | 0.65 | 0.68 | 0.62 | 0.54 | 0.66 | 0.60 |
| (0.3,0.6,0.1) | 0.59 | 0.63 | 0.65 | 0.56 | 0.64 | 0.65 |
| (0.0,0.7,0.3) | 0.58 | 0.64 | 0.19 | 0.24 | 0.66 | 0.18 |
| (0.1,0.7,0.2) | 0.65 | 0.68 | 0.52 | 0.46 | 0.64 | 0.51 |
| (0.2,0.7,0.1) | 0.59 | 0.63 | 0.62 | 0.54 | 0.64 | 0.62 |
| (0.0,0.8,0.2) | 0.52 | 0.59 | 0.17 | 0.25 | 0.65 | 0.19 |
| (0.1,0.8,0.1) | 0.58 | 0.64 | 0.52 | 0.46 | 0.61 | 0.52 |
| (0.0,0.9,0.1) | 0.38 | 0.47 | 0.18 | 0.25 | 0.63 | 0.20 |

**Cuadro 4.6:** Exactitud media de las 11 secuencias para cada espacio de color y ponderación de canales. Se han marcado en negrita los mejores resultados para cada espacio de color.

## 4.5. Discusión

Existen tres cuestiones fundamentales que, gracias a los experimentos llevados a cabo, se han puesto de relieve. Bajo estas líneas vamos a discutir cada una de ellas.

El espacio de color RGB utilizado habitualmente en las imágenes digitales no es el más adecuado para realizar la detección de movimiento. No obstante, en el caso de que se utilice este espacio, la ponderación uniforme es una buena alternativa ya que consigue el mejor rendimiento medio de entre las diferentes ponderaciones analizadas. En general, el espacio de color óptimo depende de las características de la secuencia.

Los espacios de color cilíndricos como son HSL y HSV son lo que mejores resultados medios obtienen en las secuencias estudiadas, en ambos casos utilizando una ponderación no uniforme. Los experimentos prueban que la componente de luminancia es la más relevante. No obstante si se ignoran las componentes cromáticas, los resultados no son buenos, por lo tanto deben utilizarse aunque con un peso menor. El espacio de color YCbCr obtiene un rendimiento medio similar a RGB. Los dos espacios CIE (Lab y Luv) tienen comportamientos diferentes, Lab logra mejores resultados medios que Luv. La relevancia de la componente de brillo es alta, pero también lo es la de la tercera componente $b^*$ y $v^*$ debido a las características del color del fondo de las secuencias consideradas.

La ponderación uniforme de los canales de color que se usa por defecto no ofrece los mejores resultados en la mayoría de los casos. Salvo en RGB, todos los espacios de color muestran un mejor rendimiento medio al usar una ponderación no uniforme. La mejoría es aún mayor si consideramos cada una de las secuencias por separado, es decir, en prácticamente todos los casos se ha encontrado una ponderación que mejora los resultados de la ponderación uniforme.

## 4.6. Conclusiones

En este capítulo se ha presentado un marco común, tanto para elegir el espacio de color óptimo, como para el escalado de los componentes de distintos espacios de color para la detección de objetos de primer plano. Dicho marco ha sido aplicado a un modelo probabilístico de fondo, el cual está basado en redes neuronales autoorganizadas.

Hemos probado el funcionamiento de nuestra propuesta usando varios vídeos de prueba bien conocidos en la literatura relacionada con el procesamiento de vídeo, encontrando que las componentes de color correspondientes a la luminancia son las más relevantes para esta tarea.

Este trabajo abre la posibilidad de ajustar otros modelos de fondo al marco de trabajo propuesto con el fin de lograr un mejor funcionamiento en la detección de objetos.

# 5 Selección de rasgos

En ocasiones los algoritmos de detección del primer plano se basan en procedimientos creados para un tipo de problema o situación concreta. Esto sucede incluso cuando se utilizan modelos basados en distribuciones de mixtura, los cuales tienen potencialmente un rango de aplicación amplio. Por otro lado, es habitual descuidar el hecho de que los canales de color, o rasgos de entrada de forma más general, suelen tener una varianza diferente y que además no son independientes entre sí, lo cual empeora el rendimiento. En este capítulo se propone un modelo de fondo que no está sujeto a ninguna elección de rasgos en particular y que tiene en cuenta la variabilidad y dependencias entre los rasgos de entrada. Utiliza un marco de trabajo basado en la aproximación estocástica. Del mismo modo, también presentamos un conjunto de rasgos para los píxeles de entrada y estudiamos su idoneidad para el problema de la detección del primer plano. Finalmente, el procedimiento propuesto es comparado con otras alternativas del estado del arte, logrando resultados satisfactorios.

Este capítulo está estructurado tal y como sigue. En primer lugar, la Sección 5.1 contiene la introducción donde se realiza un repaso de algunas de las limitaciones que tienen los modelo de fondo habituales y que abordaremos en este capítulo. A continuación, en la Sección 5.2 se muestran las características propias del modelo de mixtura probabilística propuesto, haciendo hincapié en las diferencias con el modelo expuesto en la Sección 2.5 y en el cual se basa. El conjunto de rasgos para los píxeles de entrada que hemos elegido se define y estudia en la Sección 5.3. La Sección 5.4 está dedicada a describir los experimentos realizados y analizar los resultados de los mismos comparando nuestra propuesta con otros métodos actuales. Finalmente, en las Secciones 5.5 y 5.6 se realiza, respectivamente, la discusión de las características más relevantes de nuestra propuesta y las conclusiones.

## 5.1. Introducción

A la hora de diseñar un algoritmo de detección del primer plano realista se deben tomar dos decisiones fundamentales: la elección del modelo de fondo y el número y tipo de rasgos de entrada.

En la actualidad existe un gran número de algoritmos de detección del primer plano. En las Subsecciones 2.2.1.3 (modelos básicos) y 2.3.2 (modelos actuales) se exponen los más relevantes. La mayoría de las propuestas consisten en construir un modelo del fondo basado en alguna estadística obtenida de los fotogramas previos. Los modelos

básicos calculan medidas estadísticas sencillas como la media o la mediana de las intensidades pasadas de un píxel para obtener estimaciones robustas del modelo de fondo. Algunos de estos modelos exhiben una buena resistencia frente al ruido y a los artefactos típicos [32], pero son computacionalmente costosos si el fotograma es grande, lo cual puede ser atenuado realizando una estimación rápida, aunque menos precisa, de la medida que se esté considerando. En cuanto a los modelos actuales, podemos diferenciar a grandes rasgos los modelos no paramétricos y los paramétricos. Los primeros suelen tener unos requisitos demasiado altos. Por otro lado, los modelos basados en distribuciones de mixtura suelen tener menos requisitos de tiempo y memoria. Este motivo junto con el hecho de que permiten la posibilidad de introducir mecanismos específicos para afrontar diferentes tipos de problemas los convierten en una solución habitual para el problema de la detección del primer plano [150, 177].

A pesar de la popularidad de los métodos basados en distribuciones de mixtura, la mayoría de ellos asumen un conjunto de simplificaciones que pueden mermar su rendimiento. Por un lado, es habitual que usen el espacio de color RGB para representar los valores de entrada de los píxeles [193]. Por otro lado, algunos algoritmos de modelado de fondo ampliamente conocidos utilizan la misma varianza para modelar todas las variables de entrada [128, 126, 125], aunque no existe una razón de peso para no utilizar matrices de covarianza completas. En otras palabras, el uso de matrices de covarianza completas es una opción que existe en teoría pero que muy rara vez se implementa en la práctica. Las matrices de covarianza esféricas no suelen ser la elección óptima debido a que la varianza de las variables de entrada puede ser diferente, lo cual significa que estos métodos no se adaptan a la dispersión característica de cada variable. Por ejemplo, es de esperar que las variabilidades de los rasgos filtrados con un filtro mediana sean mucho menores que las de los rasgos que no son filtrados. También los rasgos basados en la información de los bordes (filtros paso alto) tienden a tener más variabilidad que los rasgos basados en filtros paso bajo. Además, los modelos que no consideran las covarianzas entre las variables están asumiendo de facto que son independientes entre sí, lo cual no es cierto en la mayoría de las ocasiones. Por ejemplo, las componentes rojo, verde y azul de un color típicamente crecen o disminuyen conjuntamente conforme se incrementa o reduce la luz, respectivamente. Esto significa que las componentes RGB están altamente correladas. Otro ejemplo se da en los rasgos basados en bordes tanto horizontales como verticales, en este caso los objetos con texturas tendrán valores altos en ambos tipos de rasgos, mientras que los objetos con colores homogéneos tendrán valores bajos.

La segunda de decisión a tomar para diseñar el algoritmo, y no por ello menos importante, consiste en definir qué rasgos se van a utilizar como entrada al algoritmo. Lo más inmediato consiste en utilizar directamente la información RGB de la cámara, siendo además la opción elegida por muchas propuestas. Esto tiene la ventaja de que modelar e implementar la entrada del algoritmo es fácil y directo, sin embargo es ampliamente conocido que el espacio RGB padece limitaciones a causa de los cambios de iluminación, entre otros artefactos. Separar cromaticidad y luminancia

puede ayudar a reducir estos efectos no deseados, tal y como se hace en [57], donde se utiliza el espacio de color Y'UV. Como se expuso en la Sección 2.1, el canal Y' codifica la luminancia, mientras que U y V hacen lo propio para la cromaticidad. Otra alternativa consiste en eliminar la información relacionada con la luminancia, lo cual suele hacerse dividiendo las componentes RGB por la suma de todas ellas para obtener el conjunto de rasgos RGB normalizados [194, 195, 60, 196]. De este modo se logra cierta resistencia frente a los cambios de iluminación, pero las diferentes características ópticas de los objetos hacen que los valores RGB normalizados puedan variar cuando las luces se encienden o se apagan. Una forma de eliminar la dependencia del color y por tanto de la iluminación, consiste en usar rasgos basados en las texturas de los objetos [144, 197], sin embargo no son útiles en escenas que contienen objetos o fondo con pocas texturas o sin ellas.

En la literatura se pueden encontrar propuestas que emplean combinaciones de diferentes tipos de rasgos y que han demostrado buen comportamiento en muchas situaciones. En [121] se combinan rasgos de tipo espectral, espacial y temporal bajo un marco de clasificación bayesiano. Otro enfoque consiste en elegir los rasgos mediante un algoritmo de *boosting* [198], pero es necesario un periodo de entrenamiento con fotogramas en los que no aparezcan objetos del primer plano. Es conveniente señalar que estos trabajos tienen diferencias fundamentales con nuestra propuesta ya que [121] estima las probabilidades del fondo y del primer plano mediante histogramas multivariable, mientras que [198] emplea estimadores basados en funciones núcleo con gaussianas esféricas, lo cual adolece de los inconvenientes mencionados previamente.

A priori cualquier algoritmo de detección del primer plano que esté basado en distribuciones de mixtura con gaussianas esféricas puede adaptarse para emplear matrices de covarianza completas con cualquier número de variables. No obstante, cada algoritmo asume sus propias simplificaciones, las cuales no son compatibles con el uso de matrices de covarianza completas. Por tanto, cada uno de ellos debería ser reformulado desde el principio. Para evitar esto, se ha optado por adaptar el modelo publicado en [159] y descrito en la Sección 2.5, para que pueda hacer uso de cualquier cantidad y tipo de rasgos, ya que en su concepción inicial utiliza el espacio RGB.

Por lo tanto, el objetivo de este capítulo consiste en desarrollar un método que sea capaz de superar las limitaciones que acabamos de comentar y proponer un conjunto de rasgos que logren buenos resultados. Nuestra propuesta define un modelo probabilístico que maneja cualquier cantidad de rasgos para describir un píxel. Además, dado que tiene en cuenta las correlaciones entre los rasgos, se obtiene un modelo más realista.

## 5.2. Modelo

En esta sección presentamos un modelo de mixtura probabilística para la detección del primer plano. Como se mencionó en la sección anterior, es una mejora del modelo

[159]. El nuevo modelo se define en la Subsección 5.2.1. La estimación del ruido de cuantización ha sido modificada tal y como recoge la Subsección 5.2.2. Por último, en la Subsección 5.2.3 se detallan varios aspectos destacados de la implementación realizada.

### 5.2.1. Definición

Para modelar la distribución de los rasgos de cada píxel se emplea un marco de trabajo basado en aproximación estocástica. Al igual que en [159], se usa una distribución de mixtura con un componente gaussiana para el fondo y una uniforme para el primer plano. De este modo, el modelo probabilístico $\mathrm{p}(\boldsymbol{p}_t)$ para la distribución de los valores de los rasgos en la posición $p$ viene dado por:

$$\mathrm{p}(\boldsymbol{p}_t) = \pi_{B,t} G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{B,t}(p), \boldsymbol{C}_{B,t}(p) + \boldsymbol{\Psi}\right) + \pi_{F,t} U\left(\boldsymbol{p}_t\right) \tag{5.1}$$

donde:

$$G\left(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = (2\pi)^{-D/2} \det\left(\boldsymbol{\Sigma}\right) \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x} - \boldsymbol{\mu}\right)\right) \tag{5.2}$$

$$U\left(\boldsymbol{p}_t\right) = \begin{cases} 1/Vol\left(\mathcal{S}\right) & \boldsymbol{p}_t \in H \\ 0 & \boldsymbol{p}_t \notin H \end{cases} \tag{5.3}$$

$$H = [d_1, e_1] \times [d_2, e_2] \times ... \times [d_D, e_D] \tag{5.4}$$

$$Vol\left(\mathcal{S}\right) = \prod_{h=1}^{D} \left(d_h - e_h\right) \tag{5.5}$$

Nótese que $\boldsymbol{C}_{B,t}(p)$ es una matriz de covarianza completa, tal y como se propone en [159]. Como puede observarse, este modelo difiere del planteado originalmente en el número de rasgos que determinan el valor de un píxel. Si bien en dicho trabajo se tomaba como entrada el color de un píxel, en este modelo se emplea un enfoque más amplio ya que considera como entrada $\boldsymbol{p}_t \in \mathbb{R}^D$, donde la dimensión $D$ del vector de rasgos no tiene que ser 3. De este modo pueden utilizarse tantos rasgos como se considere oportuno según las características de la secuencia que se vaya a segmentar.

Las probabilidades de cada clase para un valor observado $\boldsymbol{p}_t$ se calculan de la siguiente manera:

$$\forall i \in \{B, F\},\ R_{i,t} = P\left(i|\boldsymbol{p}_t\right) = \frac{\pi_i \mathrm{p}\left(\boldsymbol{p}_t|i\right)}{\pi_B \mathrm{p}\left(\boldsymbol{p}_t|B\right) + \pi_F \mathrm{p}\left(\boldsymbol{p}_t|F\right)} \tag{5.6}$$

donde $R_{B,t}$ y $R_{F,t}$ son las probabilidades a posteriori de que un píxel pertenezca al fondo o al primer plano, respectivamente.

Bajo un marco bayesiano, un píxel se declararía como parte del primer plano en el instante $t$ si y sólo si $R_{F,t} > R_{B,t}$. Sin embargo, en ocasiones es interesante introducir un umbral de probabilidad $\rho \in [0, 1]$ de tal modo que el píxel se considere primer plano si se satisface la siguiente condición:

$$R_{F,n} > \rho \tag{5.7}$$

En caso de que no se mencione consideraremos decisiones bayesianas, es decir, $\rho = 0.5$.

En cuanto al algoritmo de aprendizaje, la detección de cambios repentinos en el fondo y la inicialización de variables, hay que notar que no se han modificado respecto de lo publicado en [159], por tanto para obtener más detalles sobre estas tres cuestiones se pueden consultar, respectivamente, las Subsecciones 2.5.2, 2.5.4 y 2.5.5.

### 5.2.2. Estimación del ruido de cuantización

Al igual que en el trabajo [159], la matriz diagonal $\mathbf{\Psi}$ modela tanto el ruido producido durante la cuantización de la intensidad de la luz que incide en el sensor de la cámara, como el ruido producido durante la compresión del vídeo. Supondremos que los efectos de ambos ruidos son aproximadamente iguales para todos los píxeles de la imagen, por lo tanto el valor $\mathbf{\Psi}$ será global y constante durante toda la secuencia. El valor de dicha matriz viene dado por la siguiente expresión:

$$\mathbf{\Psi} = \begin{pmatrix} \sigma_1^2 & 0 & ... & 0 \\ 0 & \sigma_2^2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & \sigma_D^2 \end{pmatrix} \tag{5.8}$$

Nótese que, a diferencia de lo propuesto en el trabajo previamente citado, se trata de una matriz de orden $D$ y no 3, ya que el modelo propuesto está diseñado para utilizar cualquier número de rasgos.

Como vimos en la Subsección 2.5.3, idealmente los valores de $\sigma_i^2$ son estimados comparando el valor original de los píxeles sin comprimir $\boldsymbol{\tau}(p)$ con los observados en el vídeo de entrada $\boldsymbol{p}_t$, esto es:

$$\forall i \in \{1, ..., D\} \;\; \sigma_i^2 = E\left[\left(\tau_i(p) - p_{i,t}\right)^2\right] \tag{5.9}$$

Sin embargo, en la práctica lo habitual es que los valores sin comprimir $\boldsymbol{\tau}(p)$ no estén disponibles. En este capítulo y a diferencia de lo propuesto en [159], resolvemos este

problema estimando su valor original mediante la aplicación de un filtro gaussiano rotacionalmente simétrico de paso bajo y de tamaño $3 \times 3$ píxeles con una desviación típica de 0.5. Este procedimiento logra resultados satisfactorios con una menor carga computacional. Los valores filtrados $\widehat{\tau}_i$ se consideran estimaciones de los valores originales $\tau_i$:

$$\forall i \in \{1, ..., D\} \ \sigma_i^2 \approx E\left[\left(\widehat{\tau}_i\left(p\right) - p_{i,t}\right)^2\right] \tag{5.10}$$

donde la esperanza se calcula promediando el color de todos los píxeles del primer fotograma.

### 5.2.3. Implementación

En esta subsección se exponen los detalles de la implementación realizada. En adelante denominaremos a este modelo MFBM (*Multiple Feature Background Model*). Dado que no está restringido el número de rasgos $D$, es fundamental realizar una implementación rápida y numéricamente estable para que el método tenga aplicación práctica.

La parte más lenta del método es la que corresponde a la ecuación 5.2, es decir, el cálculo de la densidad de probabilidad gaussiana. Su complejidad es $O\left(D^3\right)$ debido a la inversión de la matriz de covarianzas $\mathbf{\Sigma}$ y el cálculo de su determinante. El resto de ecuaciones son $O\left(D^2\right)$ o inferior. Además, la inversión de la matriz tiende a producir errores numéricos. Por lo tanto, todos los esfuerzos deben centrarse en optimizar la implementación de dicha ecuación.

El primer paso consiste en calcular la factorización de Cholesky de la matriz $\mathbf{\Sigma}$, lo cual siempre es posible gracias a que $\mathbf{\Sigma}$ es definida positiva:

$$\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^T \tag{5.11}$$

donde $\mathbf{L}$ es una matriz triangular inferior de tamaño $D \times D$. Por lo que si definimos:

$$\mathbf{y} = \mathbf{L}^{-1}\left(\boldsymbol{p}_t - \boldsymbol{\mu}\right) \tag{5.12}$$

nos permite obtener $\mathbf{y}$ como solución de un sistema triangular inferior de ecuaciones lineales:

$$\mathbf{L}\mathbf{y} = \boldsymbol{p}_t - \boldsymbol{\mu} \tag{5.13}$$

Por tanto, tenemos que:

$$\left(\boldsymbol{p}_t - \boldsymbol{\mu}\right)^T \mathbf{\Sigma}^{-1}\left(\boldsymbol{p}_t - \boldsymbol{\mu}\right) = \mathbf{y}^T\mathbf{y} \tag{5.14}$$

lo cual nos permite obtener la distancia de Mahalanobis en la ecuación 5.2 sin necesidad de calcular explícitamente $\mathbf{\Sigma}^{-1}$. Este procedimiento es más rápido y preciso que el cálculo explícito de $\mathbf{\Sigma}^{-1}$.

El último paso es el cálculo del determinante de $\mathbf{\Sigma}$:

$$\det(\mathbf{\Sigma}) = \det(\mathbf{L}) \det\left(\mathbf{L}^T\right) = (\det(\mathbf{L}))^2 \tag{5.15}$$

donde $\det(\mathbf{L})$ se obtiene como producto de los elementos de la diagonal de $\mathbf{L}$. De nuevo esto es una manera más rápida y precisa de calcular $\det(\mathbf{\Sigma})$.

## 5.3. Rasgos

A continuación presentamos el conjunto de rasgos para los píxeles de entrada que hemos considerado. La Subsección 5.3.1 define cada uno de ellos, algunos son ya conocidos, mientras que otros son nuevos. Todos han sido ajustados para que su valor este acotado en el intervalo $[0, 1]$. La Subsección 5.3.2 estudia las propiedades de los rasgos, las cuales han servido como criterio para guiar la elección de rasgos. Para concluir esta sección, en la Subsección 5.3.3, se estudia la capacidad de detección de los rasgos en tres secuencias de prueba con el objetivo de conocer el grado de idoneidad para la detección de objetos.

### 5.3.1. Descripción

Se han definido un total de 24 rasgos, denotando $\mathbf{F}_i$ al i-ésimo rasgo, donde cada rasgo es una matriz de tamaño $fil \times col$, es decir, el tamaño de los fotogramas de entrada. Los tres primeros rasgos: $\mathbf{F}_1$, $\mathbf{F}_2$ y$\mathbf{F}_3$, son los canales rojo, verde y azul del fotograma de entrada. Asumiremos que el valor de cada canal está comprendido en el intervalo $[0, 1]$.

Como ya se explicó en la Sección 5.1, los valores RGB normalizados son usados frecuentemente debido a su robustez respecto a los cambios de iluminación [194, 195, 60, 196]. Su valor se obtiene a partir de la suma de los canales RGB originales, tal y como se define a continuación:

$$\forall i \in \{4, 5, 6\}\,,\ \mathbf{F}_i = \mathbf{F}_{i-3} \oslash \mathbf{G} \tag{5.16}$$

$$\mathbf{G} = \mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3 \tag{5.17}$$

donde $\oslash$ es el operador de división elemento a elemento y $\mathbf{G}$ es la suma de los tres canales originales. En el caso de que alguno de los elementos de $\mathbf{G}$ sea cero, asumiremos que el resultado de la división es cero para esos elementos.

A continuación vamos a definir los rasgos tipo Haar de tamaño $9 \times 9$ píxeles usados en [199], los cuales contienen información de textura [200]:

$$\forall i \in \{7, ...12\}\,,\ \mathbf{F}_i = \frac{1}{3\sum_{j,k=1}^{9} h_{i-6,j,k}} \mathbf{G} * \mathbf{H}_{i-6} \tag{5.18}$$

donde $*$ representa el operador convolución. Los seis filtros $\mathbf{H}$ se definen como sigue:

$$\mathbf{H}_1 = \begin{pmatrix} \mathbf{1}_{9,3} & \mathbf{0}_{9,3} & \mathbf{1}_{9,3} \end{pmatrix} \tag{5.19}$$

$$\mathbf{H}_2 = \mathbf{H}_1^T \tag{5.20}$$

$$\mathbf{H}_3 = \frac{1}{2} \begin{pmatrix} 2 \cdot \mathbf{1}_{4,4} & \mathbf{1}_{1,1} & \mathbf{0}_{4,4} \\ \mathbf{1}_{1,1} & \mathbf{1}_{1,1} & \mathbf{1}_{1,1} \\ \mathbf{0}_{4,4} & \mathbf{1}_{1,1} & 2 \cdot \mathbf{1}_{4,4} \end{pmatrix} \tag{5.21}$$

$$\mathbf{H}_4 = \frac{1}{2} \begin{pmatrix} 2 \cdot \mathbf{1}_{9,4} & \mathbf{1}_{9,1} & \mathbf{0}_{9,4} \end{pmatrix} \tag{5.22}$$

$$\mathbf{H}_5 = \mathbf{H}_4^T \tag{5.23}$$

$$\mathbf{H}_6 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \otimes \mathbf{1}_{3,3} \tag{5.24}$$

donde $\otimes$ representa el producto de Kronecker, $\mathbf{0}_{m,n}$ es una matriz de ceros de tamaño $m \times n$ y $\mathbf{1}_{m,n}$ es una matriz de unos de tamaño $m \times n$.

Los rasgos basados en el gradiente son muy útiles porque son menos sensibles a los cambios de iluminación y además proporcionan información local de texturas [121]. En este capítulo estimamos las componentes del gradiente mediante el operador Sobel y después las escalamos y trasladamos para que estén en el intervalo $[0, 1]$. Como ya se dijo anteriormente, el modelo de fondo no se ve afectado por estas transformaciones:

$$\mathbf{F}_{13} = \frac{1}{2}\mathbf{1}_{fil,col} + \frac{1}{3 \cdot 8}\mathbf{G} * \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{5.25}$$

$$\mathbf{F}_{14} = \frac{1}{2}\mathbf{1}_{fil,col} + \frac{1}{3 \cdot 8}\mathbf{G} * \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{5.26}$$

Los siguientes dos rasgos consideran tanto información del color como del píxel adyacente, es un indicador de la textura local al píxel en cuestión. El primero es el canal rojo del píxel actual normalizado con la suma de los tres canales de dicho píxel y del inmediatamente a la izquierda del mismo:

$$\mathbf{F}_{15} = \mathbf{F}_1 \oslash (\mathbf{G} + \mathbf{G}\mathbf{U}_{col}) \tag{5.27}$$

siendo $\mathbf{U}_h$ la matriz de desplazamiento de tamaño $h \times h$, en la que todos sus elementos son nulos a excepción de la diagonal superior:

$$\forall j,k \in \{1,...h\}\,,\ U_{j,k} = \mathbb{I}\,(j+1=k) \tag{5.28}$$

donde $\mathbb{I}$ denota la función característica.

El segundo rasgo es el canal verde del píxel inferior derecho del píxel en cuestión, normalizado con la suma de los canales del píxel actual:

$$\mathbf{F}_{16} = (\mathbf{U}_{fil}\mathbf{F}_2\mathbf{Q}_{col}) \oslash \mathbf{G} \tag{5.29}$$

donde $\mathbf{Q}_h$ es la matriz de desplazamiento inferior de tamaño $h \times h$ :

$$\forall j,k \in \{1,...h\}\,,\ Q_{j,k} = \mathbb{I}\,(j=k+1) \tag{5.30}$$

Para incluir algunos rasgos más robustos, se han añadido versiones de los rasgos 1-6 filtrados con un filtro mediana bidimensional de tamaño $5 \times 5$ píxeles, el cual denotaremos como $\text{mediana}_{5,5}(\cdot)$. Este filtro se ha elegido para ofrecer equilibrio entre robustez (lo que requiere filtros de mayor tamaño) y precisión a la hora de detectar los objetos (lo que demanda tamaños más pequeños). De este modo tenemos:

$$\forall i \in \{17,...,22\}\,,\ \mathbf{F}_i = \text{mediana}_{5,5}\,(\mathbf{F}_{i-16}) \tag{5.31}$$

Para los dos últimos rasgos utilizamos dos filtros de tamaño $3 \times 3$ píxeles con el fin de extraer información sobre la textura en las inmediaciones del píxel en cuestión:

$$\mathbf{F}_{23} = \frac{1}{3\cdot 6}\mathbf{G} * \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \tag{5.32}$$

$$\mathbf{F}_{24} = \frac{1}{3\cdot 8}\mathbf{G} * \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{5.33}$$

Esto completa el conjunto de 24 rasgos que se utilizarán en este capítulo. Los rasgos 15 a 22 son nuevos, mientras que los demás son más o menos conocidos en la

literatura. Dado un conjunto de rasgos seleccionados $\mathcal{F} \subset \{1, ..., 24\}$, el patrón de entrada para el píxel cuya posición en el fotograma es $p$ viene dado por:

$$\boldsymbol{p}_t = (F_{i,t,p})_{i \in \mathcal{F}} \tag{5.34}$$

donde $\boldsymbol{p}_t \in [0,1]^D$ ya que hemos añadido los escalados y las traslaciones pertinentes en las definiciones de los rasgos (ecuaciones 5.16-5.33).

### 5.3.2. Propiedades

En esta subsección se estudian las propiedades de los rasgos presentados previamente y que además han servido de guía para seleccionar los más interesantes. Para empezar, debemos tener en cuenta que una de las dificultades más típicas que padecen los algoritmos de detección del primer plano se debe a los cambios de iluminación. Por otro lado, los artefactos que se producen durante la cuantización y compresión de la imagen son fuente de ruido que pueden mermar el rendimiento de la detección. Por estos dos motivos, vamos a realizar a continuación un estudio teórico de la robustez de los rasgos considerados respecto de estas dificultades. Adicionalmente investigaremos cómo afecta modificar qué píxeles se consideran vecinos de un píxel a la hora de calcular los rasgos.

| Rasgo | Iluminación | Ruido | Radio |
|---|---|---|---|
| 1 | No | No | 0 |
| 2 | No | No | 0 |
| 3 | No | No | 0 |
| 4 | Sí | No | 0 |
| 5 | Sí | No | 0 |
| 6 | Sí | No | 0 |
| 7 | No | No | 4 |
| 8 | No | No | 4 |
| 9 | No | No | 4 |
| 10 | No | No | 4 |
| 11 | No | No | 4 |
| 12 | No | No | 4 |

| Rasgo | Iluminación | Ruido | Radio |
|---|---|---|---|
| 13 | No | No | 4 |
| 14 | No | No | 4 |
| 15 | Sí | No | 1 |
| 16 | Sí | No | 1 |
| 17 | No | Sí | 2 |
| 18 | No | Sí | 2 |
| 19 | No | Sí | 2 |
| 20 | Sí | Sí | 2 |
| 21 | Sí | Sí | 2 |
| 22 | Sí | Sí | 2 |
| 23 | No | No | 1 |
| 24 | No | No | 1 |

**Cuadro 5.1:** Resumen de las propiedades de los rasgos. En las columnas de iluminación, "Sí" indica que a priori el rasgo es invariante frente a cambios de iluminación. En las columnas de ruido, "Sí" indica que el rasgo es inmune frente a una pequeña porción de píxeles corrompidos por el ruido que producen los artefactos de cuantización y compresión. Por último, las columnas denotadas como radio indican cuál es la distancia al píxel más lejano que puede afectar al valor del rasgo del píxel que se esté considerando.

Un cambio de iluminación puede expresarse aproximadamente como una transformación lineal que escala las tres componentes RGB del mismo modo, por lo tanto después de un cambio de iluminación, los rasgos RGB varían de la siguiente manera:

$$\forall i \in \{1, 2, 3\}\,,\ \bar{\mathbf{F}}_i = \lambda \mathbf{F}_i \tag{5.35}$$

donde $\lambda$ es el factor de escala. Esto implica que los rasgos 1-3 son sensibles a los cambios de iluminación, lo cual no es deseable. Por otro lado, los rasgos RGB normalizados, es decir, aquellos en los que los valores RGB son divididos por la suma de las tres componentes RGB, satisfacen que:

$$\forall i \in \{4, 5, 6, 15, 16\}\,,\ \bar{\mathbf{F}}_i = \mathbf{F}_i \tag{5.36}$$

lo que significa que a priori los rasgos 4, 5, 6, 15 y 16 son invariantes frente a cambios de iluminación. Los rasgos tipo Haar, es decir, los basados en el gradiente y en pequeños filtros de texturas se comportan del mismo modo que los rasgos RGB:

$$\forall i \in \{7, ..., 14, 23, 24\}\,,\ \bar{\mathbf{F}}_i = \lambda \mathbf{F}_i \tag{5.37}$$

donde la traslación realizada en los rasgos basados en el gradiente (13 y 14) para que estén en el intervalo $[0, 1]$ no se ve afectada por el factor $\lambda$. Los rasgos basados en los filtros mediana (17 a 22) heredan el comportamiento de los rasgos originales (1 a 6).

La robustez ante los artefactos que se generan durante la cuantización y compresión significa que el valor del rasgo no se ve afectado cuando una fracción relativamente pequeña de píxeles es corrompida con este tipo de ruido. De los 24 rasgos propuestos, esta propiedad es satisfecha únicamente por aquellos que están basados en medidas estadísticas robustas, es decir, los rasgos basados en el filtro mediana (17 a 22). En el resto de rasgos, incluso si sólo se corrompe un píxel, afectará a la salida final.

Finalmente estudiamos el radio de vecindad de los píxeles, el cual contiene todos los píxeles que afectarán al valor del rasgo en una posición en dada. Su importancia reside en que un radio grande implica que los rasgos son menos sensibles a los pequeños detalles, lo cual puede ser beneficioso si estos detalles son irrelevantes, sin embargo puede ser un problema si los detalles corresponden a porciones de los objetos del primer plano. El radio óptimo va a depender del nivel de ruido, el tamaño de los objetos a detectar y la robustez de los rasgos. Los rasgos 1 a 6 dependen únicamente del píxel en cuestión, por lo que el radio es cero. Los rasgos tipo Haar (7 a 12) están definidos con una ventana de tamaño $9 \times 9$, por lo que el radio es 4. Los rasgos basados en gradiente y pequeñas texturas (13, 14, 23 y 24) están definidos con una ventana de tamaño $3 \times 3$, por lo que el radio es 1. Los rasgos de texturas normalizados (15 a 16) también tienen radio 1 debido a que dependen del píxel actual y de un vecino adyacente. Por último, los rasgos basados en filtros mediana (17 a 22) tienen radio 2 porque el tamaño de la máscara es de tamaño $5 \times 5$.

El Cuadro 5.1 resume las propiedades de los rasgos estudiados. La etiqueta "Sí" corresponde a características deseables de un rasgo. En la última columna se puede observar que hemos considerado un amplio rango de radios para cubrir diferentes situaciones posibles.

### 5.3.3. Capacidad de detección

A continuación vamos a medir la capacidad de detección de los rasgos referidos previamente. Hemos utilizado un banco de pruebas compuesto por las secuencias Campus, Fountain y Video4. Todas ellas ya han sido utilizadas en el Capítulo 4. Recordemos que Campus y Fountain fueron creadas para el trabajo [121] y están disponibles en su página web[1]. Por otro lado, la secuencia Video4 fue diseñada para la conferencia internacional VSSN'06[2].

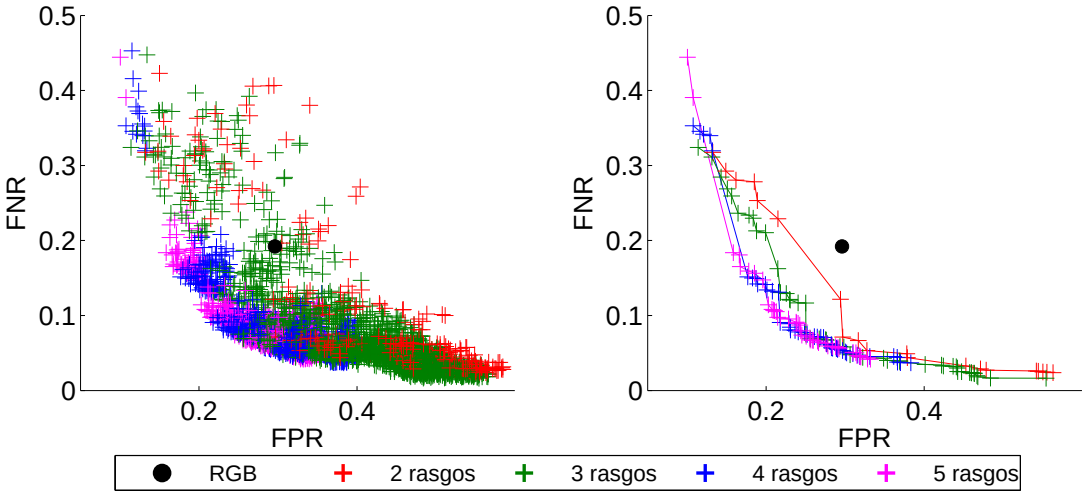



**Figura 5.1:** FNR vs FPR. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

Como se puntualizó en la Subsección 2.4.2, hay dos maneras de calcular el rendimiento en la detección de objetos: considerar globalmente las medidas básicas de los píxeles de todos los fotogramas de la secuencia para obtener la medida de rendimiento deseada, o bien sumar las medidas básicas de todos los píxeles de un fotograma y obtener la medida deseada en cada fotograma, para luego calcular la media de todos estos rendimientos. Marcaremos con un apostrofe aquellas medidas de rendimiento que hayan sido calculadas mediante la primera opción.

En primer lugar hemos probado todas las combinaciones posibles de pares y ternas de rasgos descritos en la Subsección 5.3.1, calculando en cada caso seis medidas

[1]http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
[2]http://mmc36.informatik.uni-augsburg.de/VSSN06-OSAC

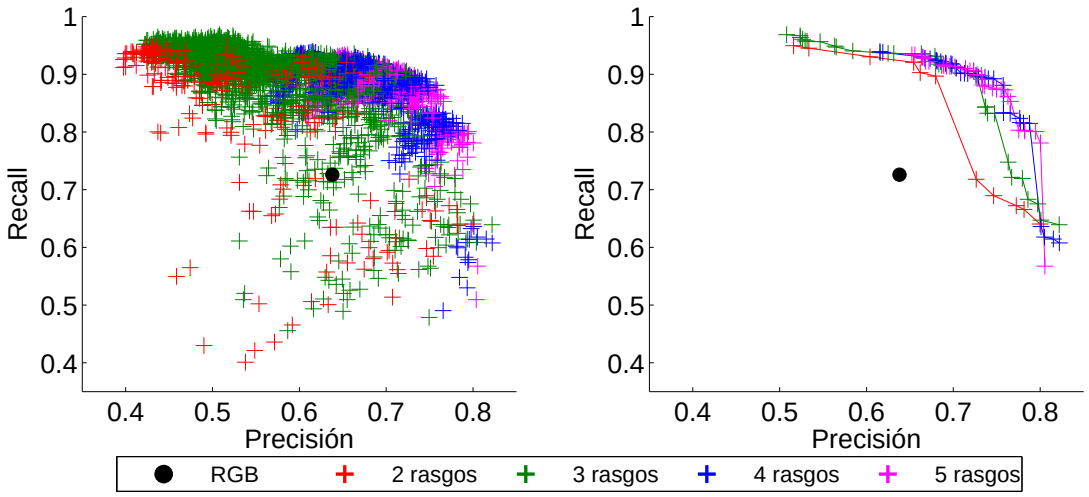



**Figura 5.2:** Precisión vs *recall*. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

de rendimiento y finalmente la media de cada una de ellas en las tres secuencias. Se ha observado que en general los mejores resultados se obtienen combinando los rasgos de este conjunto: $\{1,\ 2, ...,\ 6,\ 17,\ 18, ...,\ 24\}$. Por lo tanto, hemos usado este conjunto reducido de 14 rasgos para llevar a cabo pruebas con combinaciones de 4 o 5 rasgos. Nótese que sería inviable llevar a cabo pruebas con todas las combinaciones de 4 o 5 rasgos del conjunto original de 24 rasgos.

Debemos tener en cuenta que existen dos pares de medidas que se compensan entre sí: FPR vs FNR y precisión vs *recall*. Para identificar fácilmente las mejores combinaciones de rasgos bajo este contexto de optimización multiobjetivo, consideramos el concepto de frente de Pareto [201, 202, 203, 204]. En concreto vamos a calcular el frente de Pareto del conjunto de combinaciones de rasgos con $i \in \{2, 3, 4, 5\}$ rasgos para cada uno de los siguientes pares de medidas: $\{FNR,\ FPR\}$, $\{PR,\ RC\}$ y $\{AC,\ FM\}$. Una combinación de $i$ rasgos pertenece al frente de Pareto del conjunto de combinaciones de $i$ rasgos para un par de medidas de rendimiento si y sólo si no existe otra combinación de $i$ rasgos que sea mejor en ambas medidas de rendimiento. Las configuraciones (combinaciones de rasgos) que pertenecen al frente de Pareto son las más interesantes.

Las Figuras 5.1-5.3 muestran los resultados obtenidos. Cada punto de las gráficas corresponde al rendimiento medio obtenido por un conjunto de rasgos en concreto, calculado sobre las tres secuencias mencionadas previamente. Hemos marcado el rendimiento del conjunto de rasgos RGB con un punto negro grande, es decir, los resultados del conjunto de rasgos $\{1,\ 2,\ 3\}$. A partir de estos resultados se pueden obtener las siguientes conclusiones:

- Los rasgos que conforman el espacio RGB estándar logran resultados pobres en todos los aspectos. La mayoría de las otras combinaciones los superan.

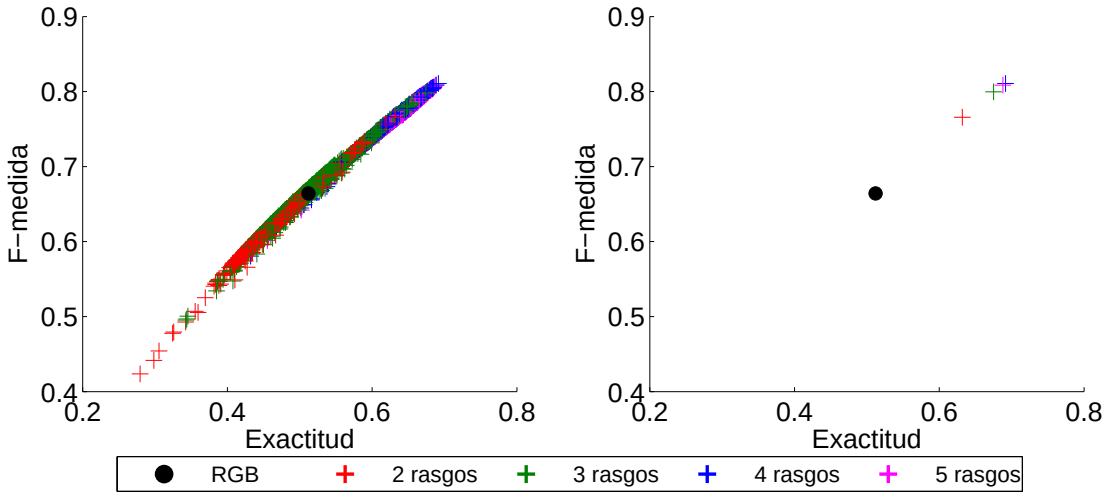



**Figura 5.3:** Exactitud vs F-medida. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

- La medida exactitud y F-medida están muy correladas (Figura 5.3) y pueden verse como medidas globales de rendimiento. Por otro lado, existe compensación entre FPR - FNR, y entre precisión - *recall* (Figuras 5.1-5.2), tal y como se había mencionado.
- En general el rendimiento mejora conforme añadimos más rasgos, pero la mejoría es progresivamente menor (ver segunda columna de las Figuras 5.1-5.3). De este modo, hemos llegado a un punto donde no compensa incrementar la complejidad computacional para ganar relativamente poco rendimiento.
- En el caso de FPR vs FNR y de la precisión vs *recall* (Figuras 5.1-5.2), la mayoría de las configuraciones están cerca del frente de Pareto, sólo una minoría de ellas son realmente malas. Por otro lado, la Figura 5.3 que corresponde a exactitud vs F-medida es más clara y permite diferenciar aquellas configuraciones realmente buenas.

Hemos seleccionado 10 configuraciones que obtienen resultados particularmente buenos de entre aquellas que están en el frente de Pareto y que logran valores elevados de AC y FM. Estas configuraciones se muestran en el Cuadro 5.2 junto con el rendimiento obtenido. De dicho cuadro podemos extraer la siguiente información útil:

- Todas las configuraciones contienen una mayoría de rasgos basados en el filtro mediana (17 a 22), lo cual significa que el filtro mediana es muy efectivo para eliminar el ruido de fondo.
- De entre los rasgos basados en el filtro mediana, lo que más aparecen son los 20 y 22, es decir, los canales RGB normalizados filtrados con la mediana. Por lo tanto, la robustez frente a los cambios de iluminación de la normalización también mejora la discriminación entre el fondo y el primer plano.

| Rasgos | FPR | FNR | PR | RC | AC | FM |
|---|---|---|---|---|---|---|
| 3,20,21,22 | **0.1767** | 0.1515 | **0.7877** | 0.8149 | 0.6718 | 0.7941 |
| 4,19,20,21 | 0.2287 | 0.0890 | 0.7464 | 0.8819 | 0.6822 | 0.8047 |
| 4,19,20,21,22 | 0.2039 | 0.1050 | 0.7683 | 0.8652 | 0.6911 | 0.8102 |
| 5,6,19,20,22 | 0.2146 | 0.0975 | 0.7592 | 0.8732 | 0.6879 | 0.8085 |
| 5,19,20,22 | 0.2307 | 0.0808 | 0.7465 | 0.8921 | 0.6885 | 0.8087 |
| 5,19,21,22 | 0.2305 | 0.0845 | 0.7450 | 0.8866 | 0.6850 | 0.8056 |
| 5,19,20,21,22 | 0.1993 | 0.1054 | 0.7730 | 0.8648 | **0.6951** | **0.8126** |
| 6,19,20,21,22 | 0.2081 | 0.1005 | 0.7646 | 0.8701 | 0.6914 | 0.8104 |
| 19,20,22 | 0.2559 | **0.0690** | 0.7235 | **0.9048** | 0.6750 | 0.7996 |
| 19,20,21,22 | 0.2176 | 0.0907 | 0.7573 | 0.8807 | 0.6915 | 0.8108 |

**Cuadro 5.2:** Algunas configuraciones con buen rendimiento. Se muestran ordenadas lexicográficamente. Los mejores resultados para cada medida están marcados en negrita.

- Los filtro tipo Haar, filtros basados en el gradiente y texturas de color no aparecen. Por lo tanto no son óptimos para el modelado basado en mixtura probabilística.

Ahora que hemos estudiado el rendimiento de los 24 rasgos considerados, estamos en disposición de comparar nuestro método con otras propuestas del estado del arte, lo cual se lleva a cabo en la Sección 5.4.

## 5.4. Resultados experimentales

En esta sección vamos a mostrar el rendimiento de nuestra propuesta en comparación con otros métodos de segmentación actuales. El código fuente junto con algunos vídeos demostrativos están disponibles en la web[3]. Para este estudio hemos probado varias secuencias que muestran diversas situaciones y un gran número de configuraciones para cada método.

En primer lugar, los métodos competidores y las secuencias de prueba son descritas en las Subsecciones 5.4.1 y 5.4.2, respectivamente. Después, en la Subsección 5.4.3, se analiza el efecto de utilizar matrices de covarianza completas en nuestra propuesta. La Subsección 5.4.4 está dedicada a discutir la elección de parámetros para las propuestas probadas. Los resultados cualitativos y cuantitativos obtenidos en las pruebas propuestas se muestran, respectivamente, en las Subsecciones 5.4.5 y 5.4.6. Para concluir hemos considerado dos conjuntos de experimentos adicionales, la Subsección 5.4.7 corresponde a un estudio mediante curvas ROC (*Receiver Operating Characteristic*) de nuestro método para varios conjuntos de rasgos; la Sub-

[3]http://www.lcc.uma.es/ %7Eezeqlr/backfeat/backfeat.html

sección 5.4.8 realiza una comparación del rendimiento de los métodos en secuencias grabadas bajo condiciones especialmente difíciles.

### 5.4.1. Métodos

Nuestro método analiza la escena píxel a píxel, por lo tanto hemos elegido cinco métodos de referencia de este tipo, es decir, métodos a nivel de píxel. Todos ellos son descritos en detalle en la Subsección 2.3.2. El más antiguo es el método Pfinder [57] que se caracteriza por utilizar una única distribución gaussiana para modelar el fondo. Hemos probado también tres métodos basados en mixtura de gaussianas: GrimsonGMM [125], KaewGMM [126] y ZivkovicGMM [127, 128]. Y también la variante difusa del método SOBS basado en mapas autoorganizados [145] y denominada FASOM.

Para probar estos métodos se ha empleado la versión 1.3.0 de la biblioteca BGS. Nuestra propuesta se ha implementado en Matlab, utilizando ficheros MEX para aquellas partes con mayores requisitos de tiempo y *script* Matlab para el resto.

Al igual que en otras ocasiones, no se ha utilizado ningún post-procesado en ninguno de los método para tratar de ser los más justos posible. Todos los experimentos mostrados en este capítulo han sido ejecutados en un ordenador personal con un quad core 3.10 GHz CPU, 6 GB RAM y hardware estándar. La implementación de nuestra propuesta no utiliza recursos GPU, por lo que no requiere ningún hardware gráfico particular.

### 5.4.2. Secuencias

En la videovigilancia se producen situaciones problemáticas que dificultan la segmentación, en concreto es habitual situaciones como: cambios de iluminación abruptos, rasgos de los píxeles del primer plano que deberían ser subsumidos por el modelo de fondo y fondos dinámicos. Por ello hemos elegido un conjunto de seis vídeos de prueba estándar que muestran escenas tanto de interior como de exterior y que ejemplifican buena parte de estos problemas ( Subsecciones 5.4.5-5.4.7). También hemos probado nuestra propuesta en cuatro vídeos adicionales que se caracterizan por mostrar situaciones particularmente desafiantes (Subsección 5.4.8).

Los vídeos de prueba estándar han sido descritos en el capítulo anterior. Del trabajo [121] hemos utilizado las secuencias Meeting room (denominada MR) y Water surface (WS), disponibles en su página web[4], cuyo interés reside en el movimiento continuo del fondo. Otro de los vídeos considerados es Lobby (LB) cuya característica más relevante consiste en los cambios de iluminación bruscos. También se ha empleado la secuencia Video2 (V2) con objetos de primer plano insertados artificialmente y que fue creada para la conferencia internacional VSSN'06[5]. Del repositorio

[4]http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
[5]http://mmc36.informatik.uni-augsburg.de/VSSN06-OSAC

CAVIAR[6] sólo se ha elegido la secuencia OneShopOneWait1cor (OS) debido a que el *ground truth* no está disponible y ha sido necesario segmentarlo manualmente. Para concluir el conjunto de vídeos estándar hemos elegido la secuencia LevelCrossing (LC) [192].

El conjunto de secuencias adicional ha sido elaborado a partir de dos repositorios ampliamente conocidos: ChangeDetection.net[7] [149] y BMC 2012 benchmark[8] [205]. Las secuencias elegidas muestran condiciones adversas tales como viento, nieve o iluminación deficiente que dificultan la segmentación de la escena. Los dos primeros vídeos son blizzard (BL) y snowFall (SF), ambos de ChangeDetection.net y representan situaciones de niebla y nieve, respectivamente. Los dos siguientes son Tunnel (TL), que consiste en una escena con iluminación heterogénea y Windy day (WD), donde se muestra una cámara de tráfico con vehículos circulando rápidamente y viento, ambas secuencias pertenecen al BMC 2012 benchmark.

### 5.4.3. Estudio sobre el tipo de matriz de covarianzas

Como ya se apuntó previamente, nuestro método utiliza matrices de covarianza completas. Para valorar su rendimiento frente al uso de matrices de covarianza diagonales (o esféricas), hemos comparado cuantitativamente ambas alternativas. Los resultados se muestran en las Figuras 5.4-5.6 y corresponden a los rendimientos medios de las configuraciones estudiadas (ver Subsección 5.4.4), tanto para las matrices de covarianza completas como para las diagonales.

El uso de matrices de covarianza completas logra una menor FPR sin incrementar significativamente FNR (véase Figura 5.4). En términos de PR y RC, las matrices de covarianza completas son capaces de mejorar PR sin empeorar RC (Figura 5.5). Basándose en los resultados de AC vs F-medida, podemos confirmar que los mejores resultados se obtienen utilizando matrices de covarianza completas (Figura 5.6). Es destacable que las configuraciones que peores resultados obtienen utilizando matrices de covarianza completas, tienen un rendimiento similar a las mejores configuraciones que utilizan matrices de covarianza diagonales (primera columna de la Figura 5.6).

Por lo tanto, estos resultados fundamentan el uso de matrices de covarianza completas en nuestro modelo (Sección 5.2). Sin embargo, debe tenerse en cuenta que cualquier distribución puede aproximarse con el grado de precisión deseado utilizando un número suficientemente alto de componentes gaussianas diagonales en la mixtura.

[6] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[7] http://www.changedetection.net/
[8] http://bmc.univ-bpclermont.fr/

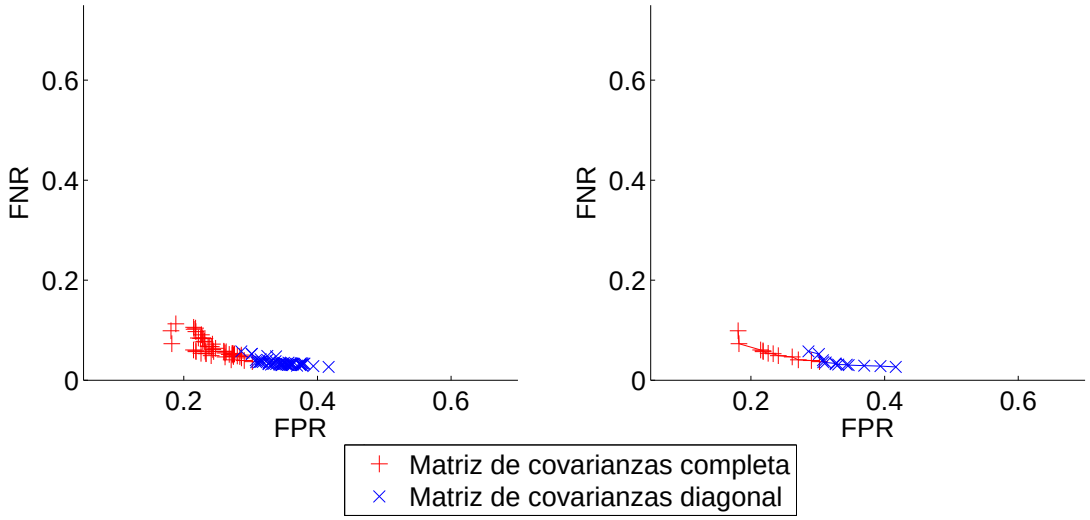



**Figura 5.4:** FNR vs FPR de cada tipo de matriz de covarianza. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

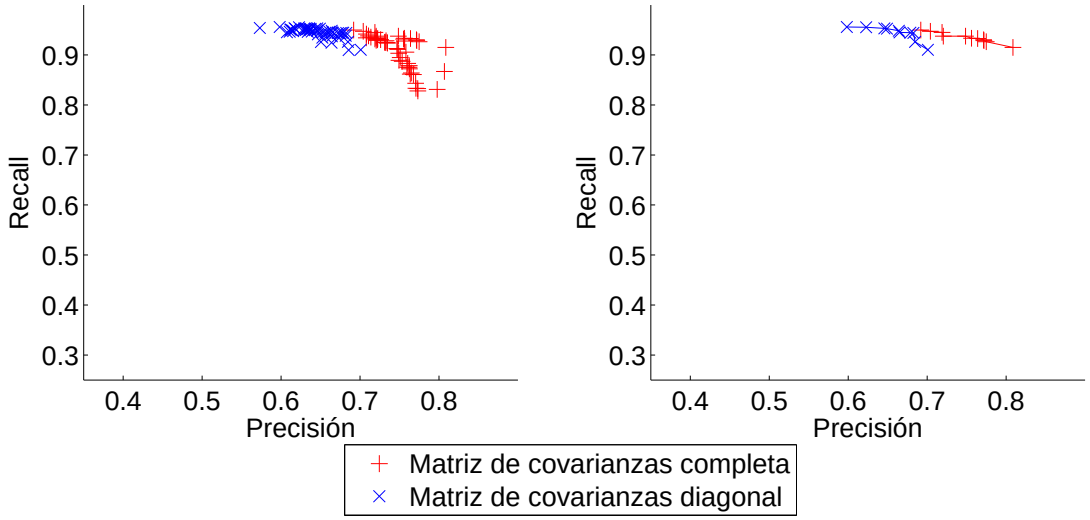



**Figura 5.5:** Precisión vs *recall* de cada tipo de matriz de covarianza. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

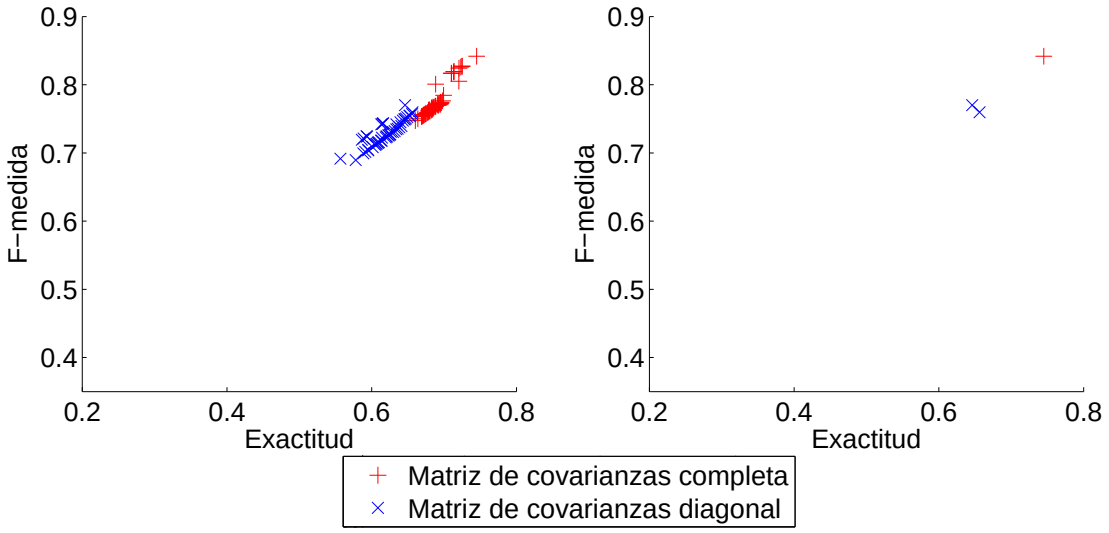



**Figura 5.6:** Exactitud vs F-medida de cada tipo de matriz de covarianza. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

### 5.4.4. Selección de parámetros

Para el algoritmo propuesto hemos ajustado el tamaño del paso $\varepsilon$ y el conjunto de rasgos $\mathcal{F}$, utilizando 5 y 10 valores distintos respectivamente, por lo que hemos probado un total de 50 configuraciones diferentes. En lo que se refiere a la inicialización hemos empleado 100 fotogramas, así pues $L = 100$. Por otro lado, mediante el estudio de varios vídeos, hemos determinado que $Z = 250$ representa un buen compromiso entre adaptación a los cambios de iluminación bruscos y estabilidad del modelo de fondo, por lo que hemos usado dicho valor en todos los experimentos.

En el Cuadro 5.3 se muestra el conjunto de parámetros de prueba. Como se puede observar, hemos probado más configuraciones en los competidores que en nuestra propuesta con el fin de tratar de ser lo más justos posible. Los valores probados para los competidores son aquellos recomendados en los artículos correspondientes a dichos métodos, también los que se usan por defecto en la biblioteca BGS y además hemos añadido otros que logran buenos resultados, pero que no están entre los mencionados previamente. El rango de valores considerados ha sido mayor en aquellos parámetros que afectan de manera más significativa al resultado. Por ejemplo, hemos considerado más de 300 configuraciones para el método FASOM debido a su alto número de parámetros.

### 5.4.5. Resultados cualitativos

El procedimiento de evaluación consiste en probar todas las configuraciones expuestas más arriba en cada una de las seis secuencias referidas en la Subsección 5.4.2.

| Método | Parámetros |
|---|---|
| Pfinder | Umbral, $T = \{10, 12, 14, 16\}$<br>Tasa de aprendizaje, $\alpha = \{0.0001, 0.002, 0.004, 0.006, 0.008, 0.01\}$<br>Fotogramas de aprendizaje, $N = \{20, 30, 40\}$ |
| GrimsonGMM | Umbral de emparejamiento, $T_{\sigma^2} = \{8, 9, 10, 11, 12\}$<br>Tasa de aprendizaje,<br>$\alpha = \{0.0025, 0.005, 0.0075, 0.01, 0.0125, 0.015, 0.0175, 0.02\}$<br>Número de componentes gaussianas en la mixtura del modelo, $K = \{3, 4, 5\}$ |
| KaewGMM | Tasa de aprendizaje,<br>$\alpha = \{0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01\}$<br>Fotogramas de aprendizaje, $N = \{100, 200\}$<br>Número de componentes gaussianas en la mixtura del modelo, $K = \{4, 5, 6\}$<br>Umbral, $T = \{0.6, 0.7, 0.8\}$ |
| ZivkovicGMM | Tasa de aprendizaje,$\alpha = \{0.0001, 0.0005, 0.001, 0.005, 0.01\}$<br>Fotogramas de aprendizaje, $N = \{400, 500\}$<br>Umbral de emparejamiento, $T_{\sigma^2} = \{8, 10, 12, 14, 16, 18, 20\}$ |
| FASOM | Fotogramas para ordenación, $N = \{79, 81, 83\}$<br>Umbral durante la ordenación, $\epsilon_0 = \{200, 220, 240, 260\}$<br>Umbral durante la convergencia, $\epsilon_1 = \{80, 90, 100\}$<br>Tasa de aprendizaje durante la ordenación, $\alpha_0 = \{240, 255, 270\}$<br>Tasa de aprendizaje durante la convergencia, $\alpha_1 = \{36, 38, 40\}$ |
| MFBM | Tamaño del paso, $\varepsilon = \{0.0001, 0.0005, 0.001, 0.005, 0.01\}$<br>Conjunto de rasgos mostrados en el Cuadro 5.2, $\mathcal{F}$ |

**Cuadro 5.3:** Parámetros utilizados en los experimentos, la combinación de todos ellos conforma el conjunto de configuraciones probadas.

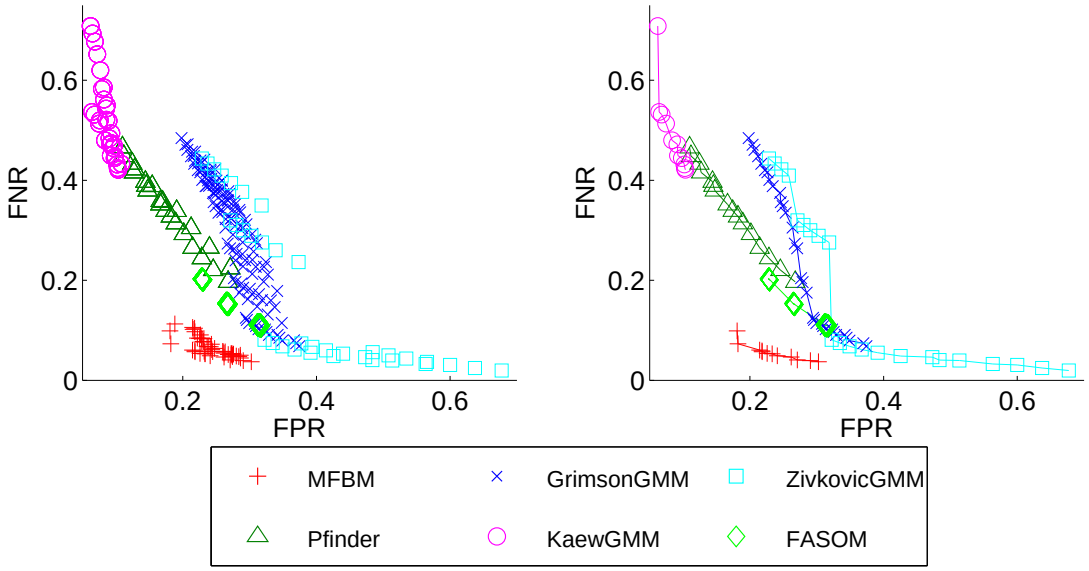



**Figura 5.7:** FNR vs FPR de cada método en las secuencias estándar. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto.

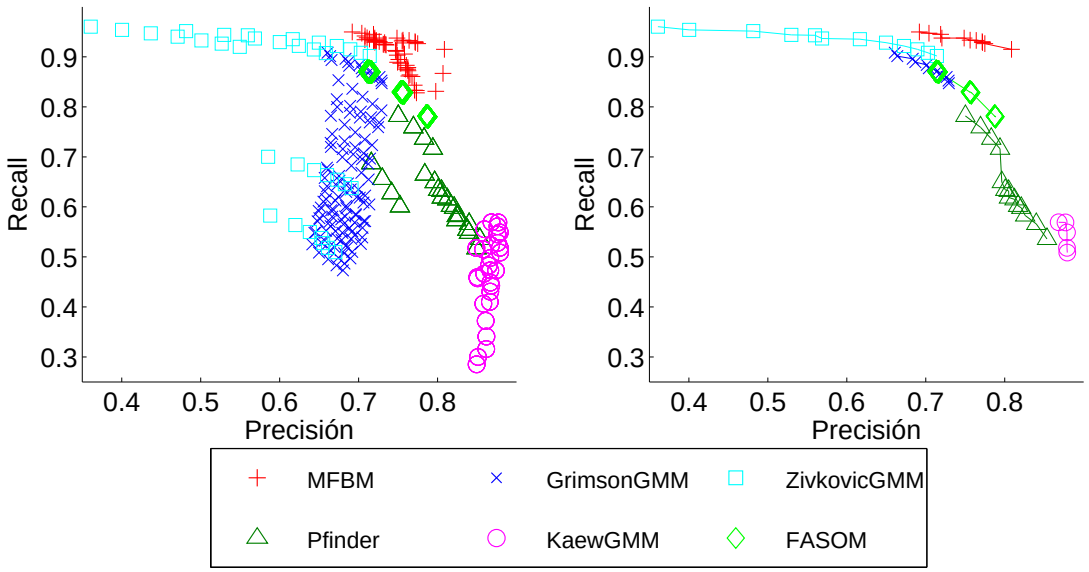



**Figura 5.8:** Precisión vs *recall* de cada método en las secuencias estándar. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto

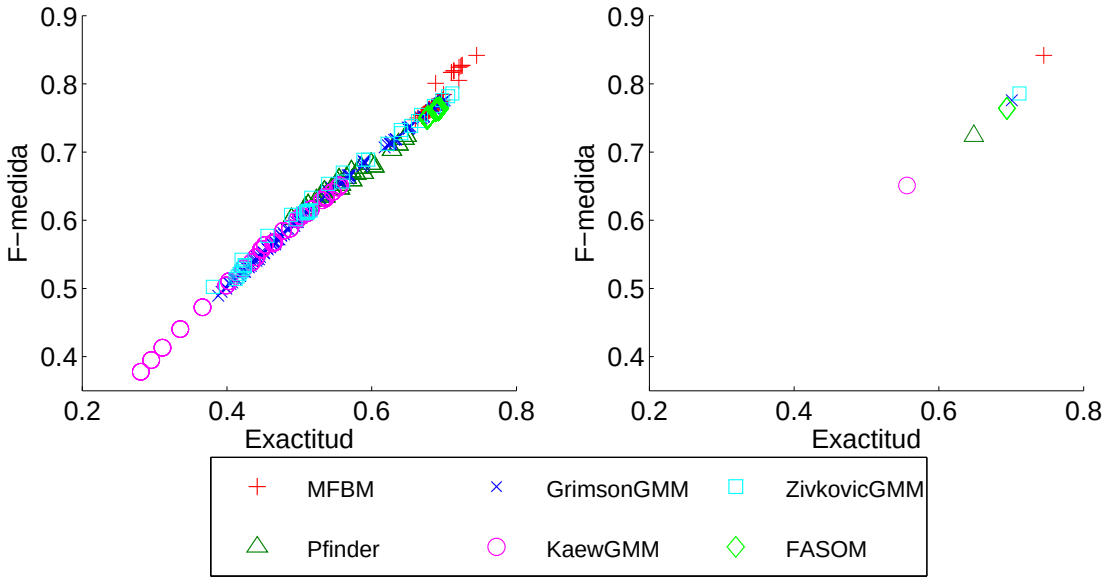



**Figura 5.9:** Exactitud vs F-medida de cada método en las secuencias estándar. A la izquierda se muestran todas las configuraciones probadas, a la derecha las configuraciones en el frente de Pareto

El rendimiento medio de dichas configuraciones en las secuencias es mostrado en las Figuras 5.7-5.9. Cada punto de estas gráficas indica el valor medio obtenido por una configuración en los seis vídeos de prueba estándar.

En las Figuras 5.10-5.13 mostramos algunas salidas de los métodos estudiados. Las filas corresponden a diferentes fotogramas de los vídeos de prueba. La primera y segunda columna son el fotograma original y el *ground truth,* respectivamente. El resto de columnas son, en este orden, los resultados de aplicar los métodos MFBM, FASOM, ZivkovicGMM, KaewGMM, GrimsonGMM y Pfinder a dichos fotogramas. Los resultados mostrados corresponden a las configuraciones que mejor rendimiento obtienen en términos de F-medida para cada una de las secuencias, véase el Cuadro 5.6 para más detalles.


**Figura 5.10:** Resultados experimentales en escenas con objetos de primer plano estacionarios. Los fotogramas corresponden a la secuencia WS donde el fondo de la escena se mueve continuamente y una persona aparece caminando de izquierda a derecha. Las tres filas corresponden, respectivamente, a los fotogramas 1499, 1523 y 1605.

Al igual que en otros casos, se producen situaciones típicamente problemáticas de cara a la segmentación, por ejemplo las que se derivan del camuflaje y de la sombra proyectada. En el caso del camuflaje, parece que a nuestra propuesta le afecta en menor grado que al resto de métodos. Este problema se presenta cuando el método no es capaz de distinguir los píxeles del primer plano debido a que son muy similares a los del modelo de fondo, por lo que los segmenta erróneamente como parte del fondo. Si bien en todas las secuencias se observan casos en los que se producen efectos de camuflaje (Figuras 5.10-5.13), en las secuencias MR y LB es muy evidente (filas 2 y 3 de las Figuras 5.11 y 5.12). A pesar de todo, es necesario puntualizar que existen situaciones especialmente difíciles donde nuestra propuesta también falla durante la segmentación de los píxeles del primer plano debido al camuflaje, tal y como le ocurre a los métodos competidores, por ejemplo, secuencia OS (fila 4 de la Figura 5.13).

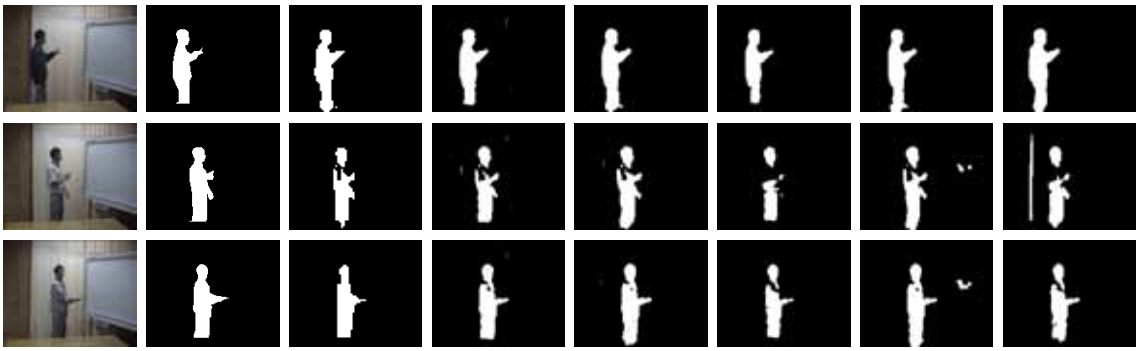



**Figura 5.11:** Resultados experimentales en una secuencia con variabilidad en el fondo. En el vídeo MR aparece una persona en tres ocasiones con diferentes indumentarias y las cortinas se mueven debido al viento. Las tres filas corresponden, respectivamente, a los fotogramas 2774, 3226 y 3857.

La sombra que proyectan los objetos se convierte en un problema cuando los métodos la marcan como parte del primer plano, lo cual es un error. Buenos ejemplo de dicho problema son los fotogramas de la secuencia MR (Figura 5.11), algunos de LB (segunda y tercera fila de la Figura 5.12) y en particular la secuencia OS (filas 4 y 5 de la Figura 5.13). En estos casos observamos que el método KaewGMM sortea prácticamente todos estos efectos indeseados, pero a costa de un número elevado de falsos negativos, situación que se muestra en la Figura 5.7. En este sentido Pfinder también es robusto y, aunque también sufre muchos falsos negativos, no es tan acusado como en el caso de KaewGMM.

Otro de problema habitual consiste en marcar como primer plano los píxeles correspondiente a un fondo dinámico. La secuencia WS (filas 1 y 2 de la Figura 5.10) muestra una escena exterior donde se produce esta situación. Nuestro método detecta el primer plano y también segmenta correctamente el fondo dinámico. Por otro lado, métodos como ZivkovicGMM, Pfinder y, en menor grado FASOM, fallan al segmentar el fondo. GrimsonGMM y KaewGMM no tienen este problema, sin embargo sufren otras dificultades que les hacen obtener peores resultados que nuestra propuesta.

Como ya se comentó previamente, el vídeo LB (Figura 5.12) es un ejemplo interesante de cambios de iluminación bruscos, los cuales ocurren en las aplicaciones reales. FASOM, Pfinder y MFBM exhiben un comportamiento similar cuando se produce el cambio de iluminación (fila 1 de la Figura5.12). Sin embargo, nuestro método es mejor en los fotogramas posteriores a dichos cambios de iluminación (filas 2 y 3 de la Figura 5.12), mostrando pocos efectos de camuflaje y ninguno de sombra proyectada. Esta secuencia también es relevante para analizar el desempeño de los demás métodos, ya que reaccionan correctamente en el momento del cambio de iluminación, pero en los fotogramas posteriores KaewGMM es incapaz de detectar los objetos

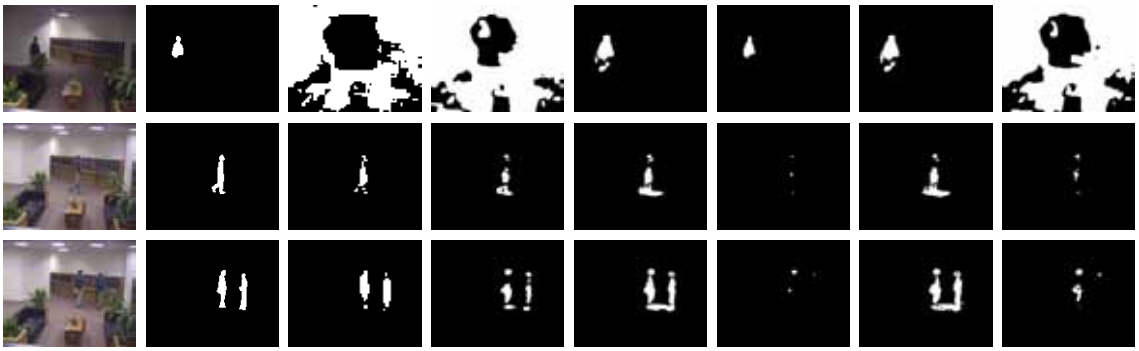



**Figura 5.12:** Resultados experimentales en el vídeo LB donde se producen cambios abruptos en la iluminación. Las tres filas corresponden, respectivamente, a los fotogramas 1634, 2265 y 2457.

del primer plano, mientras que ZivkovicGMM y GrimsonGMM tienen considerables problemas derivados de las sombras proyectadas.

### 5.4.6. Resultados cuantitativos

La subsección anterior nos ha permitido evaluar el rendimiento de las distintas propuestas de un modo visual y cualitativo. Con el ánimo de mostrar claramente las ventajas de nuestra propuesta, vamos a analizar los Cuadros 5.4 y 5.5, en donde listamos la media y la desviación típica calculada sobre los resultados obtenidos por las distintas configuraciones descritas en la Subsección 5.4.4. Como se puede observar, FASOM y ZivkovicGMM son los competidores que mejor rendimiento obtienen, sin embargo no es tan bueno como el de nuestra propuesta, salvo en las secuencias MR y LB donde nos superan. A pesar de ello, en el caso de MR los resultados medios de nuestra propuesta son muy cercanos a los del mejor método, en este caso FASOM. También es destacable el buen rendimiento de Pfinder en el vídeo OS.

Para obtener una visión aún más ajustada del rendimiento de los métodos, podemos utilizar los Cuadros 5.6 y 5.7. Aquí se muestra la mejor configuración para cada secuencia y método en términos de F-medida y exactitud, respectivamente. Por lo tanto podemos afirmar que, considerando todo el conjunto de configuraciones, existe al menos una de MFBM que es la mejor en las seis secuencias de acuerdo a la medida exactitud. Lo mismo sucede con F-medida, excepto en el caso de LB, donde ZivkovicGMM supera ligeramente a nuestra propuesta. Por lo tanto estos resultados vienen a confirmar los mostrados en los Cuadros 5.4 y 5.5.

Debe puntualizarse que los fotogramas evaluados han sido aquellos que contienen objetos del primer plano. El motivo reside en que la medida exactitud es fiable únicamente cuando hay objetos del primer plano en la escena. Nótese que si no hay

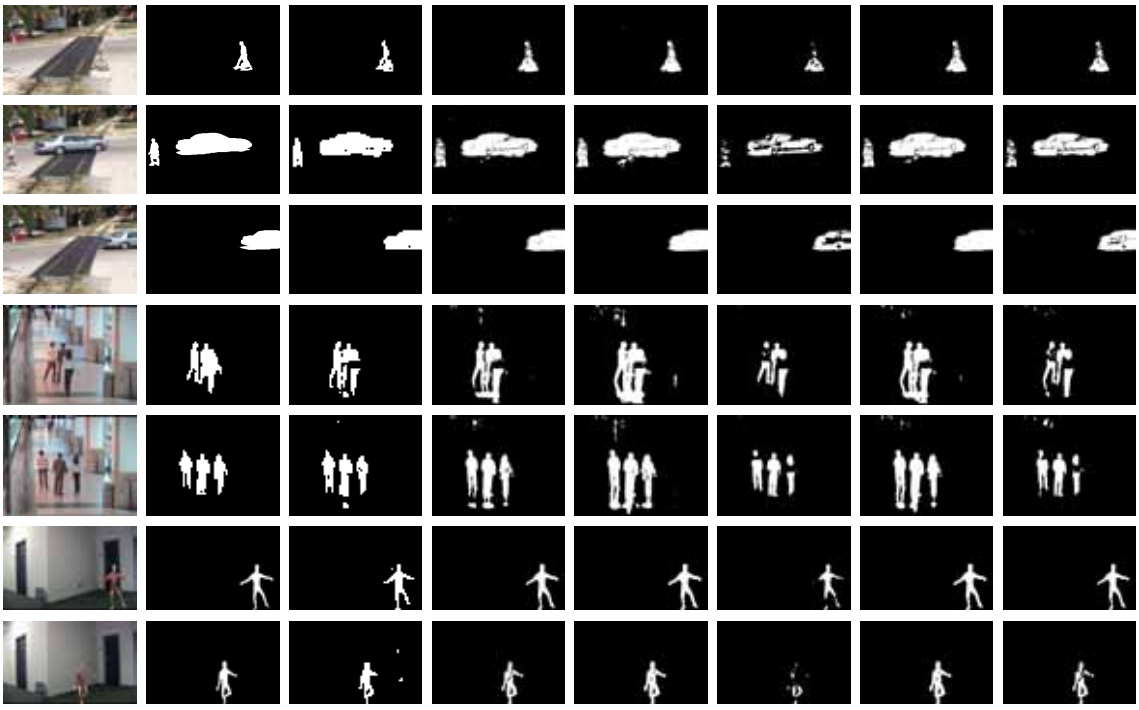



**Figura 5.13:** Resultados experimentales en varias secuencias de interior y exterior. Las tres primeras filas corresponden a la secuencia LC, fotogramas 324, 430 y 465 respectivamente. La cuarta y quinta fila a OS, fotogramas 350 y 390. La última secuencia es V2, fotogramas 332 y 469.

píxeles del primer plano, la situación más probable es $FPR = 1$, $FNR = 0$, $AC = 0$, lo cual no representa correctamente el rendimiento del método. En otras palabras, la medida exactitud se basa esencialmente en cómo de bien se detecta el primer plano y no el fondo.

En el Cuadro 5.8 podemos ver cuál ha sido el rendimiento en cada secuencia en términos de frames por segundo. Los requisitos de tiempo real se satisfacen cuando un método supera los 15 fps. Puede observarse que todos los método verifican esta condición en todas las secuencias probadas. Nuestro método es el más lento debido al procesamiento de más rasgos, pero aún es posible acelerarlo ya que nuestra implementación no utiliza recursos GPU. De entre los demás método, FASOM es el peor de ellos en este aspecto debido a que utiliza estructuras complejas como mapas autoorganizados.

Para concluir esta subsección, hemos incluido un estudio de significancia de los métodos analizados. El Cuadro 5.9 muestra los rendimientos medios de las cinco mejores configuraciones de cada secuencia y método en términos de exactitud y F-medida. Para el test de significancia hemos empleado estas configuraciones, es decir,

| Secuencia | Pfinder | Grimson GMM | Kaew GMM | Zivkovik GMM | FASOM | MFBM |
|---|---|---|---|---|---|---|
| WS | 0.87±0.03 | 0.67±0.18 | 0.80±0.12 | 0.67±0.15 | 0.91±0.00 | **0.95±0.00** |
| MR | 0.68±0.09 | 0.57±0.09 | 0.58±0.16 | 0.68±0.15 | **0.87±0.02** | 0.87±0.02 |
| LC | 0.79±0.04 | 0.86±0.02 | 0.68±0.06 | 0.80±0.07 | 0.88±0.01 | **0.92±0.01** |
| V2 | 0.53±0.04 | 0.55±0.04 | 0.45±0.05 | 0.57±0.03 | 0.60±0.01 | **0.84±0.03** |
| LB | 0.29±0.08 | 0.54±0.08 | 0.18±0.03 | **0.58±0.11** | 0.54±0.05 | 0.28±0.20 |
| OS | **0.80±0.03** | 0.54±0.12 | 0.67±0.16 | 0.53±0.12 | 0.76±0.02 | 0.79±0.03 |

**Cuadro 5.4:** Resultados cuantitativos en términos de F-medida. Cada columna corresponde a un método y cada fila a un vídeo. Las celdas muestran la media y la desviación típica del rendimiento de todas las configuraciones consideradas. Los mejores resultados para cada secuencia están marcados en negrita.

| Secuencia | Pfinder | Grimson GMM | Kaew GMM | Zivkovik GMM | FASOM | MFBM |
|---|---|---|---|---|---|---|
| WS | 0.77±0.04 | 0.55±0.20 | 0.68±0.13 | 0.53±0.16 | 0.83±0.01 | **0.91±0.01** |
| MR | 0.54±0.09 | 0.44±0.09 | 0.46±0.15 | 0.54±0.16 | **0.78±0.03** | 0.77±0.03 |
| LC | 0.66±0.06 | 0.76±0.03 | 0.53±0.06 | 0.68±0.08 | 0.78±0.02 | **0.85± 0.01** |
| V2 | 0.55±0.09 | 0.62±0.08 | 0.43±0.07 | 0.65±0.08 | 0.72±0.02 | **0.73±0.04** |
| LB | 0.21±0.06 | 0.40±0.06 | 0.12±0.03 | **0.44±0.10** | 0.41±0.05 | 0.22±0.17 |
| OS | **0.67±0.04** | 0.38±0.11 | 0.53±0.16 | 0.37±0.11 | 0.61±0.03 | 0.66±0.05 |

**Cuadro 5.5:** Resultados cuantitativos en términos de exactitud. Cada columna corresponde a un método y cada fila a un vídeo. Las celdas muestran la media y la desviación típica del rendimiento de todas las configuraciones consideradas. Los mejores resultados para cada secuencia están marcados en negrita.

el conjunto de muestras está compuesto por los cinco mejores resultados de cada secuencia y método, luego en total se usan 30 muestras por método. En negrita se ha marcado el mejor rendimiento para cada medida, mientras que el asterisco indica que la diferencia respecto del segundo mejor método es significativa con un nivel 99 % de confianza. Por lo tanto, esto viene a poner de relieve el buen rendimiento general de nuestra propuesta.

| Secuencia | Método | F-medida | Parámetros |
|---|---|---|---|
| WS | Pfinder | 0.91 | $T = 14$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.93 | $T_{\sigma^2} = 12$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.87 | $\alpha = 0.003$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.91 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.91 | $\epsilon_0 = 260$, $\epsilon_1 = 90$, $\alpha_0 = 270$, $\alpha_1 = 40$, $N = 83$ |
| | MFBM | **0.96** | $\varepsilon = 0.005$, $\mathcal{F} = \{5, 6, 19, 20, 22\}$ |
| MR | Pfinder | 0.81 | $T = 10$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.78 | $T_{\sigma^2} = 12$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.78 | $\alpha = 0.001$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.87 | $\alpha = 0.0005$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.89 | $\epsilon_0 = 240$, $\epsilon_1 = 100$, $\alpha_0 = 240$, $\alpha_1 = 40$, $N = 79$ |
| | MFBM | **0.90** | $\varepsilon = 0.005$, $\mathcal{F} = \{19, 20, 22\}$ |
| LC | Pfinder | 0.86 | $T = 12$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.89 | $T_{\sigma^2} = 10$, $\alpha = 0.0025$, $K = 4$ |
| | KaewGMM | 0.73 | $\alpha = 0.005$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.88 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.89 | $\epsilon_0 = 260$, $\epsilon_1 = 80$, $\alpha_0 = 240$, $\alpha_1 = 40$, $N = 79$ |
| | MFBM | **0.93** | $\varepsilon = 0.001$, $\mathcal{F} = \{3, 20, 21, 22\}$ |
| V2 | Pfinder | 0.60 | $T = 10$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.61 | $T_{\sigma^2} = 8$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.51 | $\alpha = 0.005$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.61 | $\alpha = 0.0005$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.61 | $\epsilon_0 = 200$, $\epsilon_1 = 80$, $\alpha_0 = 255$, $\alpha_1 = 36$, $N = 81$ |
| | MFBM | **0.88** | $\varepsilon = 0.005$, $\mathcal{F} = \{5, 19, 21, 22\}$ |
| LB | Pfinder | 0.42 | $T = 10$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.71 | $T_{\sigma^2} = 9$, $\alpha = 0.0025$, $K = 5$ |
| | KaewGMM | 0.26 | $\alpha = 0.01$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | **0.74** | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.60 | $\epsilon_0 = 200$, $\epsilon_1 = 80$, $\alpha_0 = 240$, $\alpha_1 = 36$, $N = 83$ |
| | MFBM | 0.72 | $\varepsilon = 0.0001$, $\mathcal{F} = \{3, 20, 21, 22\}$ |
| OS | Pfinder | 0.83 | $T = 16$, $\alpha = 0.002$, $N = 20$ |
| | GrimsonGMM | 0.76 | $T_{\sigma^2} = 12$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.84 | $\alpha = 0.004$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.71 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.79 | $\epsilon_0 = 220$, $\epsilon_1 = 100$, $\alpha_0 = 240$, $\alpha_1 = 40$, $N = 79$ |
| | MFBM | **0.87** | $\varepsilon = 0.005$, $\mathcal{F} = \{3, 20, 21, 22\}$ |

**Cuadro 5.6:** Mejores configuraciones en términos de F-medida. La primera columna indica el nombre de la secuencia y la segunda el método. La tercera columna indica el rendimiento obtenido por la mejor configuración de cada método para cada secuencia. En la última columna se muestra la configuración que obtiene dicho rendimiento. La F-medida del método que logra mejor resultado en cada secuencia se ha marcado en negrita.

| Secuencia | Método | Exactitud | Parámetros |
|---|---|---|---|
| WS | Pfinder | 0.84 | $T = 14$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.87 | $T_{\sigma^2} = 12$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.78 | $\alpha = 0.003$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.83 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.84 | $\epsilon_0 = 260$, $\epsilon_1 = 90$, $\alpha_0 = 270$, $\alpha_1 = 40$, $N = 83$ |
| | MFBM | **0.92** | $\varepsilon = 0.005$, $\mathcal{F} = \{5, 6, 19, 20, 22\}$ |
| MR | Pfinder | 0.69 | $T = 10$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.66 | $\mathcal{T}_{\sigma^\in} = 12$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.65 | $\alpha = 0.001$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.77 | $\alpha = 0.0005$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.80 | $\epsilon_0 = 240$, $\epsilon_1 = 100$, $\alpha_0 = 240$, $\alpha_1 = 40$, $N = 79$ |
| | MFBM | **0.82** | $\varepsilon = 0.005$, $\mathcal{F} = \{19, 20, 22\}$ |
| LC | Pfinder | 0.76 | $T = 10$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.80 | $\mathcal{N} = 9$, $\alpha = 0.0025$, $K = 5$ |
| | KaewGMM | 0.58 | $\alpha = 0.005$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.79 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.80 | $\epsilon_0 = 260$, $\epsilon_1 = 80$, $\alpha_0 = 240$, $\alpha_1 = 40$, $N = 79$ |
| | MFBM | **0.87** | $\varepsilon = 0.001$, $\mathcal{F} = \{3, 20, 21, 22\}$ |
| V2 | Pfinder | 0.73 | $T = 10$, $\alpha = 0.0001$, $N = 20$ |
| | GrimsonGMM | 0.74 | $T_{\sigma^2} = 8$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.52 | $\alpha = 0.005$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.74 | $\alpha = 0.0005$, $N = 400$, $\mathcal{T}_{\sigma^\in} = 20$ |
| | FASOM | 0.74 | $\epsilon_0 = 200$, $\epsilon_1 = 80$, $\alpha_0 = 255$, $\alpha_1 = 36$, $N = 81$ |
| | MFBM | **0.79** | $\varepsilon = 0.005$, $\mathcal{F} = \{5, 19, 21, 22\}$ |
| LB | Pfinder | 0.30 | $T = 10$, $\alpha = 0.01$, $N = 20$ |
| | GrimsonGMM | 0.55 | $\mathcal{T}_{\sigma^\in} = 9$, $\alpha = 0.0025$, $K = 5$ |
| | KaewGMM | 0.20 | $\alpha = 0.01$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.60 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.46 | $\epsilon_0 = 200$, $\epsilon_1 = 80$, $\alpha_0 = 240$, $\alpha_1 = 36$, $N = 83$ |
| | MFBM | **0.61** | $\varepsilon = 0.0001$, $\mathcal{F} = \{3, 20, 21, 22\}$ |
| OS | Pfinder | 0.71 | $T = 16$, $\alpha = 0.002$, $N = 20$ |
| | GrimsonGMM | 0.62 | $T_{\sigma^2} = 12$, $\alpha = 0.0025$, $K = 3$ |
| | KaewGMM | 0.72 | $\alpha = 0.004$, $N = 100$, $K = 4$, $T = 0.6$ |
| | ZivkovicGMM | 0.55 | $\alpha = 0.001$, $N = 400$, $T_{\sigma^2} = 20$ |
| | FASOM | 0.65 | $\epsilon_0 = 220$, $\epsilon_1 = 100$, $\alpha_0 = 240$, $\alpha_1 = 40$, $N = 79$ |
| | MFBM | **0.77** | $\varepsilon = 0.005$, $\mathcal{F} = \{3, 20, 21, 22\}$ |

**Cuadro 5.7:** Mejores configuraciones en términos de exactitud. La primera columna indica el nombre de la secuencia y la segunda el método. La tercera columna indica el rendimiento obtenido por la mejor configuración de cada método para cada secuencia. En la última columna se muestra la configuración que obtiene dicho rendimiento. La exactitud del método que logra mejor resultado en cada secuencia se ha marcado en negrita.

| Secuencia | Tamaño | Pfinder | Grimson GMM | Kaew GMM | Zivkovik GMM | FASOM | MFBM |
|---|---|---|---|---|---|---|---|
| WS | 160 x 128 | 1041.63 | 431.91 | 1637.14 | *3821.61* | 255.85 | 65.20 |
| MR | 160 x 128 | 1017.13 | 374.66 | 1655.48 | *3720.99* | 254.31 | 94.97 |
| LC | 360 x 240 | 257.65 | 96.84 | 416.57 | *1232.40* | 57.86 | 25.03 |
| V2 | 384 x 240 | 241.53 | 99.71 | 406.19 | *1154.81* | 54.18 | 20.48 |
| LB | 160 x 128 | 1039.14 | 300.62 | 1764.97 | *3301.33* | 260.87 | 77.90 |
| OS | 384 x 288 | 200.95 | 59.25 | 335.64 | *866.02* | 48.34 | 19.59 |

**Cuadro 5.8:** Fotogramas por segundo (FPS, mejor cuanto más alto) para cada secuencia analizada. La segunda columna muestra el tamaño del fotograma. Cuanto mayor sea el fotograma, menor ratio FPS. Los valores mostrados corresponden a la configuración que obtiene mejor rendimiento en términos de exactitud. El método con mayor FPS en cada secuencia se ha marcado en cursiva.

| Método | Exactitud | F-medida |
|---|---|---|
| MFBM | **0.78*** | **0.86*** |
| Pfinder | 0.67 | 0.74 |
| GrimsonGMM | 0.70 | 0.78 |
| KaewGMM | 0.57 | 0.66 |
| ZivkovicGMM | 0.70 | 0.78 |
| FASOM | 0.72 | 0.78 |

**Cuadro 5.9:** Resultados medios de las cinco mejores configuraciones de cada secuencia y método. Se ha marcado en negrita el método que mejor rendimiento medio obtiene. El asterisco indica que la diferencia respecto del segundo mejor método es significativa con un nivel 99 % de confianza.

### 5.4.7. Curvas ROC

Para comparar los diferentes conjuntos de rasgos es útil emplear las curvas ROC (*Receiver Operating Characteristic*). De este modo se aprecia cómo evoluciona el rendimiento de los métodos al variar el valor del umbral de probabilidad $\rho$ (ver ecuación 5.7) en términos de TPR vs FPR.

La Figura 5.14 muestra las curvas ROC de los cinco mejores conjuntos de rasgos. Cuanto más cerca se esté del punto $(0, 1)$, mejor rendimiento tiene el método. Como podemos apreciar, el conjunto de rasgos $\{3, 20, 21, 22\}$ es la mejor alternativa, incluso mejor que utilizar cinco rasgos. El resto de conjuntos de rasgos obtienen rendimientos similares. También se observa que en general al utilizar más rasgos, el rendimiento mejora ligeramente.

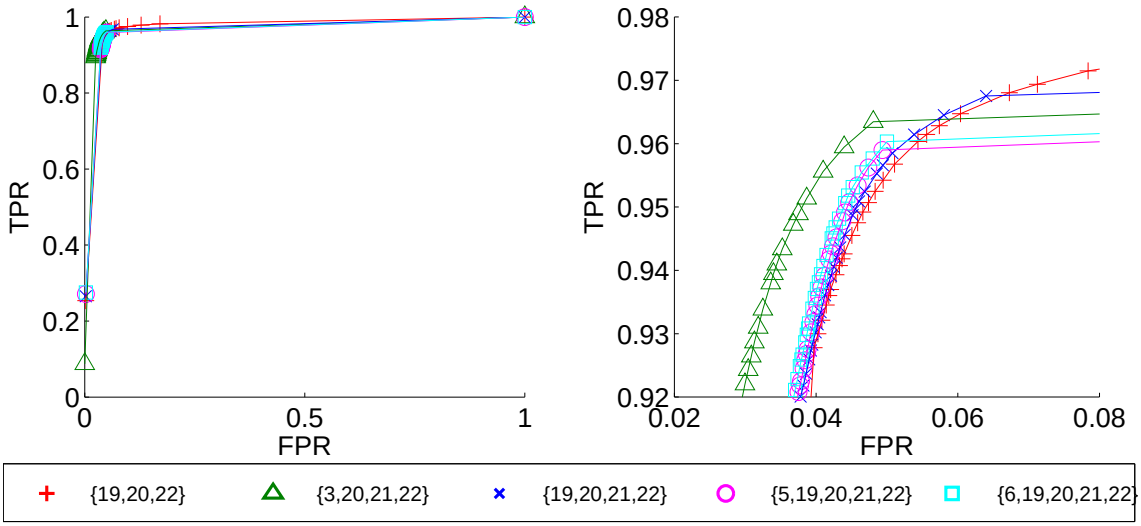



**Figura 5.14:** Curvas ROC. El gráfico de la izquierda muestra la curva ROC de cada conjunto de rasgos basándose en el valor de $\rho$, el gráfico de la derecha muestra un detalle todas ellas. Hemos utilizado $\varepsilon = 0.0001$ para obtener estos resultados.

### 5.4.8. Funcionamiento en condiciones difíciles

Esta subsección está dedicada a estudiar el funcionamiento de los diferentes métodos en secuencias grabadas bajo condiciones particularmente difíciles y así tener una visión más amplia del rendimiento de nuestra propuesta. Para este fin hemos elegido un conjunto de vídeos adicionales que ejemplifican algunas de estas situaciones, tales como nieve, niebla, viento, iluminación heterogénea, objetos a gran velocidad y temblor de la cámara.

Estas pruebas han sido llevadas a cabo utilizando una única configuración para cada método, en concreto aquella que logra el mejor rendimiento medio en las seis secuencias estándar previamente estudiadas (ver Figura 5.9). La única excepción es FASOM en las secuencias BL y SF donde hemos usados directamente los resultados que nos han facilitado los autores del método. El Cuadro 5.10 detalla qué parámetros se han usado en cada método. Para las secuencias BL y SF se ha usado un valor preestablecido $\sigma^2 = 10^{-4}$ en lugar del procedimiento habitual de estimación de ruido mostrado en la Subsección 5.2.2. El motivo es la compresión a la que han sido sometidas estas dos secuencias, la cual influye de forma diferente en estos casos ya que existe muy poco contraste en la escena.

La Figura 5.15 ilustra la salida de cada método en los vídeos seleccionados de ChangeDetection.net. Como vemos en la primer fila, es una secuencia donde los métodos tienden a sufrir problemas de camuflaje, sin embargo nuestra propuesta detecta todos los objetos del primer plano, produciendo menos falsos negativos que los competidores. La segunda fila presenta una situación donde Pfinder y ZivkovikGMM logran buenos resultados, pero MFBM es aún mejor que Pfinder en número de falsos nega-

| Método | Parámetros |
|---|---|
| Pfinder | $\mathcal{N} = 10$, $\alpha = 0.0001$, $\ell = 20$ |
| GrimsonGMM | $\mathcal{N} = 12$, $\alpha = 0.0025$, $\mathcal{K} = 3$ |
| KaewGMM | $\alpha = 0.004$, $\mathcal{H} = 100$, $\mathcal{K} = 4$, $\mathcal{N} = 0.6$ |
| ZivkovikGMM | $\alpha = 0.001$, $\mathcal{H} = 400$, $\mathcal{N} = 20$ |
| FASOM | $s_1 = 90$, $s_0 = 240$, $\alpha_1 = 38$, $\alpha_0 = 240$,$\varepsilon = 81$ |
| MFBM | $\varepsilon = 0.0001$, $\mathcal{F} = \{3, 20, 21, 22\}$ |

**Cuadro 5.10:** Parámetros utilizados para el conjunto de secuencias con condiciones difíciles. Corresponden a las configuraciones del frente de Pareto mostradas en la Figura 5.3.

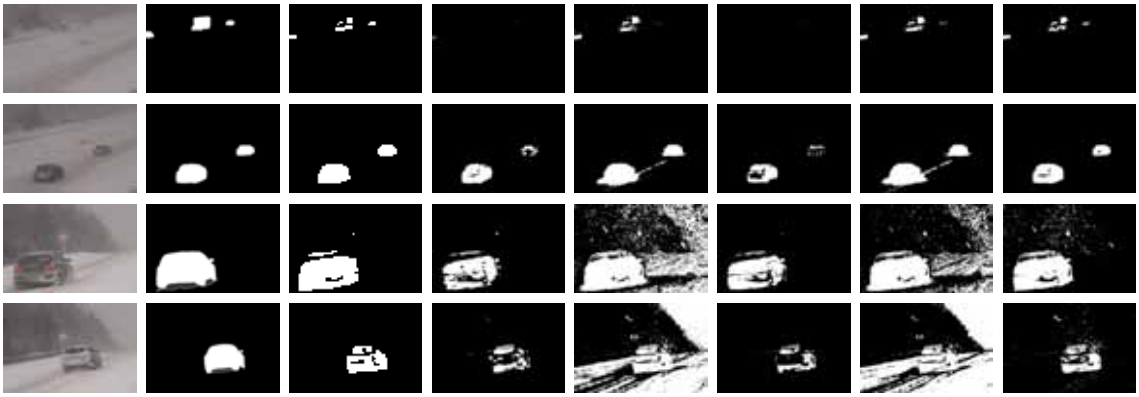



**Figura 5.15:** Resultados experimentales en los vídeos de ChangeDetection.net. Las dos primeras filas corresponden a los fotogramas 1500 y 3150 de la secuencia blizzard (BL). La tercera y cuarta filas a los fotogramas 810 y 2785 de snowFall (SF).

tivos y mejor que ZivkovikGMM en falsos positivos. Las filas 3 y 4 (secuencia SF) muestran situaciones donde está nevando o bien ha nevado hace poco. Esto causa falsos positivos en ZivkovikGMM, GrimsonGMM, Pfinder y en menor grado en MFBM. Además, nuestra propuesta comete menos falsos negativos que FASOM y KaewGMM, los cuales parecen ser robustos ante este tipo de situaciones con nieve, por tanto MFBM es nuevamente la mejor opción.

Por otro lado, la Figura 5.16 muestra los resultados de los vídeos del repositorio BMC. En TL la iluminación causa diversos problemas de sombra proyectada. Como puede apreciarse, MFBM es el método que mejor funciona en este aspecto. Únicamente KaewGMM es capaz de obtener un rendimiento similar en la primer fila de la figura, pero falla en la segunda. La secuencia WD (tercera y cuarta filas) es compleja debido a varios factores: el fondo es muy dinámico a causa del viento y del movimiento de las nubes, que además producen problemas de sombra proyectada en la carretera. Asimismo, se producen vibraciones de la cámara, lo que supone un reto adicional de cara a la segmentación. Todas estas cuestiones producen en general que los métodos estudiados obtengan una cantidad muy elevada de falsos positivos. La única excepción es KaewGMM, pero su problema es justo el contrario, padece demasiados falsos negativos. Por todo ello nuestra propuesta emerge como una opción interesante ya que produce una cantidad reducida de falsos negativos y menos falsos positivos que la mayoría de los competidores.

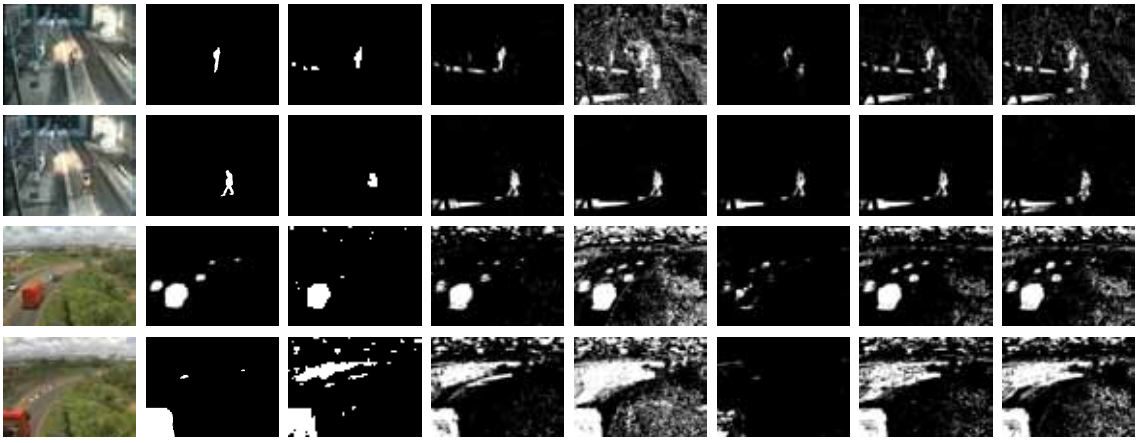



**Figura 5.16:** Resultados experimentales en los vídeos de BMC 2012. Las dos primeras filas corresponden a los fotogramas 120 y 1363 de la secuencia Tunnel (TL). La tercera y cuarta filas a los fotogramas 127 y 208 de Windy day (WD).

Para conseguir una visión más ajustada del rendimiento de cada método hemos elaborado el Cuadro 5.11, donde se muestra el rendimiento exacto de cada método en estas secuencias particularmente difíciles. Los resultados de todos los métodos son pobres respecto de los presentados en la Sección 5.4. Como ya se ha mencionado,

| Secuencia | Método | Exactitud | F-medida |
|---|---|---|---|
| BL | Pfinder | 0.35 | 0.42 |
| | GrimsonGMM | 0.38 | 0.46 |
| | KaewGMM | 0.14 | 0.19 |
| | ZivkovikGMM | 0.39 | 0.46 |
| | FASOM | 0.19 | 0.24 |
| | MFBM | **0.40** | **0.46** |
| SF | Pfinder | 0.15 | 0.19 |
| | GrimsonGMM | 0.14 | 0.18 |
| | KaewGMM | 0.08 | 0.11 |
| | ZivkovikGMM | 0.13 | 0.16 |
| | FASOM | 0.11 | 0.16 |
| | MFBM | **0.16** | **0.20** |
| TL | Pfinder | 0.18 | 0.27 |
| | GrimsonGMM | 0.28 | 0.40 |
| | KaewGMM | 0.31 | 0.44 |
| | ZivkovikGMM | 0.22 | 0.34 |
| | FASOM | 0.28 | 0.41 |
| | MFBM | **0.31** | **0.44** |
| WD | Pfinder | 0.09 | 0.15 |
| | GrimsonGMM | 0.16 | 0.26 |
| | KaewGMM | **0.22** | **0.35** |
| | ZivkovikGMM | 0.09 | 0.16 |
| | FASOM | 0.13 | 0.21 |
| | MFBM | 0.15 | 0.23 |

**Cuadro 5.11:** Resultados de las mejores configuraciones de acuerdo a exactitud y F-medida en las secuencias con condiciones difíciles. La primera columna denota el nombre de la secuencia, mientras que la segunda indica el método. La exactitud y F-medida de cada método se muestra en la tercera y cuarta columna, respectivamente. Los mejores resultados de cada secuencia se han marcado en negrita.

| Método | Blizzard | SnowFall | Skating | WetSnow | Media |
|---|---|---|---|---|---|
| Pfinder | N/A | N/A | N/A | N/A | N/A |
| GrimsonGMM | 0.74 | 0.73 | 0.88 | **0.61** | 0.74 |
| KaewGMM | N/A | N/A | N/A | N/A | N/A |
| ZivkovikGMM | 0.76 | **0.76** | 0.86 | 0.58 | 0.74 |
| FASOM | 0.62 | 0.67 | 0.76 | 0.50 | 0.64 |
| MFBM | **0.83** | 0.75 | **0.89** | 0.57 | **0.76** |

**Cuadro 5.12:** Resultados en términos de F-medida' para la categoría *BadWeather* de ChangeDetecion.net. La primera columna denota el nombre del método. De la segunda a la quinta columna se muestra el rendimiento de cada método según F-medida'. La última columna indica el rendimiento medio en la categoría *BadWeather* al completo. Nótese que los resultados de Pfinder y KaewGMM no aparecen ya que no están disponibles en el sitio web Changedetection.net, por tanto se ha marcado como N/A. Los mejores resultados para cada secuencia se han destacado en negrita.

| Método | TL | WD |
|---|---|---|
| SAM | 0.754 | 0.608 |
| VC | 0.8 | 0.5 |
| 3dSOBS+ | 0.808 | 0.752 |
| Spampinato | 0.688 | 0.573 |
| SLDP | 0.785 | 0.827 |
| StSIC | 0.741 | **0.836** |
| MFBM | **0.810** | 0.611 |

**Cuadro 5.13:** Resultados en términos de F-medida en las dos secuencias de BMC 2012 para las otras propuestas adicionales publicadas en la *Special Section on Background Models Comparison of CVIU Journal* (Mayo 2014). La primera columna indica el nombre del método. La segunda y tercera columnas muestran el rendimiento obtenido por cada método en las secuencias TL y WD, respectivamente. Se han marcado en negrita los mejores resultados para cada secuencia.

esto se debe a las condiciones extremas en las que se han grabado estas secuencias, lo cual produce artefactos tanto de ruido como de compresión. Como vemos nuestra propuesta es la mejor alternativa en las secuencias BL y SF. MFBM también es el mejor método en TL, aunque cabe destacar que KaewGMM logra resultados similares. En el vídeo WD, KaewGMM es el que mejor segmentación realiza, mientras que nuestra propuesta queda en tercer lugar. A modo de resumen, KaewGMM es el mejor competidor ya que segmenta adecuadamente las dos últimas secuencias, sin embargo los resultados son muy pobres en BL y SF, mientras que nuestro método sólo queda por detrás en la última secuencia, siendo el mejor en las otras tres. Por lo que podemos concluir que MFBM es un buen método en este tipo de secuencias.

Adicionalmente hemos añadido los Cuadros 5.12 y 5.13. El Cuadro 5.12 muestra la F-medida' obtenida por nuestra propuesta en las cuatro secuencias que componen la categoría *BadWeather* del repositorio ChangeDetection.net, utilizando la configuración $\varepsilon = 0.005$, $\mathcal{F} = \{4, 19, 20, 21\}$, que optimiza el rendimiento en términos de F-medida'. Este cuadro también muestra los resultados de tres de los competidores que están publicados en el sitio web ChangeDetection.net. Nótese que hemos usado F-medida' en lugar de la F-medida habitual porque este sitio web utiliza F-medida'. Como vemos, MFBM logra los mejores resultados medios en toda la categoría. En particular, MFBM es la mejor alternativa en dos de los cuatro vídeos y obtiene resultados competitivos en el resto. Por último, el Cuadro 5.13 compara nuestra propuesta, utilizando la configuración $\varepsilon = 0.005$, $\mathcal{F} = \{4, 19, 20, 21\}$, con otros métodos publicados en la sección especial sobre comparación de modelos de fondo de la revista *Computer Vision and Image Understanding* (Mayo 2014). Como vemos, MFBM es el mejor método en TL y una alternativa competitiva en WD. Los métodos considerados para esta comparativa son los siguientes:

- SAM [206]: Proponen un modelo de mixtura gaussiana auto-adaptativo que trabaja tanto a nivel de píxel como de fotograma y que se adapta dinámicamente a los cambios de iluminación.
- VC [207]: Este artículo presenta el modelo denominado Visual Cortex, el cual está inspirado biológicamente en el cortex visual.
- 3dSOBS+ [208]: También es un método biológicamente inspirado que utiliza un modelo de fondo neuronal generado automáticamente mediante un método autoorganizado.
- Spampinato [209]: Proponen un método para modelar el fondo y el primer plano basado en el color y en rasgos de las texturas.
- SLDP y StSIC [210]: En este trabajo se definen dos modelos: SLDP y StSIC; el primero de ellos es un modelo estadístico y el segundo se basada en rasgos locales.

## 5.5. Discusión

Gracias al estudio llevado a cabo en este capítulo, vamos a discutir los aspectos más relevantes de la propuesta planteada. En primer lugar hablaremos de lo que han puesto de relieve las pruebas realizadas respecto del uso de rasgos y después, incidiremos sobre las novedades de nuestra propuesta respecto de lo publicado en [159].

Algunos rasgos que han demostrado ser adecuados para la detección del primer plano mediante estimación basada en funciones núcleo, tales como los filtros tipo Haar [199], no son adecuados para las aproximaciones basadas en mixturas probabilísticas. Los filtros tipo Haar son rasgos débiles, por lo que se necesita una cantidad grande de ellos para obtener un clasificador fuerte. En nuestro caso necesitamos una cantidad pequeña de rasgos que tengan un poder de discriminación alto.

El rendimiento en la detección de los modelos basados en mixturas probabilísticas aumenta conforme añadimos más rasgos relevantes, pero la mejoría decrece hasta un punto donde añadir más rasgos es inútil. Según nuestros experimentos, hemos determinado que nuestra propuesta alcanza dicho punto cuando se emplean $D = 5$ rasgos.

Los canales de color normalizados y los rasgos basados en el filtro mediana logran resultados particularmente buenos. Esto se debe a su robustez frente a los cambios de iluminación (derivado de la normalización de los rasgos) y su capacidad para filtrar el ruido de fondo (derivado de la mediana). Y lo que es más, los efectos positivos se maximizan en los rasgos normalizados basados en el filtro mediana.

Nuestra propuesta supera a los métodos de segmentación actuales que hemos probado. Es posible que estos métodos competidores se puedan mejorar adaptándolos para que hagan uso de una cantidad no fija de rasgos, tal y como hace nuestra propuesta.

La búsqueda de rasgos relevantes para el modelado del fondo mediante mixturas probabilísticas es un campo de investigación abierto. Se debe tratar de encontrar rasgos más discriminatorios y que no requieran mucho tiempo de cómputo.

Existen cuatro novedades de la propuesta planteada en este capítulo respecto de lo publicado en [159]. En primer lugar el nuevo modelo admite cualquier número y tipo de rasgos para los píxeles (Subsección 5.2.1). La segunda novedad consiste en que la componente uniforme de la mixtura que modela los objetos del primer plano ha sido modificada para admitir rasgos con cualquier rango de valores posibles (ecuaciones 5.4-5.5). La tercera novedad es que se ha incluido un umbral de probabilidad $\rho$ que permite al usuario ajustar la sensibilidad y especificidad de la detección del primer plano. Por último, la implementación del método ha sido rehecha completamente debido a que la implementación de [159] no escala más de 3 rasgos. En concreto, la factorización de Cholesky de la matriz de covarianzas se usa para calcular el determinante de la matriz de covarianzas y la distancia de Mahalanobis del vector

de rasgos del píxel de entrada (Subsección 5.2.3). Este procedimiento es más rápido y numéricamente estable porque la inversa de la matriz de covarianzas nunca se calcula explícitamente.

## 5.6. Conclusiones

Se ha propuesto un nuevo modelo para la detección de objetos del primer plano. Es capaz de emplear cualquier cantidad y tipo de rasgos para los píxeles. Hemos propuesto además un conjunto de rasgos relevantes de acuerdo con varias propiedades. Se ha averiguado que los rasgos normalizados y basados en el filtro mediana superan a las componentes RGB estándar. Por otro lado, algunos rasgos que habían demostrado su buen funcionamiento en propuestas basadas en KDE, no sirven para nuestro método. El rendimiento de nuestra propuesta ha sido comparado con otras alternativas del estado del arte en un conjunto de vídeos de prueba y los resultados han sido favorables. Es previsible que otros métodos se puedan mejorar modificándolos para que puedan admitir cualquier tipo y número de rasgos.

# 6 Cambios de iluminación

Los cambios de iluminación son eventos que suceden habitualmente en gran cantidad de escenas reales. Entre los casos más típicos está apagar o encender la luz, aunque también hay otros más sutiles como el oscurecimiento de la escena que produce el paso de una nube, o también el cambio de iluminación debido al cambio del día a la noche y viceversa. Este tipo de situaciones constituyen un problema fundamental para los métodos de detección de movimiento ya que suele ser difícil diferenciar un cambio de color debido al movimiento de un objeto del primer plano, de uno producido por un cambio de iluminación. La mayoría de las estrategias para solventarlo consisten en determinar la manera en la que se transforma el color del píxel cuando se produce un cambio de iluminación. En este capítulo exponemos una propuesta que no asume ninguna forma específica para dicha transformación. Se basa en una evaluación cuantitativa de la suavidad de la transformación del color local entre el fotograma actual y el modelo de fondo. Adicionalmente, llevamos a cabo una evaluación de los estados de iluminación de los píxeles con ayuda de lógica difusa. Para demostrar el buen funcionamiento de nuestra propuesta en diferentes situaciones, mostramos los resultados obtenidos en un amplio conjunto de experimentos.

El capítulo ha sido estructurado de la siguiente manera. La Sección 6.1 presenta un estudio del estado del arte en el ámbito de la segmentación de secuencias con cambios de iluminación. El método propuesto para la detección de cambios de iluminación se expone en la Sección 6.2. La Sección 6.3 muestra los resultados experimentales que avalan la capacidad de nuestra propuesta para gestionar escenas complejas. Sus características y propiedades más destacadas se discuten en la Sección 6.4. Para terminar, la Sección 6.5 está dedicada a las conclusiones.

## 6.1. Introducción

La mayoría de los métodos de detección del primer plano crean un modelo de fondo que se actualiza progresivamente conforme pasa el tiempo. De este modo, los cambios de iluminación se convierten en un reto, ya que la apariencia del fondo no encaja con las observaciones pasadas. Este hecho ha sido estudiado durante bastante tiempo [211, 212, 213, 214] debido a lo relevante que es para las aplicaciones de videovigilancia. Las cámaras de videovigilancia suelen estar en entornos donde las condiciones de iluminación varían a lo largo del tiempo. Por ejemplo, en las escenas de interior es normal que la luz se apague o se encienda, mientras que en las esce-

nas de exterior el paso de las nubes puede afectar el área que graba la cámara (ver Figura 6.1).

**Figura 6.1:** Ejemplos de cambios de iluminación en una misma secuencia. Cada fila muestra dos fotogramas de una misma secuencia. La primera columna representa un fotograma con mayor iluminación que el de la segunda columna. La fila superior corresponde a una escena de interior donde el cambio de luz se debe al encendido/apagado de la luz, mientras que en la segunda fila el motivo del cambio de iluminación es el paso de una nube.

Básicamente existen dos formas de abordar este problema. Por un lado, los algoritmos que analizan la escena a nivel de píxel y, por tanto, toman una decisión independiente para cada píxel. Y por otro lado, los algoritmos que trabajan a un nivel mayor, es decir, a nivel de bloque, región o fotograma; en estos casos las decisiones tienen en cuenta una cantidad mayor de píxeles. En otras ocasiones, una tercera opción consiste en aplicar un enfoque mixto, es decir, métodos de segmentación que analizan simultáneamente a varios niveles. Un caso representativo de esto último sería el sistema *Wallflower* [122], que realiza un procesamiento a nivel de píxel, bloque y fotograma de manera simultánea. Una de las desventajas atribuidas a este método consiste en que no utiliza un criterio flexible en las situaciones reales, ya que elige un conjunto de modelos de fondo que representan diferentes situaciones reales y cada fotograma de entrada debe ser asignado al modelo que produce menos píxeles del primer plano. Esto implica que cada posible situación debe ser predicha con antelación y además, se debe encontrar un modelo de fondo representativo de cada situación. Por lo tanto, este enfoque es menos adecuado para escenas donde suceden eventos impredecibles que afectan al fondo, por ejemplo, un objeto abandonado o un vehículo estacionado que comienza a moverse.

En cuanto a los métodos que analizan la escena a nivel de bloque o región, podemos citar el método propuesto en [215]. En particular esta propuesta analiza la imagen a nivel de bloque y también a nivel píxel. La idea fundamental consiste en que cada píxel pertenece a varios bloques solapados, de este modo se determina si pertenece o no al fondo en función de cómo sea clasificado por cada uno de estos bloques.

Un enfoque más reciente es el algoritmo propuesto en [216] que procesa la escena a nivel de píxel y de fotograma. La propuesta consiste en usar una arquitectura de dos capas basadas en un modelo de mixtura gaussiana para representar el fondo. El resultado es optimizado empleando un marco de trabajo basado en los campos aleatorios de Markov.

Existen muchos algoritmos de segmentación a nivel de píxel. Por ejemplo, [217] se basa en la aplicación de un filtro homomórfico, mientras que [218] realiza un análisis con visión estereoscópica y utiliza un modelo de disparidad generado previamente para mitigar el consumo extra de tiempo de CPU requerido para este tipo de procesamiento. En el caso de [144] se usan rasgos de textura discriminatorios para generar estadísticas del fondo mediante operadores de textura como LBP (*local binary patterns*). Por último, en [211] se presenta un algoritmo adaptativo que usa múltiples subespacios de rasgos y análisis de componentes principales para capturar y aprender diferentes condiciones de iluminación.

El objetivo de este capítulo consiste en desarrollar un sistema de detección de cambios de iluminación que trabaje conjuntamente con un algoritmo de detección de movimiento existente. En [219] ya se propuso un complemento para la detección de cambios de iluminación. Sin embargo, a diferencia de nuestra propuesta, ellos asumen que cuando se produce un cambio de iluminación, el orden de los valores de los píxeles se mantiene en los entornos locales, fundamentando su propuesta en el análisis de propiedades físicas, como por ejemplo la radiancia. En cambio, nosotros no estamos interesados en la forma concreta de la transformación del color, nos centramos en sus propiedades de suavidad. En este sentido, consideramos que cualquier transformación de color que no sea suave, no puede corresponder a un cambio de iluminación y por lo tanto se debe a un objeto del primer plano. Así pues, somos capaces de detectar variaciones en la iluminación independientemente de las características particulares de la transformación del color que se haya producido. Es más, el procedimiento propuesto es prácticamente independiente del modelo de fondo que se use, por lo que puede utilizarse para mejorar el rendimiento de un gran número de métodos actuales que no están diseñados específicamente para funcionar en secuencias con cambios de iluminación.

## 6.2. Descripción del método

El procedimiento que proponemos para gestionar los cambios de iluminación consta de dos partes. La primera de ellas clasifica los píxeles del fotograma de entrada de

acuerdo con su estado actual respecto de los cambios de iluminación (ver Subsección 6.2.1). La segunda parte utiliza esta información para decidir qué píxeles deben ser reiniciados debido a que un cambio de iluminación ha hecho que sus modelos de fondo estén desactualizados (ver Subsección 6.2.2).

### 6.2.1. Estimación del tipo de iluminación

Aquí debemos estimar el estado de iluminación de cada uno de los píxeles que componen el fotograma de entrada actual. El estado de iluminación del píxel situado en la posición $p$ se forma mediante tres variables difusas: $Brusquedad$, $Diferencia$, y $Base$. El valor (grado de pertenencia) de cada una de esta variables lo denotaremos mediante $\alpha_p, \beta_p, \gamma_p \in [0, 1]$, respectivamente. La interpretación de cada una de estas variables es la siguiente:

- $Brusquedad$: indica si la transformación de los colores del fotograma previo a los colores del fotograma actual no ha sido suave en el entorno local del píxel $p$. Si $\alpha_p$ tiene un valor elevado, entonces es improbable que se haya producido un cambio de iluminación, porque los cambios de iluminación producen cambios suaves de los colores tanto del fondo como de los objetos del primer plano.
- $Diferencia$: indica si el color del píxel $p$ en el fotograma actual es muy diferente del que hay almacenado el modelo de fondo para dicho píxel. Así pues, si $\beta_p$ es alto, entonces se ha producido un cambio de iluminación o bien existen un objeto de fondo en esa posición.
- $Base$: indica si el modelo de fondo base ha detectado un objeto del primer plano en el píxel $p$. Nótese que $\gamma_p \in \{0, 1\}$ en aquellos modelos que no proporcionan un grado de confianza para la detección de movimiento.

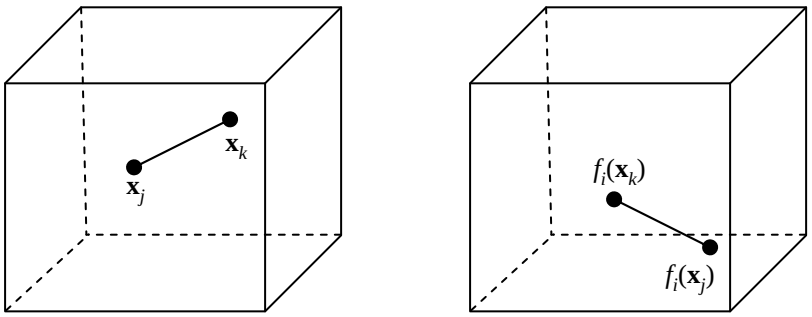
A continuación vamos a describir cómo calculamos el valor difuso de las variables $\alpha_p$ y $\beta_p$; ya que el valor de $\gamma_p$ es directamente la salida del modelo base.

Centremos nuestra atención en el entorno local $W_p$ del píxel $p$. En nuestros experimentos hemos considerado una ventana cuadrada de tamaño $5 \times 5$ píxeles, lo que ofrece un buen equilibrio entre eficiencia y exactitud. A continuación definimos un campo vectorial que representa la transformación de los colores en el entorno $W_p$ del modelo de fondo a los colores en el entorno $W_p$ en el fotograma actual:

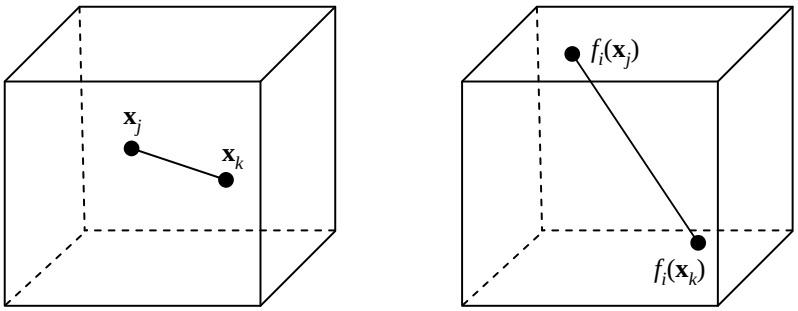
$$f_p \ : \ \mathbb{R}^3 \rightarrow \mathbb{R}^3 \tag{6.1}$$

donde se asume que el color se representa con tres componentes, pero no se fija ningún espacio de color en concreto. Nuestra tarea es detectar aquellos campos vectoriales $f_p$ que no son suaves. Una manera simple y rápida de medir la suavidad de $f_p$ consiste en el cálculo de los siguientes cocientes:

$$q_p\left(j,k\right) = \frac{a_p\left(j,k\right)}{b_p\left(j,k\right)} \tag{6.2}$$




**(a)**




**(b)**

**Figura 6.2:** Transformaciones del color entre el modelo de fondo y el fotograma actual: (a) suave, (b) brusco. Cada cubo representa el espacio de color, donde cada dimensión es una componente de color.

$$a_p(j,k) = \|f_p(\boldsymbol{j}) - f_p(\boldsymbol{k})\|^2 \tag{6.3}$$

$$b_p(j,k) = \|\boldsymbol{j} - \boldsymbol{k}\|^2 \tag{6.4}$$

donde $\boldsymbol{j}$, $\boldsymbol{k}$ son los colores en el modelo de fondo de dos píxeles $j, k \in W_p$, y $f_p(\boldsymbol{j})$, $f_p(\boldsymbol{k})$ son los colores de dichos píxeles en el fotograma actual. El campo vectorial $f_p$ no es suave si $q_p$ alcanza valores altos para algún par $j, k \in W_p$. Como se indicó en la Figura 6.2, una transformación suave es aquella que asigna colores similares en el fotograma actual a aquellos colores que son similares en el modelo de fondo, incluso si los colores de un píxel cambian considerablemente entre el modelo de fondo y el fotograma actual. Es decir, las distancias $\|\boldsymbol{j} - f_p(\boldsymbol{j})\|$ son irrelevantes para la suavidad de $f_p$. De este modo podemos gestionar el encendido y apagado de cualquier tipo de luz independientemente de su color, por ejemplo, luces amarillas o blancas. Debe tenerse en cuenta que los valores bajos de $q_p$ no son interesantes porque

habitualmente se asocian con objetos del primer plano homogéneos que pasan por delante de fondos texturizados, lo cual se gestiona de forma adecuada por la mayoría de los algoritmos de detección estándar.

En la práctica, el ruido en los píxeles de la imagen puede producir grandes errores en la estimación de $q_p$, en concreto si el denominador $b_p$ contiene ruido. Hemos atenuado este efecto considerando una cantidad filtrada $\phi_p$:

$$\phi_p(j,k) = \begin{cases} \frac{a_p(j,k)}{B_{inf}} & b_p(j,k) < B_{inf} \\ 0 & b_p(j,k) > B_{sup} \\ q_p(j,k) & en\,otro\,caso \end{cases} \tag{6.5}$$

donde $B_{inf}$ y $B_{sup}$ son, respectivamente, un umbral inferior y superior adecuados para el valor del denominador $b_p$ y $B_{inf} < B_{sup}$. Nótese que $B_{inf}$ gestiona las condiciones de poca iluminación, donde los valores de un píxel tienen poca precisión. Por otro lado, $B_{sup}$ garantiza que los fondo altamente texturizados (que se asocian a valores altos de $a_p$ y $b_p$) no se confundan como transformaciones no suaves de color. Finalmente utilizamos el mayor $\phi_p(j,k)$ como medida de la brusquedad de $f_p$:

$$\alpha_p = \text{mín}\left(1, \frac{1}{K_\alpha} \max_{j,k \in W_p} \phi_p(j,k)\right) \tag{6.6}$$

donde la función mín garantiza que $\alpha_p \in [0,1]$ y $K_\alpha$ es un parámetro de escala.

Por otro lado, el valor de la segunda variable difusa, $\beta_p$, se obtiene calculando la distancia euclídea al cuadrado entre el color del píxel $p$ en el fotograma actual y el color almacenado en el modelo de fondo para dicho píxel:

$$\beta_p = \frac{1}{K_\beta} \|\boldsymbol{B}(p) - \boldsymbol{p}\|^2 \tag{6.7}$$

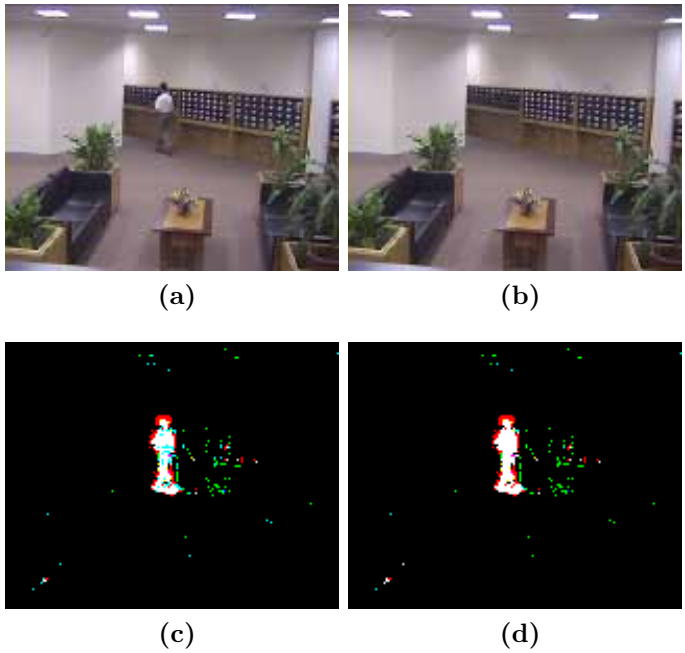
donde $\boldsymbol{B}(p)$ y $\boldsymbol{p}$ son los colores del píxel en cuestión en el modelo de fondo y en el fotograma actual, respectivamente; mientras que $K_\beta$ es otro factor de escala.

Los valores posibles de las tres variables difusas son asignados a ocho estados posibles de iluminación para el píxel $p$ de acuerdo al Cuadro 6.1. Se usa el conjunto de operaciones difusas estándar para calcular el grado de pertenencia a los ocho estados según $\alpha_p, \beta_p, \gamma_p$. Después se declara la pertenencia del píxel al estado con mayor grado de pertenencia.

Los píxeles en estado dudoso se deben corregir y asignarse a "Objeto del primer plano" o bien a "Fondo con cambio de iluminación". De este modo, dado un píxel con una transformación suave del color, una diferencia importante respecto del color almacenado en el modelo de fondo y que es clasificado como primer plano por el algoritmo base, no podemos decir si se trata de un objeto del primer plano o un píxel del fondo que ha experimentado un cambio de iluminación. A nivel de píxel no

| *Brusquedad* | *Diferencia* | *Base* | Pseudo-color | Descripción del estado |
|---|---|---|---|---|
| Falso | Falso | Falso | Negro | Fondo inactivo |
| Falso | Falso | Verdadero | Azul | Error en el modelo de fondo (habitualmente corresponde al fondo) |
| Falso | Verdadero | Falso | Verde | Fondo con cambio de iluminación |
| Falso | Verdadero | Verdadero | Cían | Dudoso (debe ser corregido) |
| Verdadero | Falso | Falso | Rojo | Fondo cerca de un objeto del primer plano |
| Verdadero | Falso | Verdadero | Magenta | Suceso poco frecuente (puede asignarse al primer plano) |
| Verdadero | Verdadero | Falso | Amarillo | Error en el modelo de fondo (habitualmente corresponde al primer plano) |
| Verdadero | Verdadero | Verdadero | Blanco | Objeto del primer plano |

**Cuadro 6.1:** Estados de iluminación posibles para un píxel.



**(a)** **(b)**

**(c)** **(d)**

**Figura 6.3:** Estimación del estado de iluminación: (a) fotograma actual, (b) modelo de fondo, (c) estados de iluminación antes de la corrección de píxeles dudosos, (d) estados de iluminación después de la corrección de píxeles dudosos.

se puede resolver esta ambigüedad. Por lo tanto, proponemos un procedimiento a nivel de fotograma que realice esta tarea. Para cada píxel $p$ dudoso trazamos líneas en ocho direcciones principales: arriba, abajo, izquierda, derecha y en cada una de las cuatro diagonales. Si alguna de estas líneas encuentra un píxel en alguno de los estados "Objeto del primer plano" o "Fondo con cambio de iluminación", contamos un voto para dicho estado. Las líneas que salgan del fotograma sin haber encontrado ningún píxel en alguno de estos estados, no suman ningún voto. Finalmente, el estado del píxel $p$ será aquel que cuente con más votos. Si existe un empate, no se corrige el píxel. La lógica que subyace en este procedimiento consiste en que un píxel dudoso que pertenece al primer plano habitualmente se encuentra en el interior de un objeto. Así pues, en estos píxeles la mayoría de las líneas encontrará algún píxel en el estado "Objeto del primer plano", al menos los que están en el borde del objeto. Debe tenerse en cuenta que los píxeles dudosos son más frecuentes en los objetos con un interior homogéneo. Por otro lado, los píxeles dudosos que pertenecen al fondo habitualmente están rodeados de píxeles en el estado "Fondo con cambio de iluminación".

La Figura 6.3 ejemplifica el procedimiento de estimación de estado de iluminación descrito en esta subsección. Nótese que los estados están representados en los pseudo-colores especificados en el Cuadro 6.1. Como puede observarse, los píxeles dudosos que hay dentro de la persona son corregidos al estado "Objeto del primer plano", mientras que los que caen en la sombra proyectada son corregidos al estado "Fondo con cambio de iluminación". Adicionalmente, la Figura 6.4 muestra el diagrama de flujo de nuestra propuesta para el fotograma k-ésimo.

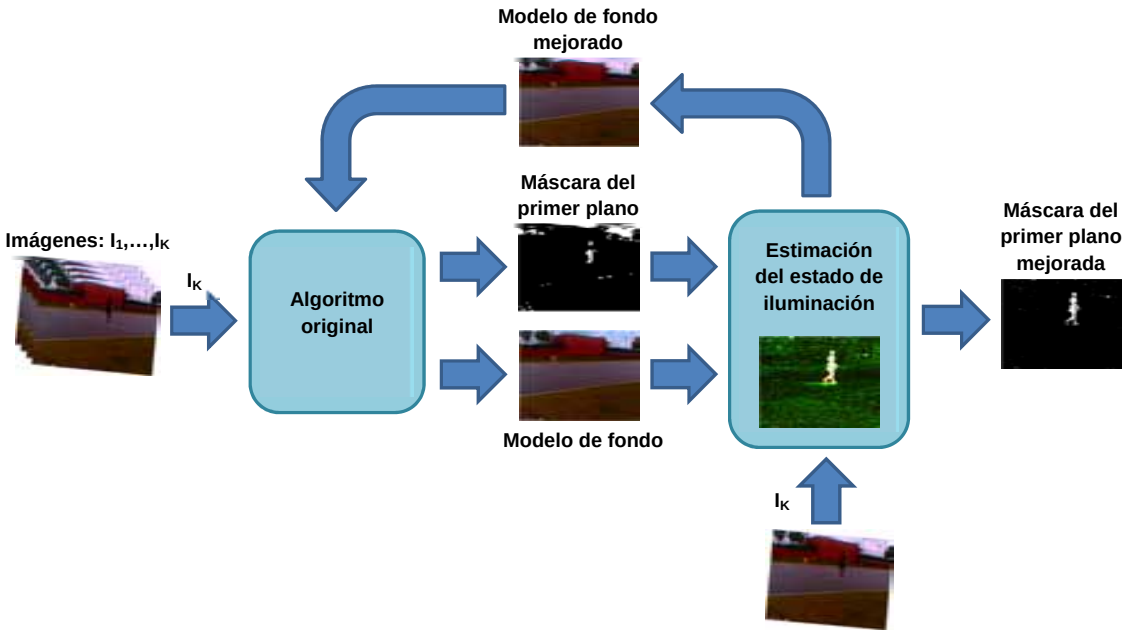



**Figura 6.4:** Diagrama de flujo de nuestra propuesta.

### 6.2.2. Reinicio de píxeles

Una vez que se calcula el estado de iluminación de cada píxel siguiendo el procedimiento explicado en la subsección anterior, se debe decidir qué píxeles deben reiniciarse. El reinicio de un píxel debería llevarse a cabo cuando el sistema falla en la detección de un cambio de iluminación en el fondo, lo que implica que el modelo de fondo no se ha actualizado durante ese fallo y los píxeles afectados han permanecido erróneamente en el estado "Objeto del primer plano" durante varios fotogramas. Para afrontar esta situación, mantenemos un contador $c_p$ para cada píxel $p$ que representa el número de fotogramas consecutivos en los que el modelo base lo ha declarado como primer plano. Si $c_p$ supera un límite preestablecido $C_{límite}$, entonces asumimos que el algoritmo base ha fallado, y por tanto el píxel debe ser reiniciado. El reinicio de un píxel se lleva a cabo aplicando el procedimiento de inicialización del algoritmo base que se esté empleando.

El estado de iluminación de los píxeles se usa para realizar un reinicio temprano de algunos píxeles mediante el incremento de su contadores $c_p$ en el caso de que se detecte un cambio de iluminación que les afecte. En este sentido se aplicando dos reglas en cada secuencia. La primera de ellas, denominada reinicio débil, indica que si un píxel $p$ se encuentra en el estado "Fondo con cambio de iluminación" en el fotograma actual, debe modificar su contador de la siguiente manera:

$$c_p = \text{máx}\left(c_p, C_{débil}\right) \tag{6.8}$$

La segunda regla, que denominamos reinicio fuerte, indica que si un píxel $p$ ha permanecido en el estado "Fondo con cambio de iluminación" durante los últimos $K_{fuerte}$ fotogramas, debe modificar su contador de la siguiente manera:

$$c_p = \text{máx}\left(c_p, C_{fuerte}\right) \tag{6.9}$$

donde

$$0 < C_{débil} \leq C_{fuerte} \leq C_{límite} \tag{6.10}$$

Las reglas de reinicio débil y fuerte modelan situaciones en las que la probabilidad de un cambio de iluminación es baja y alta, respectivamente. Conforme más tiempo esté un píxel en el estado "Fondo con cambio de iluminación", más probable es que el cambio de iluminación haya sido real y por lo tanto el modelo de fondo debe ser corregido.

## 6.3. Resultados experimentales

En esta sección analizamos el rendimiento de nuestra propuesta, en la Web[1] está disponible el código fuente junto con algunos vídeos demostrativos. Para llevar a cabo este análisis, hemos comparado las versiones originales de varios métodos de segmentación respecto de estos mismos métodos, pero modificados con nuestra propuesta, o lo que es lo mismo: usándolos como métodos base. Hemos usado un amplio conjunto de secuencias y parámetros para los métodos con el fin de obtener una idea clara de las ventajas de cada alternativa. Los métodos probados son citados en la Subsección 6.3.1. En la Subsección 6.3.2 se muestran las secuencias utilizadas. El conjunto de parámetros empleado en los experimentos se expone en la Subsección 6.3.3 y además, se analiza el efecto de los valores de dichos parámetros en el rendimiento de los métodos. Por último, las subsecciones 6.3.4 y 6.3.5 están dedicadas, respectivamente, a analizar los resultados de manera cualitativa y cuantitativa.

### 6.3.1. Métodos

Dado que nuestra propuesta analiza la escena a nivel de píxel, hemos elegido cinco métodos de este tipo para los experimentos: Pfinder, GrimsonGMM, KaewGMM, ZivkovicGMM y FASOM. Todos ellos son ampliamente conocidos y además no están diseñados específicamente para trabajar con cambios de iluminación, de ahí que hayan sido elegidos como métodos base. Hemos utilizado las implementaciones que ofrece la versión 1.3.0 de la biblioteca BGS. Para más detalles sobre estos métodos se puede consultar la Subsección 2.3.2.

Para completar el estudio, se han probado tres métodos de segmentación diseñados específicamente para este tipo de problemas y por tanto no han sido modificados con nuestra propuesta: Agrawal [220], Horprasert [212] y Reddy [221]. Hemos empleado la implementación que hay disponible en la página de los autores en el caso de Agrawal[2], para Horprasert hemos usado la implementación libre que hay en la Web[3] y por último, los autores del método Reddy nos han proporcionado el código fuente de su propuesta.

Excepto los métodos Agrawal y Horprasert que usan Matlab, todo el código está escrito en lenguaje C++. Se ha usado la versión 2.4.3 de la biblioteca OpenCV. Para reducir el tiempo de cómputo de nuestra propuesta, hemos usado la versión 4.1 de la biblioteca TBB[4] (*Threading Building Blocks*).

Todos los experimentos han sido ejecutados en un ordenador personal con un quad core 3.10 GHz CPU, 6 GB RAM, hardware estándar y como en otras ocasiones, sin ningún tipo de aceleración GPU.

[1] http://www.lcc.uma.es/%7Eezeqlr/lighting/lighting.html
[2] http://www.umiacs.umd.edu/~aagrawal/cvpr06/EdgeSuppression.html
[3] http://www.markyd13.com/code/background_subtraction/
[4] https://www.threadingbuildingblocks.org/

Para realizar una comparación clara y justa, no se ha realizado ningún post-procesado adicional, ni en los algoritmos originales, ni en las versiones modificadas.

### 6.3.2. Secuencias

Con el objetivo de analizar el comportamiento de los métodos en situaciones reales donde se experimentan cambios de iluminación, hemos considerado secuencias en las que dichos cambios son producidos tanto por la luz ambiental, como por focos de luz. En total hemos usado ocho secuencias reales: cinco de interiores y tres de exteriores. Todas ellas están disponibles en Internet. Los objetos a detectar son personas y vehículos. Las situaciones que representan son variadas, aunque fundamentalmente de dos tipos: cambios de luz abruptos, por ejemplo, debido al encendido o apagado de luces; o cambios de luces graduales, por ejemplo, debido al paso de las nubes en las escenas de exterior. Si bien la mayoría de las secuencias representan objetos a media distancia, también hemos contemplado otras en las que hay objetos lejanos y cercanos.

En cuanto a las secuencia de interior, la primera es Lobby (LB), disponible en este sitio web[5] y que se caracteriza tanto por sufrir cambios de iluminación bruscos, como por tener áreas del fondo particularmente vulnerables a los efectos de camuflaje. La siguiente secuencia de interior es RecCenter (RC) que presenta cambios de iluminación graduales y un gran número de personas atravesando la escena, pertenece al repositorio GTILT [222] y está disponible en el sitio web[6]. El caso contrario lo representa LightSwitch (LS), donde los cambios de iluminación son abruptos y únicamente hay una persona que atraviesa la escena en varias ocasiones a lo largo de la secuencia.

Por otro lado, el conjunto de secuencias de exterior está compuesto por: Cars1 (C1), Cars3 (C3), Roadside (RS), PED1 (P1) y Bank (BK). Todas pertenecen al repositorio GTILT. C1 y C2 representan entornos urbanos y una de sus características es que todos los objetos del primer plano están relativamente cercanos a la cámara. RS es una secuencia que muestra un entorno rural y presenta cambios de iluminación que afectan progresivamente a toda la imagen conforme se mueven las nubes por la escena. P1 es muy interesante porque experimenta cambios de iluminación significativos y hay objetos del primer plano tanto cercanos como lejanos. Por último, BK es una secuencia con grandes cambios de iluminación en donde aparecen vehículos y personas a corta distancia.

En general, los fotogramas *ground truth* no se proporcionan en los repositorios a los que pertenecen estas secuencias, por lo tanto hemos tenido que segmentar manualmente un conjunto de fotogramas de cada secuencia para poder comparar el rendimiento de cada alternativa. Para intentar ser lo más justos, hemos empleado

[5]http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
[6]http://www.ece.gatech.edu/research/pica/GTILT.html

una tasa de muestreo constante para determinar qué fotogramas van a ser usados como *ground truth*. Dado que la frecuencia con la que aparecen objetos en la escena es diferente en cada vídeo, la tasa de muestreo se ha adaptado de manera que la cantidad de *ground truth* sea similar en cada caso. Nótese que, como en otras ocasiones, únicamente hemos considerado como *ground truth* aquellos fotogramas que contienen objetos del primer plano. De esta manera hemos corregido casos como el de la secuencia Lobby en donde la mitad de los *ground truth* que proporciona el repositorio sólo consideran los últimos 200 fotogramas, siendo una secuencia de 1546 fotogramas. Nosotros hemos muestreado sobre 1446 fotogramas (los cien primeros fotogramas se han reservado para entrenamiento) de tal forma que se tienen en cuenta situaciones con diferentes niveles de iluminación. Por otro lado, dado que el repositorio GTILT proporciona como máximo la segmentación de un fotograma, ha sido inevitable la segmentación manual de los fotogramas *ground truth* en todas las secuencias de dicho repositorio.

### 6.3.3. Selección de parámetros

En esta subsección exponemos los parámetros utilizados para los métodos competidores y los que hemos usado para nuestra alternativa. El Cuadro 6.2 muestra los parámetros probados en las simulaciones. Se han incluido todos los valores que han demostrado buen rendimiento en alguna de las secuencias consideradas. Además, hemos añadido los valores recomendados en la biblioteca BGS y en las implementaciones originales. Una búsqueda exhaustiva de los parámetros no es realizable debido a la cantidad de parámetros a ajustar para cada método y secuencia. Nuestra propuesta únicamente necesita ajusta el valor de un parámetro $C_{límite}$, el cual determina cuándo se debe reiniciar un píxel, tal y como se expone en la Subsección 6.2.2. El resto de parámetros de nuestra propuesta no afectan notablemente al rendimiento, por lo que hemos usado los siguientes valores para cada uno de ellos: $\beta_{inf} = 10$, $\beta_{sup} = 50$, $K_\alpha = 100$, $K_\beta = 20$, $C_{débil} = 25$ y $C_{fuerte} = 50$. La combinación de todos los valores considerados para cada parámetro conforma el conjunto de configuraciones probadas para cada algoritmo.

### 6.3.4. Resultados cualitativos

En esta subsección se evalúa el rendimiento de los métodos de manera cualitativa. Las Figuras 6.5-6.8 muestran los resultados obtenidos en algunos fotogramas de las secuencias estudiadas tanto por los algoritmos originales, como por los modificados con nuestra propuesta. Se ha empleado la mejor configuración en cada uno de los casos. En general se observa que nuestra propuesta es capaz de eliminar una cantidad significativa de falsos positivos y, en menor medida, eliminar también falsos negativos.

| Método | Parámetros |
|---|---|
| Pfinder | Umbral, $T = \{6, 8, 10, 12, 14\}$<br>Tasa de aprendizaje, $\alpha = \{0.0025, 0.005, 0.0075, 0.01\}$<br>Fotogramas de aprendizaje, $N = \{20, 30, 60\}$ |
| GrimsonGMM | Umbral de emparejamiento, $T_{\sigma^2} = \{8, 9, 10, 11, 12\}$<br>Tasa de aprendizaje,<br>$\alpha = \{0.0025, 0.006, 0.0095, ...$<br>$0.013, 0.0165, 0.02\}$<br>Número de componentes gaussianas en la mixtura del modelo,<br>$K = \{3, 4, 5\}$ |
| KaewGMM | Fotogramas de aprendizaje, $N = \{50, 100, 150\}$<br>Número de componentes gaussianas en la mixtura del modelo,<br>$K = \{2, 3, 5\}$<br>Umbral, $T = \{0.4, 0.55, 0.7\}$<br>Desviación típica mínima, $\sigma_{mín} = \{5, 10, 15\}$ |
| ZivkovicGMM | Fotogramas de aprendizaje, $N = \{50, 100, 200, 300\}$<br>Umbral de emparejamiento, $T_{\sigma^2} = \{14, 16, 18, 20, 22, 24\}$ |
| FASOM | Fotogramas para ordenación, $N = \{60, 100\}$<br>Umbral durante la ordenación, $\epsilon_0 = \{100, 200, 300\}$<br>Umbral durante la convergencia, $\epsilon_1 = \{65, 90, 115\}$<br>Tasa de aprendizaje durante la ordenación, $\alpha_0 = \{180, 240, 300\}$<br>Tasa de aprendizaje durante la convergencia, $\alpha_1 = \{20, 40\}$ |
| Agrawal | Sigma, $\sigma = \{0.3, 0.4, 0.5\}$<br>Umbral menor, $N = \{10, 15, 25\}$<br>Umbral bajo, $\tau_1 = \{0.15, 0.3, 0.45\}$<br>Umbral alto, $\tau_2 = \{0.9, 0.8, 0.7\}$ |
| Horprasert | Distorsión de cromaticidad normalizada,<br>$\tau_{CD} = \{20000, 200000, 2000000, 20000000\}$<br>Distorsión de brillo normalizada,<br>$\tau_{\alpha lo} = \{-10, -20, -30\}$<br>Umbral brillo bajo, $\tau_{\alpha 1} = \{6, 10\}$<br>Umbral brillo alto,$\tau_{\alpha 2} = \{-6, -10\}$ |
| Reddy | Tamaño del bloque, $B_S = \{2, 8, 14\}$<br>Avance del bloque, $B_A = \{1, 2, 4, 8\}$ |

**Cuadro 6.2:** Valores de los parámetros utilizados en los experimentos. La combinación de todos ellos conforma el conjunto de todas las configuraciones probadas para los algoritmos originales. Para nuestra propuesta hemos probado los parámetros de los cinco primeros métodos en combinación con un valor de $C_{límite} = \{50, 80, 110\}$.

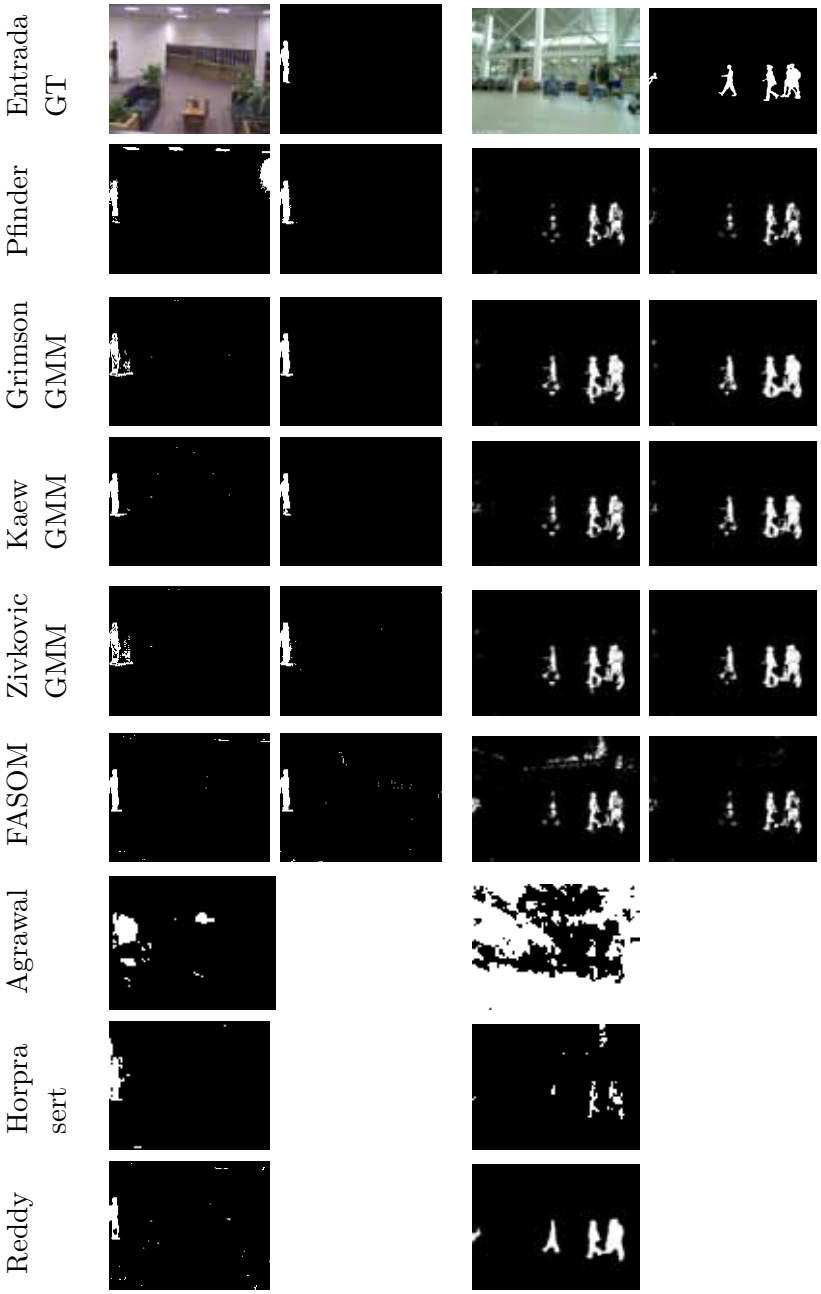



**Figura 6.5:** Resultados experimentales en secuencias de exterior. Las columnas pares muestran el fotograma original y las salidas de los algoritmos originales. Las columnas impares muestran el *ground truth* y los resultados de la versión modificada con nuestra propuesta. De izquierda a derecha: Lobby y RecCenter; fotogramas 2240 y 490, respectivamente. Como puede apreciarse, nuestra propuesta atenúa los problemas que produce la sombra proyectada.

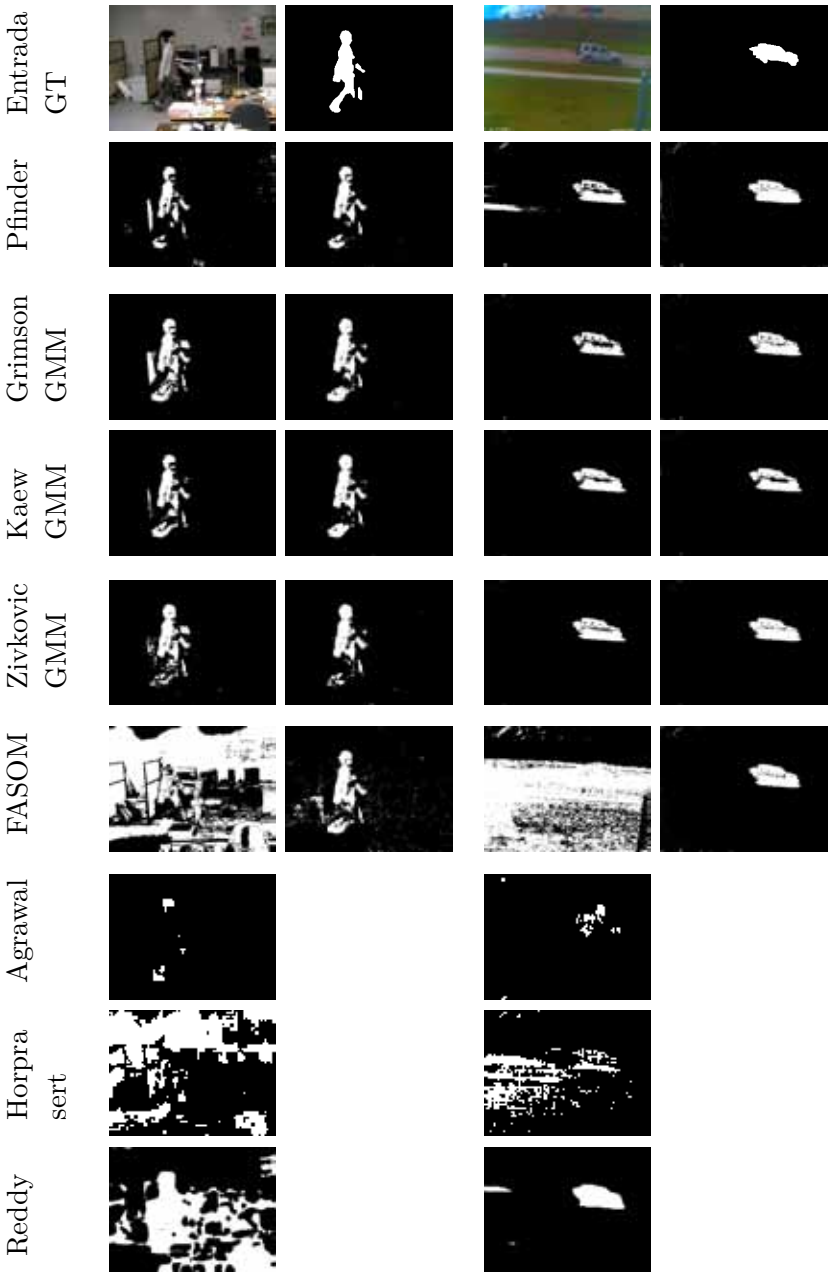



**Figura 6.6:** Resultados experimentales en secuencias de exterior e interior. Las columnas pares muestran el fotograma original y las salidas de los algoritmos originales. Las columnas impares muestran el *ground truth* y los resultados de la versión modificada con nuestra propuesta. De izquierda a derecha: LightSwitch y Cars3; fotogramas 1220 y 310 respectivamente. Los algoritmos originales segmentan algunos objetos del fondo como primer plano debido a los cambios de iluminación, nuestra propuesta corrige este comportamiento.

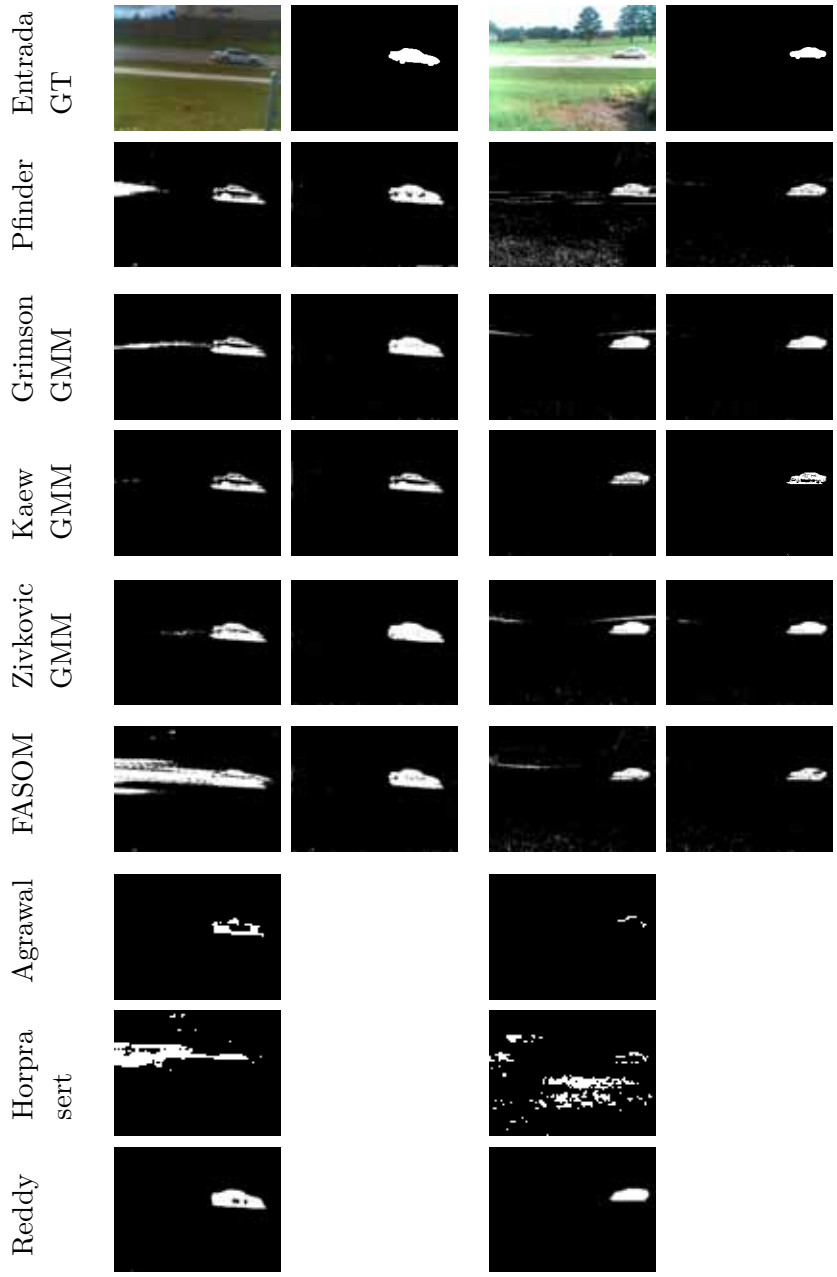



**Figura 6.7:** Resultados experimentales en secuencias de exterior. Las columnas pares muestran el fotograma original y las salidas de los algoritmos originales. Las columnas impares muestran el *ground truth* y los resultados de la versión modificada con nuestra propuesta. De izquierda a derecha: Cars1 y Roadside; fotogramas 178 y 135 respectivamente. Las regiones de falsos positivos son eliminadas en su mayoría por nuestro algoritmo y además los huecos en los objetos del primer plano son rellenados.

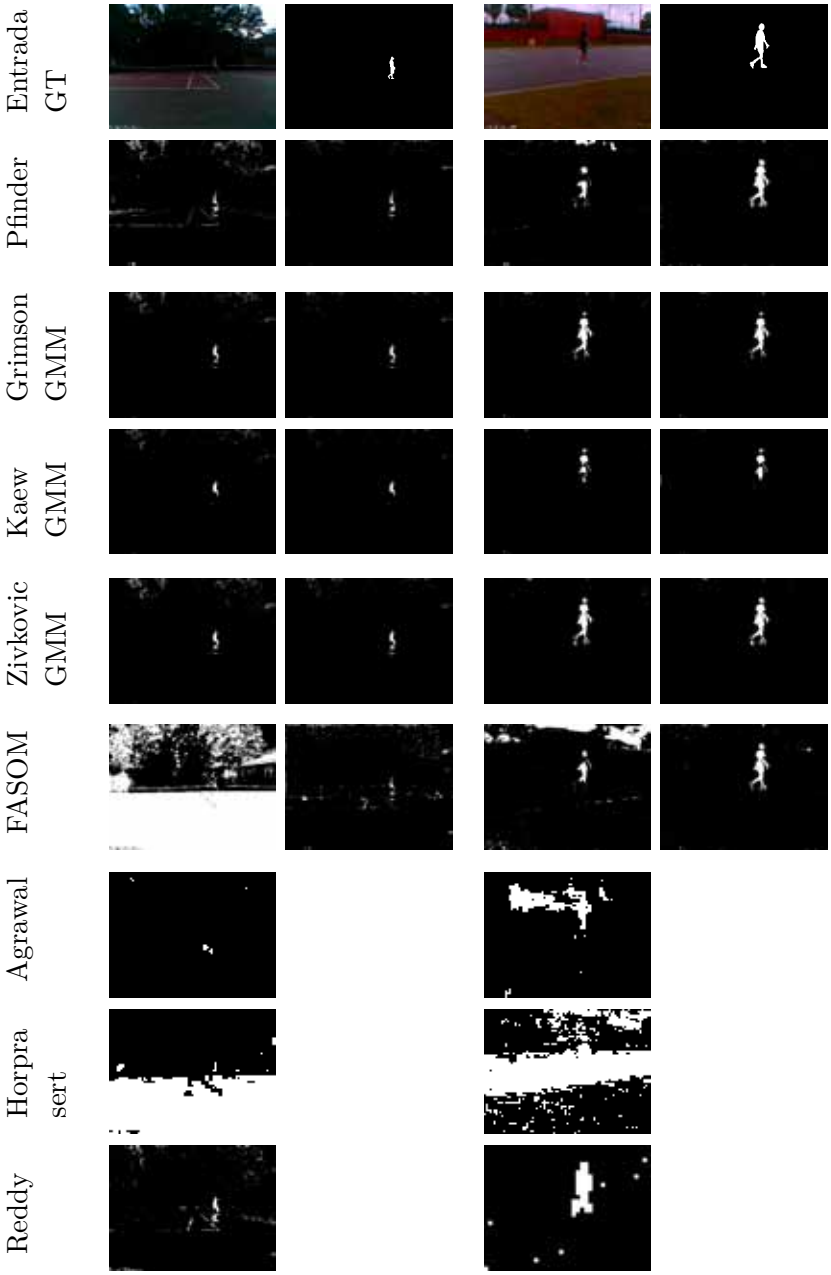
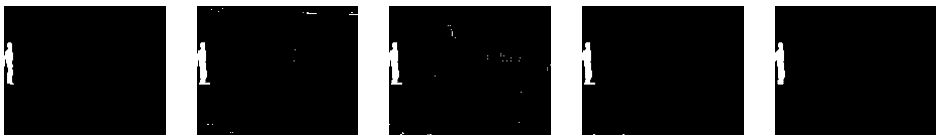



**Figura 6.8:** Resultados experimentales en secuencias de exterior. Las columnas pares muestran el fotograma original y las salidas de los algoritmos originales. Las columnas impares muestran el *ground truth* y los resultados de la versión modificada con nuestra propuesta. De izquierda a derecha: Ped1 y Bank; fotogramas 430 y 477 respectivamente. Los falsos positivos aislados son eliminados en su mayoría por nuestra propuesta.



GT Original Propuesta Elemento cuadrado Elemento diamante

**Figura 6.9:** Ejemplo de post-procesado empleando operadores morfológicos. De izquierda a derecha: *Ground truth*, salida del algoritmo original, modificación propuesta sin post-procesado, modificación propuesta usando post-procesado con el elemento estructurante cuadrado y por último, modificación propuesta usando post-procesado con el elemento estructurante diamante.

Es interesante señalar que en las secuencias de la Figura 6.5, Pfinder y FASOM sufren falsos positivos debido a los cambios en la luz procedente de focos, pero nuestra propuesta es capaz de corregir este comportamiento erróneo. Además, prácticamente todos los algoritmos originales sufren problemas derivados de las sombras proyectadas y de nuevo nuestra propuesta atenúa sus efectos en varios casos. La sombra proyectada es una dificultad relacionada con la iluminación, ya que es el foco de luz el que proyecta la sombra de los objetos, cuanto menor sea la luz ambiental respecto del foco de luz, más intensa será la sombra proyectada, lo que aumenta la probabilidad de que el método de segmentación falle al segmentar como primer plano la sombra proyectada de los objetos en movimiento.

El camuflaje es otro de los problemas habituales en los métodos de segmentación probados. Por ejemplo, en la secuencia Lobby (primera columna) y en particular en los métodos Pfinder y GrimsonGMM (tercera y quinta filas respectivamente), la persona no es segmentada por completo. Sin embargo, la segunda columna muestra que nuestra propuesta es capaz de aliviar este problema de forma significativa.

En la última secuencia de interior, es decir, LightSwitch en la Figura 6.6, el fotograma elegido es un claro ejemplo en el que los objetos del fondo son segmentados como primer plano debido a los cambios de iluminación. En la parte izquierda de la escena hay un objeto estático que es segmentado como parte el primer plano. Nuestra propuesta detecta la ocurrencia de un cambio de iluminación en esa zona y logra segmentarla correctamente. En este mismo fotograma podemos observar que FASOM segmenta pobremente la escena produciendo una gran cantidad de falsos positivos; este comportamiento no es un caso aislado, también se da en otras secuencias. Como puede apreciarse, al complementar este método con nuestra propuesta, la segmentación es mejor y se consigue corregir las dificultades comentadas hasta tal punto que el algoritmo FASOM modificado logra resultados similares al resto de algoritmos.

A continuación vamos a comentar el comportamiento de los métodos en el resto de vídeos, siendo todas ellas secuencias de exterior (últimas dos columnas de la Figura 6.6 y las Figuras 6.7 y 6.8). En este tipo de situaciones se produce un número

significativo de falsos positivos en el fondo debido a los cambios de iluminación, de nuevo nuestra propuesta es capaz de superar estas dificultades. Por otro lado, las salidas de los algoritmos originales arrojan una cantidad importante de falsos negativos, por ejemplo, los fotogramas correspondientes a C3, C1 y BK donde el vehículo no es segmentado por completo, nuestra propuesta es capaz de rellenar los huecos de forma satisfactoria en un buen número de ocasiones.

Como vemos en la Figura 6.5 nuestra propuesta produce algunos píxeles del primer plano aislados en la secuencia LB cuando se usa el método FASOM. Para corregir estos efectos podemos usar post-procesado en dos pasos, en primer lugar erosionando la imagen de salida con un elemento estructurante y después dilatando el resultado con el mismo elemento. En este caso hemos considerado dos elementos estructurantes: el elemento cuadrado $\mathbf{E}_{cuadrado}$ y el elemento con forma de diamante $\mathbf{E}_{diamante}$:

$$\mathbf{E}_{cuadrado} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{6.11}$$

$$\mathbf{E}_{diamante} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{6.12}$$

La Figura 6.9 muestra a título ilustrativo los resultados de dicho post-procesado. En concreto se muestran los resultados de la versión original y modificada del método FASOM en un fotograma de la secuencia LB. Si no se usa post-procesado, nuestra propuesta logra una F-medida de 0.79. Cuando utilizamos $\mathbf{E}_{cuadrado}$ como elemento estructurante, la salida mejora hasta una F-medida de 0.88 (cuarta columna). Por último, si empleamos $\mathbf{E}_{diamante}$ se logra un resultado aún mejor, 0.89 de F-medida (quinta columna). Debe señalarse que no hemos utilizado post-procesado en ninguno de los experimentos salvo en los que han servido para generar la Figura 6.9.

### 6.3.5. Resultados cuantitativos

La Figura 6.10 muestra los resultados de cada algoritmo en términos de varias medidas de rendimiento. Cada punto representa el rendimiento medio de una configuración en las secuencias de prueba. En el caso de FNR vs FPR, cuanto más cerca esté del origen de coordenadas, mejor es la segmentación. En el caso de las otras medidas, cuanto más alejado, mejor es el resultado. Por tanto, esta figura pone de manifiesto que nuestra propuesta mejora el algoritmo original en todos los métodos considerados ya que para cada configuración del algoritmo original, existe al menos una configuración para nuestra propuesta que la supera. También cabe destacar que los métodos Agrawal, Horprasert y Reddy son incapaces de obtener buenos resultados en estas secuencias.

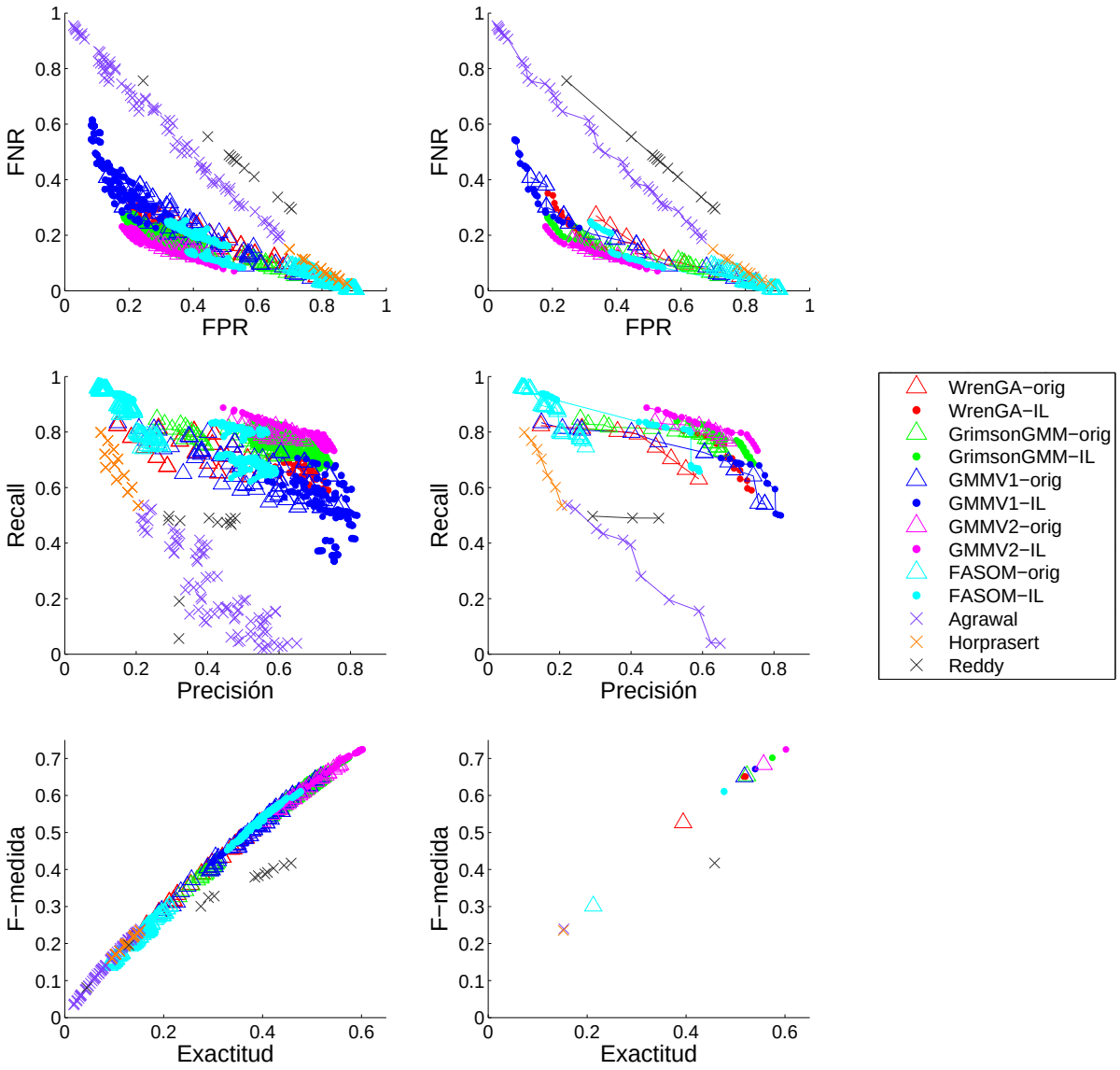



**Figura 6.10:** Rendimiento cuantitativo de las configuraciones elegidas. La primera columna muestra, respectivamente, tasa de falsos negativos (FNR) vs tasa de falsos positivos (FPR), recall (RC) vs precisión (PR) y F-medida (FM) vs exactitud (AC). La segunda columna muestra las configuraciones en el frente de Pareto. Cada punto corresponde a una configuración de un algoritmo y representa su rendimiento medio en las secuencias. Las versiones originales se han marcado con triángulos, nuestra propuesta con puntos y Agrawal, Horprasert y Reddy con cruces.

| Secuencia | Método | Propuesta | Original | Secuencia | Propuesta | Original |
|---|---|---|---|---|---|---|
| LB | Pfinder | **0.72 ± 0.13** | 0.68 ± 0.19 | C1 | **0.68 ± 0.14** | 0.54 ± 0.19 |
| | GrimsonGMM | **0.77 ± 0.08** | 0.75 ± 0.06 | | **0.72 ± 0.14** | 0.61 ± 0.17 |
| | KaewGMM | 0.74 ± 0.17 | **0.80 ± 0.06** | | 0.57 ± 0.25 | **0.60 ± 0.12** |
| | ZivkovicGMM | **0.68 ± 0.17** | 0.65 ± 0.21 | | **0.72 ± 0.14** | 0.71 ± 0.17 |
| | FASOM | **0.74 ± 0.09** | 0.63 ± 0.31 | | **0.69 ± 0.16** | 0.28 ± 0.20 |
| | Agrawal | N/A | 0.29 ± 0.19 | | N/A | 0.25 ± 0.25 |
| | Horprasert | N/A | 0.36 ± 0.20 | | N/A | 0.28 ± 0.19 |
| | Reddy | N/A | 0.49 ± 0.21 | | N/A | 0.57 ± 0.05 |
| RC | Pfinder | 0.63 ± 0.15 | **0.64 ± 0.15** | RS | **0.72 ± 0.12** | 0.35 ± 0.26 |
| | GrimsonGMM | **0.59 ± 0.31** | 0.57 ± 0.28 | | **0.75 ± 0.17** | 0.65 ± 0.25 |
| | KaewGMM | **0.68 ± 0.20*** | 0.65 ± 0.17 | | **0.71 ± 0.12** | 0.65 ± 0.20 |
| | ZivkovicGMM | **0.60 ± 0.32** | 0.57 ± 0.31 | | **0.80 ± 0.13*** | 0.70 ± 0.18 |
| | FASOM | **0.63 ± 0.16** | 0.51 ± 0.23 | | **0.68 ± 0.10** | 0.44 ± 0.18 |
| | Agrawal | N/A | 0.34 ± 0.19 | | N/A | 0.27 ± 0.21 |
| | Horprasert | N/A | 0.37 ± 0.20 | | N/A | 0.24 ± 0.20 |
| | Reddy | N/A | 0.56 ± 0.09 | | N/A | 0.44 ± 0.19 |
| LS | Pfinder | **0.77 ± 0.08** | 0.65 ± 0.17 | P1 | **0.56 ± 0.18** | 0.26 ± 0.19 |
| | GrimsonGMM | **0.80 ± 0.07** | 0.76 ± 0.13 | | **0.67 ± 0.11** | 0.53 ± 0.20 |
| | KaewGMM | **0.78 ± 0.08** | 0.75 ± 0.10 | | **0.65 ± 0.09** | 0.60 ± 0.11 |
| | ZivkovicGMM | **0.78 ± 0.08** | 0.74 ± 0.14 | | **0.74 ± 0.09*** | 0.57 ± 0.15 |
| | FASOM | **0.67 ± 0.14** | 0.18 ± 0.22 | | **0.32 ± 0.19** | 0.06 ± 0.09 |
| | Agrawal | N/A | 0.26 ± 0.14 | | N/A | 0.26 ± 0.19 |
| | Horprasert | N/A | 0.22 ± 0.18 | | N/A | 0.07 ± 0.10 |
| | Reddy | N/A | 0.22 ± 0.14 | | N/A | 0.39 ± 0.08 |
| C3 | Pfinder | **0.80 ± 0.04** | 0.71 ± 0.11 | BK | **0.82 ± 0.05** | 0.63 ± 0.16 |
| | GrimsonGMM | **0.84 ± 0.03** | 0.82 ± 0.05 | | 0.82 ± 0.09 | **0.82 ± 0.07** |
| | KaewGMM | **0.82 ± 0.06** | 0.81 ± 0.05 | | 0.71 ± 0.17 | **0.73 ± 0.14** |
| | ZivkovicGMM | **0.85 ± 0.02** | 0.84 ± 0.02 | | **0.84 ± 0.05*** | 0.83 ± 0.05 |
| | FASOM | **0.82 ± 0.03** | 0.10 ± 0.10 | | **0.76 ± 0.08** | 0.42 ± 0.27 |
| | Agrawal | N/A | 0.18 ± 0.14 | | N/A | 0.47 ± 0.18 |
| | Horprasert | N/A | 0.12 ± 0.13 | | N/A | 0.32 ± 0.29 |
| | Reddy | N/A | 0.53 ± 0.05 | | N/A | 0.40 ± 0.13 |

**Cuadro 6.3:** Mejores resultados en términos de F-medida. La primera y quinta columnas denotan el nombre de la secuencia, mientras que la tercera y cuarta columnas muestran, respectivamente, los mejores resultados de acuerdo a la F-medida de nuestra propuesta y del algoritmo original para las cuatro primeras secuencias. La sexta y séptima columnas corresponden a los resultados de las otras cuatro secuencias. La mejor versión de cada método se ha marcado en negrita para todas las secuencias. El asterisco denota que un resultado es significativamente mejor que el segundo mejor resultado para una secuencia dada.

El Cuadro 6.3 se muestran los rendimientos medios de cada método utilizando las configuraciones óptimas en términos de F-medida. Es posible observar las diferencias entre los algoritmos originales y las versiones modificadas. Como podemos ver, nuestra propuesta supera a los algoritmos originales en la mayoría de los casos, únicamente la versión original de KaewGMM es capaz de obtener mejores resultados en las secuencias LB, C1 y BK. Debemos mencionar que los algoritmos originales de Pfinder y GrimsonGMM también logran mejores resultados en RC y BK, pero la diferencia con nuestra propuesta es menor.

De forma complementaria a la F-medida, en el Cuadro 6.4 se mide el rendimiento en términos de exactitud. En este caso se observa que las versiones modificadas con nuestra propuesta mejoran los algoritmos originales incluso en mayor proporción que

| Secuencia | Método | Propuesta | Original | Secuencia | Propuesta | Original |
|---|---|---|---|---|---|---|
| LB | Pfinder | **0.58 ± 0.16** | 0.55 ± 0.20 | C1 | **0.53 ± 0.15** | 0.39 ± 0.16 |
| | GrimsonGMM | **0.63 ± 0.11** | 0.60 ± 0.09 | | **0.58 ± 0.16** | 0.46 ± 0.15 |
| | KaewGMM | 0.62 ± 0.19 | **0.67 ± 0.09** | | **0.44 ± 0.23** | 0.44 ± 0.18 |
| | ZivkovicGMM | **0.55 ± 0.20** | 0.51 ± 0.20 | | **0.58 ± 0.16** | 0.57 ± 0.16 |
| | FASOM | **0.59 ± 0.11** | 0.53 ± 0.27 | | **0.55 ± 0.17** | 0.18 ± 0.14 |
| | Agrawal | N/A | 0.18 ± 0.15 | | N/A | 0.17 ± 0.19 |
| | Horprasert | N/A | 0.24 ± 0.15 | | N/A | 0.18 ± 0.14 |
| | Reddy | N/A | 0.61 ± 0.30 | | N/A | 0.67 ± 0.10* |
| RC | Pfinder | 0.49 ± 0.19 | **0.49 ± 0.15** | RS | **0.58 ± 0.14** | 0.24 ± 0.22 |
| | GrimsonGMM | **0.48 ± 0.28** | 0.45 ± 0.27 | | **0.62 ± 0.17** | 0.52 ± 0.23 |
| | KaewGMM | **0.54 ± 0.19** | 0.50 ± 0.17 | | **0.56 ± 0.14** | 0.51 ± 0.19 |
| | ZivkovicGMM | **0.49 ± 0.28** | 0.46 ± 0.27 | | **0.67 ± 0.13*** | 0.57 ± 0.17 |
| | FASOM | **0.49 ± 0.21** | 0.37 ± 0.20 | | **0.52 ± 0.11** | 0.30 ± 0.15 |
| | Agrawal | N/A | 0.22 ± 0.15 | | N/A | 0.17 ± 0.16 |
| | Horprasert | N/A | 0.25 ± 0.16 | | N/A | 0.16 ± 0.14 |
| | Reddy | N/A | 0.66 ± 0.14* | | N/A | 0.51 ± 0.31 |
| LS | Pfinder | **0.63 ± 0.10** | 0.50 ± 0.17 | P1 | **0.41 ± 0.16** | 0.17 ± 0.14 |
| | GrimsonGMM | **0.67 ± 0.11** | 0.62 ± 0.15 | | **0.52 ± 0.14** | 0.39 ± 0.16 |
| | KaewGMM | **0.65 ± 0.10** | 0.61 ± 0.13 | | **0.49 ± 0.10** | 0.44 ± 0.11 |
| | ZivkovicGMM | **0.65 ± 0.10** | 0.60 ± 0.15 | | **0.59 ± 0.11*** | 0.41 ± 0.15 |
| | FASOM | **0.52 ± 0.14** | 0.12 ± 0.20 | | **0.21 ± 0.14** | 0.03 ± 0.06 |
| | Agrawal | N/A | 0.16 ± 0.09 | | N/A | 0.16 ± 0.14 |
| | Horprasert | N/A | 0.14 ± 0.14 | | N/A | 0.04 ± 0.06 |
| | Reddy | N/A | 0.17 ± 0.19 | | N/A | 0.33 ± 0.11 |
| C3 | Pfinder | **0.66 ± 0.05** | 0.56 ± 0.13 | BK | **0.70 ± 0.08** | 0.48 ± 0.17 |
| | GrimsonGMM | **0.72 ± 0.04** | 0.70 ± 0.07 | | **0.70 ± 0.12** | 0.69 ± 0.09 |
| | KaewGMM | **0.69 ± 0.09** | 0.69 ± 0.07 | | 0.57 ± 0.20 | **0.59 ± 0.17** |
| | ZivkovicGMM | **0.73 ± 0.03** | 0.73 ± 0.04 | | **0.73 ± 0.07*** | 0.72 ± 0.07 |
| | FASOM | **0.69 ± 0.04** | 0.06 ± 0.06 | | **0.63 ± 0.11** | 0.31 ± 0.24 |
| | Agrawal | N/A | 0.11 ± 0.09 | | N/A | 0.33 ± 0.16 |
| | Horprasert | N/A | 0.07 ± 0.08 | | N/A | 0.23 ± 0.23 |
| | Reddy | N/A | 0.58 ± 0.10 | | N/A | 0.37 ± 0.16 |

**Cuadro 6.4:** Mejores resultados en términos de exactitud. La primera y quinta columnas denotan el nombre de la secuencia, mientras que la tercera y cuarta columnas muestran, respectivamente, los mejores resultados de acuerdo a la exactitud de nuestra propuesta y del algoritmo original para las cuatro primeras secuencias. La sexta y séptima columnas corresponden a los resultados de las otras cuatro secuencias. La mejor versión de cada método se ha marcado en negrita para todas las secuencias. El asterisco denota que un resultado es significativamente mejor que el segundo mejor resultado para una secuencia dada.

en el caso de la F-medida. Sólo existen tres vídeos donde hay un método en el que nuestra propuesta no es capaz de superar el rendimiento del algoritmo original: LB, RC y BK. De nuevo el algoritmo original de KaewGMM es el ejemplo más claro de esta circunstancia, obteniendo mejores resultados tanto en LB como en BK. Pfinder también supera a nuestra propuesta en RC, pero al igual que antes la diferencia es menor.

Si analizamos el rendimiento de los métodos de segmentación orientados a detectar sombras/iluminación, esto es: Agrawal, Horprasert y Reddy, observamos que Reddy logra mejores resultados que nuestra propuesta en dos de las secuencias en términos de exactitud. No obstante hay que señalar que esto no sucede en el caso de la F-medida. De hecho, Reddy es el único método que obtiene una magnitud mayor

al medir su rendimiento con la medida exactitud que al usar la F-medida. Este comportamiento se debe a que la exactitud favorece a los algoritmos con pocos falsos negativos, siendo menos sensible a los falsos positivos que la F-medida, la cual penaliza ambas situaciones [223]. Como puede apreciarse en la Figura 6.10, esta circunstancia es la que ocurre con el método Reddy, en donde la mayoría de las configuraciones producen más falsos positivos que negativos.

Para apoyar los resultados ya comentados y tener una visión aún más clara de la diferencia de rendimiento entre los algoritmos originales y las versiones modificadas, hemos elaborado la Figura 6.11. En particular aquí mostramos la diferencia en términos de F-medida entre ambas versiones en cada fotograma *ground truth*. De nuevo los resultados corresponden a la mejor configuración según F-medida para cada método y secuencia. La mejoría es diferente en cada caso, pero FASOM y en menor medida Pfinder son los métodos en donde se aprecia claramente un aumento del rendimiento al aplicar nuestra propuesta, ya que en la mayoría de las situaciones los resultados de la versión modificada son mejores.

La Figura 6.12 es análoga a la comentada previamente, ilustra las diferencias en cada *ground truth* entre los métodos para detectar sombras/iluminación y los algoritmos modificados con nuestra propuesta. Como podemos observar, cada gráfico corresponde a una secuencia y en todos ellos se pone de manifiesto el pobre resultado de estos tres métodos. De hecho, sólo en algunos fotogramas son capaces de mejorar a los algoritmos modificados con nuestra propuesta.

Si bien no es el objetivo principal de este trabajo, debemos señalar que, según estos cuadros el mejor método es ZivkovicGMM modificado con nuestra propuesta ya que logra los mejores resultados en cinco de las ocho secuencias. Esta afirmación se ve reforzada si observamos que en la última gráfica de la Figura 6.10, es decir, la configuración en el frente de Pareto de F-medida frente a exactitud, el mejor rendimiento medio lo obtiene una configuración de este método.

Hay que señalar que se ha llevado a cabo un estudio estadístico sobre significancia de los resultados cuantitativos mostrados en los Cuadros 6.3 y 6.4. Para ello se ha empleado la prueba no paramétrica de Friedman con la correspondiente prueba de Dunn para determinar si la diferencia entre los dos mejores resultados para una medida y secuencia es estadísticamente significativa. Estas pruebas son robustas para realizar comparaciones sobre múltiples conjuntos de datos [224]. En este estudio se ha considerado un nivel de confianza del 95 % en todos los casos. Para señalar aquellas configuraciones cuyos resultados son mejores que la segunda mejor configuración con un nivel de confianza del 95 %, hemos marcado su rendimiento con un asterisco. De este modo podemos ver que nuestra propuesta consigue resultados significativamente mejores en cuatro de las ocho secuencias en términos de F-medida y en tres en el caso de la exactitud. Ninguno de los algoritmos originales es significativamente mejor que la versión modificada con nuestra propuesta. El método Reddy es significativamente mejor en dos secuencias atendiendo únicamente a la medida de exactitud.

Para concluir esta subsección, el Cuadro 6.5 muestra el rendimiento FPS que cada

uno de los algoritmos desarrolla en las diferentes secuencias. Dado que los requisitos de tiempo real son satisfechos cuando se supera el umbral de los 15 fps, se observa que todos los algoritmos verifican esta condición a excepción de Agrawal y Reddy. Como vemos, GrimsonGMM y FASOM son los métodos que al ser mejorados con nuestra propuesta, requieren mayor tiempo de cómputo. Por otro lado, la versión modificada de ZivkovicGMM es el algoritmo modificado más rápido y, como hemos expuesto previamente, una de las opciones que logra mejores resultados. Nuestra propuesta requiere una mayor carga computacional, pero debe tenerse en cuenta que no se ha hecho uso de ninguna aceleración hardware, por lo que es posible mejorar este aspecto si fuera necesario.

| Secuencia | Método | Propuesta | Original |
|---|---|---|---|
| LB | Pfinder | 79 | 1040 |
| | GrimsonGMM | 59 | 288 |
| | KaewGMM | 124 | 606 |
| | ZivkovicGMM | 149 | 765 |
| | FASOM | 102 | 260 |
| | Agrawal | N/A | 1 |
| | Horprasert | N/A | 239 |
| | Reddy | N/A | 76 |
| RC | Pfinder | 23 | 286 |
| | GrimsonGMM | 16 | 100 |
| | KaewGMM | 25 | 194 |
| | ZivkovicGMM | 27 | 207 |
| | FASOM | 22 | 67 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 70 |
| | Reddy | N/A | 6 |
| LS | Pfinder | 22 | 283 |
| | GrimsonGMM | 16 | 41 |
| | KaewGMM | 21 | 154 |
| | ZivkovicGMM | 25 | 191 |
| | FASOM | 22 | 73 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 66 |
| | Reddy | N/A | 6 |
| C3 | Pfinder | 23 | 286 |
| | GrimsonGMM | 16 | 96 |
| | KaewGMM | 23 | 170 |
| | ZivkovicGMM | 25 | 222 |
| | FASOM | 23 | 71 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 67 |
| | Reddy | N/A | 11 |

| Secuencia | Método | Propuesta | Original |
|---|---|---|---|
| C1 | Pfinder | 22 | 291 |
| | GrimsonGMM | 17 | 110 |
| | KaewGMM | 25 | 177 |
| | ZivkovicGMM | 25 | 214 |
| | FASOM | 23 | 69 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 67 |
| | Reddy | N/A | 5 |
| RS | Pfinder | 23 | 272 |
| | GrimsonGMM | 16 | 64 |
| | KaewGMM | 20 | 172 |
| | ZivkovicGMM | 25 | 176 |
| | FASOM | 22 | 71 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 68 |
| | Reddy | N/A | 4 |
| P1 | Pfinder | 22 | 279 |
| | GrimsonGMM | 15 | 64 |
| | KaewGMM | 21 | 140 |
| | ZivkovicGMM | 25 | 175 |
| | FASOM | 22 | 71 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 69 |
| | Reddy | N/A | 9 |
| BK | Pfinder | 24 | 287 |
| | GrimsonGMM | 15 | 48 |
| | KaewGMM | 25 | 169 |
| | ZivkovicGMM | 26 | 201 |
| | FASOM | 22 | 74 |
| | Agrawal | N/A | 0.3 |
| | Horprasert | N/A | 68 |
| | Reddy | N/A | 156 |

**Cuadro 6.5:** Fotogramas por segundo de cada una de las propuestas en cada secuencia. Los valores corresponden a la mejor configuración en términos de exactitud. LB es una secuencia de tamaño $160 \times 128$ píxeles, mientras que las otras secuencias son $640 \times 480$ que han sido reducidas a $320 \times 240$ píxeles.

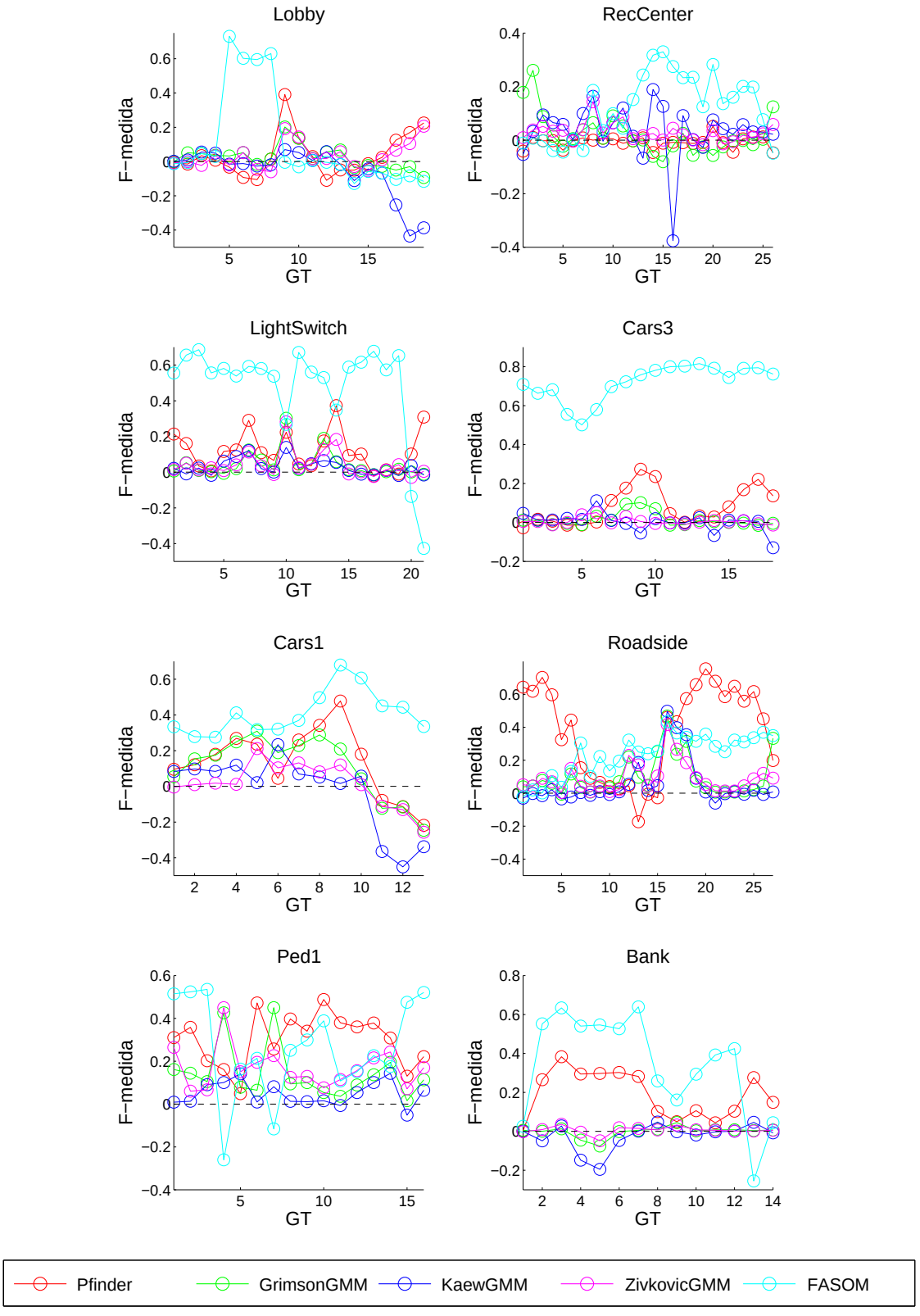



**Figura 6.11:** Diferencia de rendimiento entre el algoritmo original y el modificado. El eje horizontal denota el *ground truth* y el vertical, la diferencia entre ambas versiones. Un valor positivo significa que nuestra versión es mejor que el original.

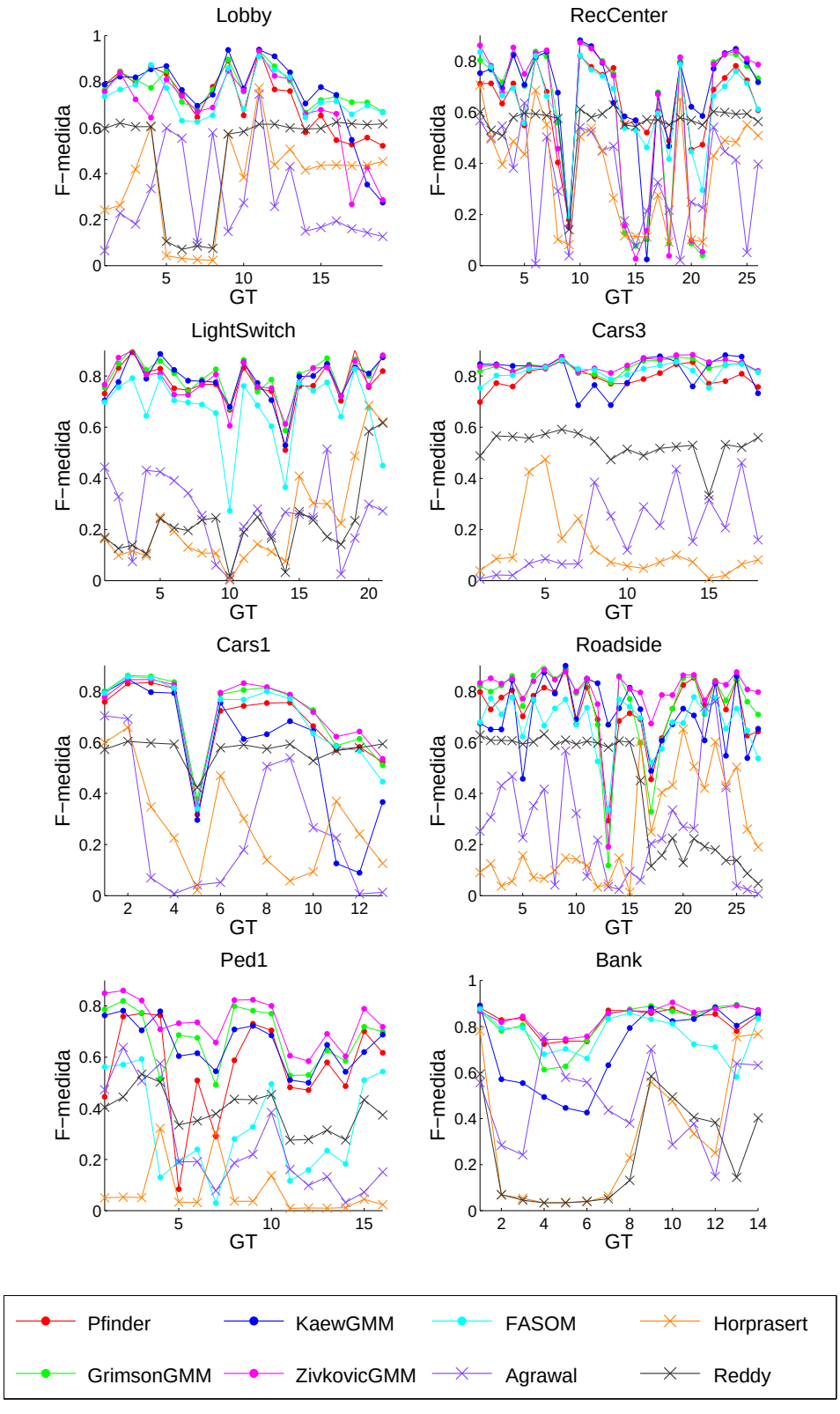



**Figura 6.12:** Comparativa entre nuestra propuesta y los métodos de detección de sombras/iluminación. El eje horizontal denota el *ground truth* y el vertical la F-medida.

## 6.4. Discusión

La diferencia principal de nuestra propuesta respecto de otros métodos de segmentación especializados en entornos de iluminación variable consiste en que no presupone una forma específica del cambio de color que se produce cuando tiene lugar un cambio de iluminación. Esto se traduce en que nuestra propuesta no depende de las propiedades físicas de la luz que hay presente en el entorno. Por lo tanto, es posible aplicar nuestra solución a una amplia variedad de situaciones donde las condiciones de la luz varían de manera impredecible. Hay que destacar que la mayoría de las propuestas actuales asumen de manera explícita o implícita un modelo para la variación del color del píxel que produce un cambio de iluminación [225, 226, 227, 228]. Un caso particularmente interesante es la propuesta realizada en [219], donde únicamente se asume que el signo de la diferencia entre dos medidas de luminancia se mantiene cuando se producen cambios de iluminación en un entorno local del fotograma. Sin embargo, esta suposición no se cumple en la práctica en las regiones texturizadas, donde algunos píxeles pueden incrementar su luminancia, mientras que otros la disminuyen. A diferencia de esto, nuestra propuesta maneja correctamente las regiones texturizadas porque comprueba que los píxeles que tienen un color similar en un entorno local antes del cambio de iluminación, siguen teniendo un color similar después de dicho cambio, de este modo los píxeles que corresponden a materiales diferentes en una textura nunca son comparados, como sí ocurre en [219].

Como indican los experimentos, los métodos planteados en [220, 212, 221] son superados por nuestra propuesta. Esto se debe a que estos métodos consideran un modelo excesivamente restrictivo para los cambios de iluminación. En [212] se usa una transformación lineal de los valores RGB de modo que el color observado es proyectado sobre una línea dentro del cubo de color RGB, la cual se supone que debe modelar todos los posibles estados de iluminación del píxel. Esto es poco realista ya que la respuesta de los materiales respecto de la luz incidente puede ser no lineal, por ejemplo cuando el contenido de color de la luz incidente es diferente en los tres canales RGB. El cambio de iluminación se maneja en [221] considerando que el ángulo subtendido entre el color observado y el color de fondo medio estimado es pequeño en los cambios de iluminación. De nuevo esto equivale a suponer que la respuesta de los materiales a la luz es similar para las tres componentes de color RGB, lo cual no es realista por las razones previamente señaladas. Por otro lado, la propuesta de [220] detecta los objetos del primer plano centrándose en aquellos cuyos bordes no están presentes en el modelo de fondo. Esta estrategia falla en las regiones homogéneas o en los vídeos con ruido donde la información local de bordes no es fiable. Además la respuesta desigual de los materiales frente a la luz puede llevar a que los bordes locales tengan diferente apariencia cuando cambia la iluminación.

Dadas las consideraciones anteriores, a continuación vamos a realizar una evaluación cualitativa de los retos a los que se enfrentan los algoritmos competidores para la detección de cambios de iluminación. Como podemos observar en la Figura 6.13 Agrawal funciona bien en situaciones donde existen bordes claramente definidos (pri-

mera fila), pero falla al segmentar aquellos fotogramas que tienen regiones planas o donde se producen efectos de camuflaje (segunda fila). Nótese que estos fallos se deben a que la información de bordes es poco fiable, tal y como se expuso previamente. Horprasert es muy dependiente de la fase de entrenamiento porque el modelo inicial no es actualizado. La primera fila de la Figura 6.14 muestra una situación donde este método segmenta correctamente la imagen, pero en la segunda fila se observa que falla al no adaptarse a los cambios de iluminación que se producen en la parte izquierda de la imagen, lo cual ejemplifica las limitaciones de su modelo de cambio de iluminación. Uno de los parámetros necesarios para configurar Reddy es el avance del bloque. Valores altos conllevan bordes pobremente definidos en los objetos del primer plano y por lo tanto, grandes cantidades de falsos positivos, véase la última imagen de la Figura 6.8 que se corresponde con un avance del bloque igual a 8. Si usamos un valor bajo, situación que se muestra en la primera fila de la Figura 6.15 y en la última imagen de la Figura 6.5, el problema es parcialmente resuelto ya que los bordes son más suaves, pero el rendimiento computacional cae de manera preocupante en términos de FPS. Otro problema común es que Reddy no funciona bien frente a cambios en la luz ambiental debido a su pobre modelo de cambio de iluminación, tal y como se pone de manifiesto en la segunda fila de la Figura 6.15.


**Figura 6.13:** Ejemplos ilustrativos de las dificultades típicas del método Agrawal. La primera y segunda filas corresponden al fotograma 2340 y 1365 de la secuencia LB. De izquierda a derecha: fotograma original, salida de Agrawal y GrimsonGMM modificada con nuestra propuesta.

En nuestra propuesta la decisión de que un píxel haya experimentado un cambio de iluminación no se materializa en variables binarias (sí/no), si no que se emplean variables difusas que son capaces de manejar la incertidumbre inherente en la determinación del estado de iluminación de un píxel. De este modo somos capaces de usar la mayor cantidad de información posible para la evaluación de las condiciones de iluminación. La validez de las propuestas basadas en lógica difusa para la detección del primer plano bajo condiciones de iluminación variables ya ha sido demostrada en la literatura [229, 230, 231].

Se ha establecido un procedimiento para combinar la información del estado de iluminación de los píxeles vecinos. Esto es de suma importancia ya que a nivel de

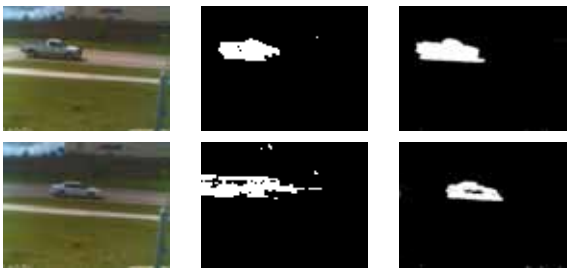



**Figura 6.14:** Ejemplos ilustrativos de las dificultades típicas del método Horprasert. La primera y segunda filas corresponden al fotograma 47 y 174 de la secuencia C1. De izquierda a derecha: fotograma original, salida de Horprasert y GrimsonGMM modificada con nuestra propuesta.


**Figura 6.15:** Ejemplos ilustrativos de las dificultades típicas del método Reddy. La primera y segunda filas corresponden al fotograma 1163 y 1630 de la secuencia LB. De izquierda a derecha: fotograma original, salida de Reddy y GrimsonGMM modificada con nuestra propuesta.

píxel no es posible distinguir un cambio de iluminación, de un objeto en movimiento con color homogéneo. Esta situación, que es común a todas las propuestas que trabajan a nivel de píxel [232, 233, 234], es modelada en nuestra propuesta mediante el estado "dudoso" de un píxel (véase Cuadro 6.1).

Los resultados experimentales muestran que nuestro procedimiento es capaz de mejorar el rendimiento de varios métodos de detección del primer plano actuales, de tal modo que la diferencia es estadísticamente significativa con un alto grado de confianza. También se observa que la mayor parte de la mejora proviene de la eliminación de falsos positivos, lo que significa que los cambios de iluminación ya no son confundidos como objetos del primer plano, tal y como se espera de este tipo de algoritmos [228, 235, 236, 222, 237].

## 6.5. Conclusiones

En este capítulo hemos presentado un procedimiento para la detección de cambios de iluminación. Nuestra propuesta está diseñada para integrarse en un algoritmo de detección de movimiento existente. Se basa en el estudio de la transformación del color local que experimentan los píxeles cuando se produce un cambio de iluminación. Una de las ventajas consiste en que sólo necesitamos ajustar un parámetro y los resultados obtenidos son buenos usando un pequeño conjunto de valores diferentes. Por lo tanto es fácil de usar y no es necesario que se tenga un alto grado de conocimiento acerca de su funcionamiento.

Con el fin de demostrar su buen rendimiento, hemos considerado un conjunto de ocho secuencias reales en donde se representan entornos complejos tanto de exterior como de interior. Todas ellas pueden encontrarse en repositorios públicos. Nuestra propuesta se ha integrado en cinco métodos de detección de movimiento actuales. Esto significa un total de 40 experimentos en los que nuestra propuesta logra superar el algoritmo original en 35 de ellos, mientras que en los otros cinco casos obtiene resultados competitivos. También hemos probado tres métodos específicamente diseñados para la detección del primer plano y de sombras, sin embargo sus resultados han sido pobres comparados con los de nuestra propuesta.

De lo anterior podemos concluir que nuestra propuesta aumenta el rendimiento de los métodos de segmentación existentes en una amplia variedad de condiciones donde se producen cambios de iluminación, lo cual demuestra su potencial para mejorar los sistemas de visión por computador, incluyendo los subsistemas de detección del primer plano.

# 7 Cámara en movimiento

La mayoría de los algoritmos de detección del primer plano no son capaces de segmentar adecuadamente las secuencias grabadas con cámaras de videovigilancia PTZ (*pan-tilt-zoom*) ni tampoco con cámaras fijas que experimentan vibraciones. Esto se debe a que su modelo de la escena asume que el fondo no se desplaza. Para afrontar este tipo de dificultades, en este capítulo proponemos un nuevo modelo basado en mixturas probabilísticas que emplea aproximación estocástica para el aprendizaje. Está diseñado para adaptarse a una amplia variedad de cambios en la cámara, en particular podrá realizar zoom y movimientos tanto en el plano horizontal como en el vertical, además es válido para secuencias donde la cámara cambia de posición (egomovimiento) siempre que estos cambios no sean bruscos. En resumen podemos decir que es un modelo no panorámico para cámaras en movimiento, donde el movimiento de la misma es tal que es posible reutilizar suficiente información del fondo del fotograma anterior. Nuestra propuesta emplea dos modelos de fondo, uno de ellos para seguir el movimiento de la cámara y el otro para detectar los objetos del primer plano. Para poder realizar ambas tareas ha sido necesario desarrollar un procedimiento que transforme e interpole las matrices de covarianza de las componentes de la mixtura gaussiana conforme la cámara se mueve. También proponemos un método para la extrapolación del fondo cuyo objetivo consiste en generar un nuevo modelo de mixtura para las regiones no vistas anteriormente, es decir, aquellas que no corresponden a ninguna región del modelo actual, en adelante las denominaremos regiones desconocidas. Dado que nuestro modelo no es panorámico, la aparición de regiones desconocidas es inherente al movimiento de la cámara. Nuestra propuesta es comparada con varias alternativas que pertenecen al estado del arte, logrando resultados competitivos tanto desde el punto de vista cualitativo como cuantitativo.

Este capítulo está organizado de la siguiente manera. La Sección 7.2 presenta el método de detección del primer plano propuesto. En la Sección 7.3 se exponen algunos resultados experimentales para demostrar la capacidad de nuestra propuesta para gestionar escenas complejas. Las características y propiedades más destacadas de nuestra propuesta son discutidas en la Sección 7.4. Por último, la Sección 7.5 está dedicada a las conclusiones.

## 7.1. Introducción

En general los algoritmos de detección del primer plano se basan en que el fondo y los objetos del primer plano tienen rasgos diferentes, lo que les permite construir un

modelo del fondo e identificar los píxeles o regiones del primer plano, además asumen que el fondo de la escena no cambia de manera significativa a lo largo del tiempo y que no se desplaza. Sin embargo esta última condición raramente se da en la práctica incluso en cámaras estáticas, por ejemplo, el viento u otros factores externos pueden hacer vibrar la cámara, lo que hace que el fondo se desplace. Este tipo de problemas se denomina *jitter* y es muy común en cámaras de videovigilancia que están en las calles o autovías. Por otro lado, debido a la propagación y avance de la tecnología de videovigilancia es cada vez más habitual encontrar cámaras no estáticas como las de tipo PTZ que pueden cubrir áreas más amplias gracias a que se desplazan tanto en el plano horizontal como en el vertical y además pueden hacer zoom. Además, los algoritmos de detección del primer plano también pueden ser útiles para segmentar el vídeo generado por cámaras portátiles o incluso cámaras que se mueven de posición, denominado egomovimiento. De nuevo los métodos de segmentación habituales para cámaras estáticas no son capaces de funcionar correctamente en estas situaciones.

**Figura 7.1:** Ejemplos de cámaras no estáticas. De izquierda a derecha: cámara PTZ en interior, cámara PTZ para videovigilancia de un cruce de calles y cámara domo en una zona de ocio. Una cámara domo es esencialmente una cámara PTZ a la que se le añade una cúpula para protegerla y ocultar su orientación. *Fuente Wikipedia.*

Existen varias propuestas que abordan la problemática de la segmentación en secuencias grabadas con cámaras en movimiento. En [192] se usa una representación que combina el color del píxel y unas estructuras espaciales con las que genera el modelo del fondo y del primer plano. Esta idea es ampliada en [238, 239] mediante el seguimiento de las trayectorias de los puntos característicos, lo cual requiere el análisis de varios fotogramas consecutivos. En [240] se necesita un número aún mayor de fotogramas, en esta propuesta se analizan las trayectorias de los puntos a lo largo del tiempo para determinar cuales pertenecen a objetos del primer plano. Este sistema no es de tiempo real dado que las decisiones se toman una vez se ha procesado toda la secuencia. Patwardhan et al. [241] descompone la escena en capas y usa la estimación de máxima verosimilitud para asignar cada píxel a una capa. Sin embargo, los experimentos ponen de manifiesto una limitación importante: únicamente pueden detectarse los objetos que se mueven despacio.

Dentro de las propuestas basadas en el flujo óptico podemos citar el trabajo realizado por Zhang et al. [242], en él se presenta un método que funciona a nivel de objeto. Su planteamiento consiste en reducir el problema a una tarea de segmentación de movimiento asumiendo que los movimientos de los objetos del primer plano y del fondo están asociados a distintos flujos ópticos. Otro ejemplo es el método planteado en [243], el cual usa el flujo óptico para determinar texturas dinámicas, por ejemplo, texturas que cambian a lo largo del tiempo y que están asociadas a objetos del primer plano. También es posible enfocar el problema a nivel de bloque, tal y como se hace en [244], donde se propagan en el tiempo modelos de apariencia de bloque, es decir, al fotograma siguiente; a su vez se relacionan en el espacio, esto es, los nuevos modelos recopilan información de las zonas colindantes en el fotograma anterior. La propuesta de [245] construye un campo aleatorio de Markov de las variables de movimiento asociadas al flujo óptico y a su vez se asocia al color otro campo aleatorio de Markov de las variables de apariencia. Una limitación fundamental de las estrategias basadas en el flujo óptico consiste en que el cálculo del flujo óptico es computacionalmente costoso comparado con otras propuestas. Una idea relacionada es la que se plantea en Mahadevan y Vasconcelos [246], donde la prominencia se define como la diferencia entre los rasgos de una ventana central y sus regiones colindantes. Así pues, los objetos del primer plano son aquellos que maximizan la prominencia. También es posible combinar el flujo óptico con el análisis de trayectorias, como se demuestra en [247]. Una manera alternativa de detectar los objetos del primer plano consiste en segmentar las regiones de la imagen de acuerdo a sus características de flujo óptico, lo cual requiere un paso adicional para etiquetar estas regiones como fondo o primer plano. Esta estrategia es la que se sigue en [248].

Existen métodos que usan parámetros internos de la cámara para calcular el modelo del fondo [249], pero hay que decir que es habitual que esos parámetros no estén disponibles. Además, los movimientos inesperados no son gestionados de forma adecuada en algunas situaciones. En [250, 251] el fotograma de entrada se compara con un mosaico del fondo con el objetivo de obtener una matriz de movimiento y realizar la segmentación del fondo. Desafortunadamente este método es sensible a los errores producidos durante el registro de la imagen debido a los efectos de paralaje, por ejemplo, es habitual que en las fotografías aéreas los objetos altos muestren una superficie distinta conforme se va desplazando la cámara.

Algunas propuestas se basan en construir un modelo panorámico de la escena [252, 253], es decir, ir colocando cada fotograma en la posición adecuada dentro de un modelo global de la escena y después detectar los objetos del primer plano utilizando un método tradicional de segmentación para cámaras estáticas. Los métodos panorámicos sólo pueden aplicarse a cámaras cuyos movimientos están limitados a una escena de tamaño fijo, por ejemplo, aquellas que se usan para grabar eventos deportivos. Además, en estos modelos el tamaño de la escena es determinante ya que de él dependen los requisitos de memoria, pudiendo llegar a ser grandes cuando la escena también lo es. Por otro lado, una posible mejora consiste en refinar la detección del primer plano realizando un paso adicional de segmentación basada en

regiones [254].

Los métodos no panorámicos como Kim et al. [255] pueden ser una buena alternativa para cámaras que se mueven libremente ya que almacenan únicamente el modelo del fondo del fotograma actual. Su mayor desafío radica en cómo reutilizar la información del modelo del fotograma actual para generar el modelo del fotograma siguiente. Habitualmente esto se realiza estimando el movimiento de la cámara, lo cual suele implicar la búsqueda de correspondencias entre los puntos característicos de los fotogramas consecutivos [256].

En general los métodos nos panorámicos pueden tener varias limitaciones, las más habituales son las siguientes: requisitos de memoria elevados, problemas con fondos muy dinámicos, cambios de iluminación, dificultades para gestionar el zoom o la necesidad de inicializar el modelo con fotogramas que sólo contengan fondo.

En este capítulo proponemos un método no panorámico para la detección del primer plano en escenas grabadas con una única cámara. Como veremos, es capaz de superar las limitaciones de algoritmos previos como [255] gracias al uso de un modelo de los píxeles del fondo más realista. Se basa fundamentalmente en el trabajo publicado en [159] y en el modelo propuesto en el Capítulo 5. Dado que ambas propuestas estaban diseñadas para cámaras estáticas, ha sido necesario implementar formas nuevas de pasar la información del fondo desde el fotograma actual al siguiente. Nuestro método no necesita ninguna información acerca de los parámetros internos de la cámara, por lo que es aplicable a una gran variedad de situaciones. Hay que señalar que este método asume que entre dos fotogramas consecutivos existe alguna fracción del modelo del fondo del fotograma precedente que puede ser reutilizado en el fotograma actual, por lo que no está diseñado para funcionar en secuencias con movimientos rápidos de cámara. En este sentido se han implementado mecanismos que permiten superar puntualmente este tipo de situaciones. No obstante, es más adecuado para ser utilizado en cámaras PTZ, compensar los efectos de *jitter* y complementariamente, en secuencias con egomovimiento donde los cambios de fondo no son abruptos.

Hay dos novedades principales en este capítulo respecto de lo que se expone en el Capítulo 5 y de otros trabajos previos:

1. Se ha propuesto un procedimiento para interpolar las matrices de covarianza completas asociadas a los modelos de los píxeles (Subsección 7.2.2). Por lo que hemos podido comprobar, la interpolación de matrices de covarianza completas no ha sido investigada en la literatura correspondiente al modelado del fondo.
2. El seguimiento de la cámara y la detección de objetos del primer plano son problemas distintos con sus propias complejidades. En este capítulo proponemos utilizar dos modelos en cada píxel, uno para cada tarea de manera que cada uno de ellos puede ajustarse a sus objetivos específicos (Subsección 7.2.1).

## 7.2. Método

En esta sección se propone un método destinado a resolver el problema de la detección del primer plano en secuencias grabadas con cámaras en movimiento. Se basa en el trabajo publicado previamente por nuestro grupo de investigación [159] y en el modelo propuesto en el Capítulo 5, los cuales están diseñados para funcionar con cámaras estáticas. Nuestro sistema considera dos modelos probabilísticos de fondo para cada uno de los píxeles del fotograma:

- El primer modelo está pensado para realizar el emparejamiento de puntos característicos entre cada par de fotogramas consecutivos. Los rasgos del modelo del fondo son directamente los valores RGB.
- El segundo modelo se usa para clasificar cada píxel como fondo o primer plano. A diferencia del primer modelo, es necesario elegir un conjunto de rasgos, tal y como se indica en la Subsección 7.2.1.

Ambos modelos de fondo se basan en mixturas probabilísticas de dos componentes, una componente uniforme para el primer plano y una gaussiana multivariable con matriz de covarianzas completa para el fondo, como se describe en la Subsección 7.2.1. Los procedimientos utilizados para el aprendizaje e inicialización de las variables que conforman ambos modelos son análogos a los expuestos en las Subsecciones 2.5.2 y 2.5.5, respectivamente.

Cada vez que se va procesar un nuevo fotograma es necesario estimar la matriz de transformación $\mathbf{H}$. Esta matriz permite realizar la transformación desde el sistema de coordenadas del píxel en el fotograma anterior al nuevo sistema de coordenadas del fotograma a procesar. De este modo se utilizan los modelos de fondo del fotograma anterior para generar los nuevos modelos de fondo en el sistema de coordenadas del fotograma actual (Subsección 7.2.2). Así pues, no se mantiene ningún modelo panorámico. El procedimiento de interpolación para obtener los parámetros de los nuevos modelos de fondo incluye el uso de matrices de correlación para poder producir aproximaciones precisas de las matrices de covarianza de las componentes de mixtura gaussiana.

Los modelos de fondo de los píxeles de las regiones desconocidas pueden inicializarse con ayuda de los modelos de los píxeles que están en el borde del fotograma precedente. Si el píxel desconocido es muy parecido al que hay más cerca en el borde conocido, se copia su modelo. Para decidir si este tipo de inicialización es adecuada para un píxel dado, se ha optado por implementar una decisión bayesiana (Subsección 7.2.3). En caso de que sean parecidos, el modelo del fondo del píxel en cuestión se incializa a un estado neutral.

La Figura 7.2 muestra el diagrama de flujo de nuestra propuesta para un fotograma cualquiera.

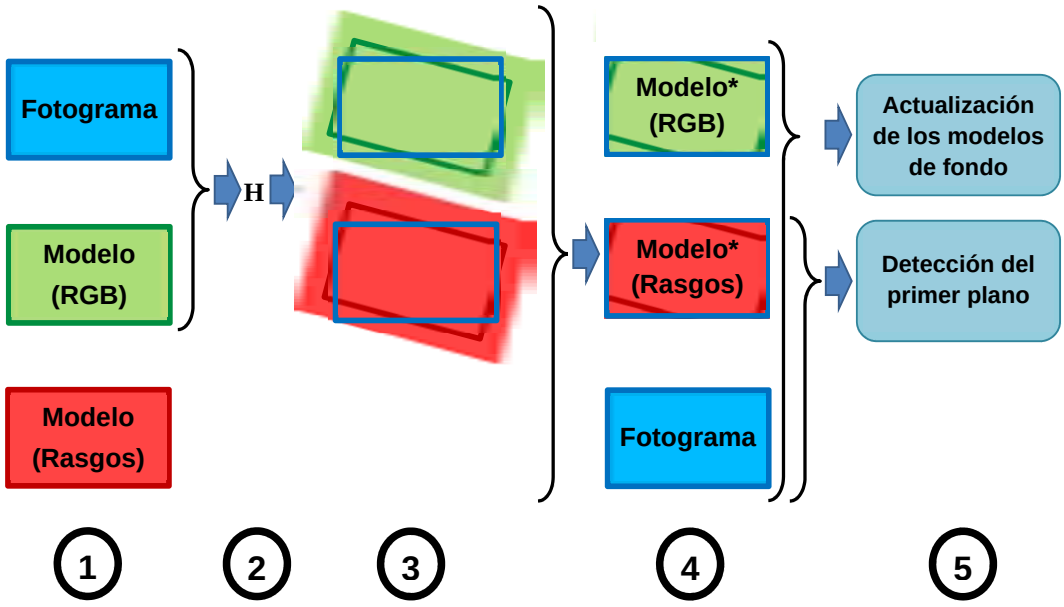



**Figura 7.2:** Diagrama de flujo de nuestra propuesta. "Modelo (RGB)" representa el primer modelo de fondo mientras que "Modelo (Feature)" representa el segundo modelo de fondo. En el primer paso tenemos el fotograma de entrada a procesar y el estado actual del algoritmo representado por "Modelo (RGB)" y " Modelo (Rasgos)". El paso dos estima la matriz de transformación **H** usando el fotograma de entrada y el modelo de fondo "Modelo (RGB)". En el tercer paso se proyectan los modelos de fondo del fotograma anterior en el espacio de coordenadas actual, para ello se usa **H**. El cuarto paso calcula el área que corresponde al fotograma actual en los modelos proyectados, lo que nos determina una región desconocida hasta este momento $\mathcal{U}$. Durante el quinto y último paso, se usa "Modelo* (Rasgos)" para realizar la detección del primer plano y se actualizan ambos modelos.

### 7.2.1. Definición

Como vimos en la Subsección 7.3.3, en el modelado de fondo mediante aproximación estocástica [159] la densidad de probabilidad asociada un píxel $p$ viene dada por la siguiente expresión:

$$\mathrm{p}(\boldsymbol{p}_t) = \pi_{B,t} G\left(\boldsymbol{p}_t \mid \boldsymbol{\mu}_{B,t}(p), \boldsymbol{C}_{B,t}(p) + \boldsymbol{\Psi}\right) + \pi_{F,t} U\left(\boldsymbol{p}_t\right) \tag{7.1}$$

En el caso de las cámaras en movimiento se ha observado que es ventajoso tener dos modelos de mixtura para cada píxel. El primero de ellos, que denotamos como $\mathrm{p}_{RGB}$, usa el espacio RGB y sirve para estimar el movimiento de la cámara. El segundo, $\mathrm{p}_{\mathcal{F}}$ tiene el objetivo de detectar los objetos del primer plano, por lo que emplea rasgos específicos para esta tarea. El motivo de utilizar dos modelos consiste en que hemos probado experimentalmente que el espacio RGB ofrece buenos resultados en términos de emparejamiento de puntos característicos entre fotogramas consecutivos,

mientras que existen otros rasgos más adecuados para la detección de primer plano. Por lo tanto, las responsabilidades o probabilidades de cada clase para un valor observado $\boldsymbol{p}_t$ vienen dadas por la siguiente expresión:

$$\forall i \in \{B, F\}\,,\; R_{i,t} = P\left(i|\boldsymbol{p}_t\right) =$$

$$\frac{\pi_{i,\mathcal{F}} \mathrm{p}_{\mathcal{F}}\left(\boldsymbol{p}_t|i\right)}{\pi_{B,\mathcal{F}} \mathrm{p}_{\mathcal{F}}\left(\boldsymbol{p}_t|B\right) + \pi_{F,\mathcal{F}} \mathrm{p}_{\mathcal{F}}\left(\boldsymbol{p}_t|F\right)} \tag{7.2}$$

donde $R_{B,t}$ y $R_{F,t}$ son las probabilidades a posteriori de que en el instante $t$ el píxel $p$ pertenezca al fondo o al primer plano, respectivamente.

Dado que este capítulo extiende el modelo propuesto para cámaras estáticas en el Capítulo 5, a priori hemos considerado los mismos 24 rasgos que se definen en él (véase Sección 5.3 para un descripción detallada de todos ellos). Como entonces, dado un conjunto de rasgos $\mathcal{F} \subset \{1, ..., 24\}$, la muestra de entrada para un píxel $p$ en el instante $t$ viene dada por la siguiente expresión:

$$\boldsymbol{p}_t = \left(F_{i,t,p}\right)_{i \in \mathcal{F}} \tag{7.3}$$

No obstante, en la Subsección 7.3.3 explicaremos cuáles han sido los rasgos que finalmente hemos usado en los experimentos.

### 7.2.2. Compensación del movimiento de la cámara

Para estimar el movimiento de la cámara entre dos fotogramas consecutivos se emplea un modelo de movimiento proyectivo de 8 parámetros, como se hace habitualmente en la literatura [254, 255]:

$$\boldsymbol{x}\left(t\right) = \mathbf{H}\left(t-1\right)\boldsymbol{x}\left(t-1\right) \tag{7.4}$$

donde $\boldsymbol{x}\left(t\right) = \left(j_t, k_t, 1\right)$ y $\boldsymbol{x}\left(t-1\right) = \left(j_{t-1}, k_{t-1}, 1\right)$ son las coordenadas afines del píxel de un cierto objeto o rasgo en el instante $t$ y $t-1$, respectivamente. $\mathbf{H}$ es una matriz real de tamaño $3 \times 3$ que transforma las coordenadas de los píxeles desde el sistema de coordenadas en el instante $t-1$ al del instante $t$. La matriz $\mathbf{H}$ es estimada extrayendo y emparejando rasgos de dos fotogramas consecutivos mediante el método SURF y después obteniendo un valor consensuado mediante el algoritmo RANSAC. Estos procedimientos son estándar para realizar la compensación de movimiento de las cámaras [257, 258, 259, 260, 261].

Tras calcular la matriz $\mathbf{H}$, debemos transferir la información de los modelos de los píxeles desde el espacio de coordenadas del fotograma anterior, al espacio del

fotograma actual. Algunos de los parámetros de la mixtura pueden transferirse directamente, en concreto los siguientes:

$$\hat{\pi}_{\boldsymbol{x}(t),i}\,(t-1) = \pi_{\mathbf{H}^{-1}\boldsymbol{x}(t-1),i}\,(t-1) \tag{7.5}$$

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{x}(t),i}\,(t-1) = \boldsymbol{\mu}_{\mathbf{H}^{-1}\boldsymbol{x}(t-1),i}\,(t-1) \tag{7.6}$$

$$\hat{\mathbf{m}}_{\boldsymbol{x}(t),i}\,(t-1) = \hat{\pi}_{\boldsymbol{x}(t),i}\,(t-1)\,\hat{\boldsymbol{\mu}}_{\boldsymbol{x}(t),i}\,(t-1) \tag{7.7}$$

donde $i \in \{B, F\}$ y los valores en el segundo miembro de las ecuaciones para los valores no enteros de $\mathbf{H}^{-1}\boldsymbol{x}$ se obtienen mediante interpolación lineal. Nótese que el primer subíndice (Ecuaciones 7.5-7.7) hace referencia a las coordenadas del píxel en cuestión.

Sean $\omega_q$ los pesos de la interpolación lineal, donde $q \in \{1, ..., Q\}$ y $Q$ es el número de píxeles que contribuyen al valor interpolado. De este modo la interpolación lineal del vector de medias viene dada por la siguiente expresión:

$$\hat{\boldsymbol{\mu}} = \sum_{q=1}^{Q} \omega_q \boldsymbol{\mu}_q = \sum_{q=1}^{Q} \omega_q E\left[\boldsymbol{p}_t | q\right] \tag{7.8}$$

donde $\boldsymbol{\mu}_q$ son los vectores de medias de los píxeles que contribuyen en la interpolación.

Por otro lado, la matriz de covarianza $\boldsymbol{C} = E\left[(\boldsymbol{p}_t - \boldsymbol{\mu})\,(\boldsymbol{p}_t - \boldsymbol{\mu})^T\right]$ no puede ser interpolada directamente de forma lineal ya que sus elementos son relativos al vector de medias $\boldsymbol{\mu}$ y dicho vector varía de un píxel a otro, por lo que la estimación sería muy poco fiable [262, 263]. De hecho si intentamos interpolar directamente las matrices de covarianza tenemos lo siguiente:

$$\sum_{q=1}^{Q} \omega_q \boldsymbol{C}_q = \sum_{q=1}^{Q} \omega_q E\left[\left(\boldsymbol{p}_t - \boldsymbol{\mu}_q\right)\left(\boldsymbol{p}_t - \boldsymbol{\mu}_q\right)^T | q\right] \tag{7.9}$$

lo cual no es la matriz de covarianza interpolada porque las covarianzas se están calculado respecto de los vectores de medias de los píxeles que contribuyen en la interpolación $\boldsymbol{\mu}_q$ y no respecto del vector de medias interpolado $\hat{\boldsymbol{\mu}}$.

La solución que proponemos consiste en interpolar linealmente las matrices de correlación $\boldsymbol{R} = E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T\right]$, denominadas también matrices de covarianza no centradas:

$$\hat{\boldsymbol{R}} = \sum_{q=1}^{Q} \omega_q \boldsymbol{R}_q = \sum_{q=1}^{Q} \omega_q E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T | q\right] \tag{7.10}$$

A continuación la matriz de covarianza interpolada puede obtenerse de la siguiente manera:

$$\hat{\boldsymbol{C}} = \sum_{q=1}^{Q} \omega_q E\left[\left(\boldsymbol{p}_t - \hat{\boldsymbol{\mu}}\right)\left(\boldsymbol{p}_t - \hat{\boldsymbol{\mu}}\right)^T |q\right] =$$

$$\sum_{q=1}^{Q} \omega_q E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T - \boldsymbol{p}_t \hat{\boldsymbol{\mu}}^T - \hat{\boldsymbol{\mu}} \boldsymbol{p}_t^T + \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T |q\right]$$

Mediante las propiedades de la esperanza matemática y la definición del vector de medias tenemos que:

$$\hat{\boldsymbol{C}} = \sum_{q=1}^{Q} \omega_q \left(E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T |q\right] - E\left[\boldsymbol{p}_t |q\right] \hat{\boldsymbol{\mu}}^T - \hat{\boldsymbol{\mu}} E\left[\boldsymbol{p}_t^T |q\right] + \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T\right) =$$

$$\sum_{q=1}^{Q} \omega_q \left(E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T |q\right] - \boldsymbol{\mu}_q \hat{\boldsymbol{\mu}}^T - \hat{\boldsymbol{\mu}} \boldsymbol{\mu}_q^T + \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T\right) \tag{7.11}$$

Por lo que aplicando la Ecuación 7.8:

$$\hat{\boldsymbol{C}} = \left(\sum_{q=1}^{Q} \omega_q E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T |q\right]\right) - \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T - \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T + \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T =$$

$$\left(\sum_{q=1}^{Q} \omega_q E\left[\boldsymbol{p}_t \boldsymbol{p}_t^T |q\right]\right) - \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T \tag{7.12}$$

Y por último, si aplicamos la Ecuación 7.10:

$$\hat{\boldsymbol{C}} = \hat{\boldsymbol{R}} - \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T$$

que expresa la matriz de covarianza interpolada en términos de la matriz de correlación interpolada y el vector de medias interpolado.

Dadas estas consideraciones, la matriz de covarianza para los modelos de los píxeles en el instante $t$ se calcula de la siguiente forma:

$$\boldsymbol{R}_{\boldsymbol{x}(t-1),B}\left(t-1\right) =$$

$$\boldsymbol{C}_{\boldsymbol{x}(t-1),B}\left(t-1\right) + \boldsymbol{\mu}_{\boldsymbol{x}(t-1),B}\left(t-1\right) \boldsymbol{\mu}_{\boldsymbol{x}(t-1),B}\left(t-1\right)^T \tag{7.13}$$

$$\hat{\boldsymbol{R}}_{\boldsymbol{x}(t),i}\left(t-1\right) = \boldsymbol{R}_{\mathbf{H}^{-1}\boldsymbol{x}(t-1),i}\left(t-1\right) \tag{7.14}$$

$$\hat{\boldsymbol{C}}_{\boldsymbol{x}(t),B}\left(t-1\right) =$$

$$\hat{\boldsymbol{R}}_{\boldsymbol{x}(t),B}\left(t-1\right) - \hat{\boldsymbol{\mu}}_{\boldsymbol{x}(t),B}\left(t-1\right)\hat{\boldsymbol{\mu}}_{\boldsymbol{x}(t),B}\left(t-1\right)^{T} \tag{7.15}$$

$$\hat{\mathbf{M}}_{\boldsymbol{x}(t),B}\left(t-1\right) = \hat{\pi}_{\boldsymbol{x}(t),B}\left(t-1\right)\hat{\boldsymbol{C}}_{\boldsymbol{x}(t),B}\left(t-1\right) \tag{7.16}$$

Para evitar transformaciones erróneas que no se corresponden con la realidad, se han definido dos tipos de errores: leves y severos. Los errores leves tienen lugar cuando el tamaño del modelo antes de la proyección es o bien muy pequeño o bien muy grande respecto del modelo proyectado. Sean $L$ y $W$, respectivamente, el alto y el ancho del modelo del fotograma anterior proyectado en las coordenadas actuales. Por otro lado, sean $l$ y $w$ el alto y el ancho del modelo antes de la proyección, es decir, el alto y el ancho del fotograma de entrada o lo que es lo mismo $fil$ y $col$, respectivamente. Un error leve tiene lugar cuando la siguiente condición es satisfecha:

$$(\lambda_1 l < L < \lambda_2 l) \vee (\lambda_1 w < W < \lambda_2 w) \tag{7.17}$$

donde $\lambda_2 < \lambda_1$. Se ha probado experimentalmente que $\lambda_1 = 2$ y $\lambda_2 = 0.6$ son valores adecuados para los vídeos considerados en este capítulo y posiblemente para la mayoría de aquellos que no tengan movimientos de cámara demasiado bruscos.

Cuando tiene lugar un error leve, se genera una nueva matriz $\mathbf{H}$ utilizando rasgos distintos. Si se producen 10 errores leves consecutivos, se considera un error severo. Cuando se produce un error severo, el fotograma actual se ignora y no se modifica ningún modelo. Si tienen lugar 3 errores severos consecutivos, ambos modelos son reiniciados al valor del fotograma actual.

### 7.2.3. Extrapolación del fondo

Al tratarse de un modelo no panorámico, conforme se mueve la cámara es inevitable que en el campo de visión aparezcan partes de la escena que no aparecían en el fotograma anterior, mientras que otras dejan de ser visibles. Esta circunstancia hace necesario diseñar un procedimiento que genere nuevos modelos de mixtura para los píxeles que conforman las regiones no vistas previamente, también denominadas regiones desconocidas y a las que denotaremos como $\mathcal{U}$. Nuestra propuesta realiza la inicialización en función de la probabilidad de que el píxel en cuestión, $p \in \mathcal{U}$, tenga características similares a las del borde de la región conocida, $\mathcal{K}$. Para ello se considera el píxel $q \in \mathcal{K}$ más cercano a $p$ y se compara la probabilidad que tiene el vector de rasgos $\boldsymbol{p}_t$ de pertenecer al fondo siguiendo el modelo de mixtura del píxel $q$. En esta comparación se considera además un cierto umbral de clasificación$\tau \in [0, 1]$:

$$P_q\left(B|\boldsymbol{p}_t\right) > \tau \tag{7.18}$$

El uso de umbrales de clasificación ajustables es una técnica habitual en el aprendizaje automático [264, 265]. Si la condición 7.18 se satisface, el modelo ya entrenado de $q$ será copiado al que modela al píxel $p$. La idea que subyace es que en este caso lo más probable es que el píxel $p$ sea una porción de la escena de fondo cuyas características son similares a las del píxel $q$. En caso contrario, se asume que el píxel $p$ es primer plano o una parte del fondo que no se parece al píxel más cercano del borde de $\mathcal{K}$, por tanto su modelo se inicia a un estado neutral, tal y como recogen las siguientes expresiones:

$$\pi_B = \pi_F = \frac{1}{2} \tag{7.19}$$

$$\boldsymbol{\mu}_B = \boldsymbol{p}_t \tag{7.20}$$

$$\boldsymbol{\mu}_F = \left(\frac{e_h + d_h}{2}\right)_{h\in\{1,\ldots,D\}} \tag{7.21}$$

$$\boldsymbol{C}_B = \boldsymbol{\Psi} \tag{7.22}$$

Si la cámara no se mueve demasiado rápido o bien el fondo es homogéneo, es habitual que muchos píxeles satisfagan esta condición (7.18), por lo que se reutiliza una cantidad significativa de información de un fotograma al siguiente. En caso contrario, el fondo es completamente nuevo, de modo que no hay reutilización de los modelos de fondo ya existentes.

## 7.3. Resultados experimentales

En esta sección se lleva a cabo un análisis del rendimiento de nuestra propuesta en diferentes situaciones. En todos los experimentos se han puesto a prueba varios métodos pertenecientes al estado del arte. En la Web[1] está disponible el código de nuestra propuesta y algunos vídeos demostrativos que ilustran las salidas de los métodos estudiados, en todos ellos se ha utilizado la mejor configuración en términos de F-medida.

La estructura de esta sección es la siguiente. En primer lugar, en la Subsección 7.3.1, presentamos los métodos competidores. En la Subsección 7.3.2 exponemos el conjunto de secuencias que ha sido utilizado para llevar a cabo las pruebas descritas.

[1] http://www.lcc.uma.es/ %7Eezeqlr/nonpan/nonpan.html

Después, en la Subsección 7.3.3, mostramos el conjunto de parámetros utilizados en los experimentos. Por último, las Subsecciones 7.3.4 y 7.3.5 están dedicadas a analizar los resultados de forma cualitativa y cuantitativa.

### 7.3.1. Métodos

Hemos comparado nuestra propuesta frente a cuatro métodos especializados en la detección de movimiento con cámaras no estáticas:

- ViBe [266]: Es un método para la detección de movimiento que no se basa en el uso de funciones de densidad de probabilidad, si bien utiliza los valores de los píxeles cercanos para llevar a cabo la detección. Los autores han colaborado con nosotros proporcionándonos una versión ejecutable y parametrizable de su método. Existen diferentes sitios web donde encontrar ejemplos de uso [2] [3]. Estos sitios nos permiten conocer la gran cantidad de situaciones en las que es aplicable con resultados altamente satisfactorios. Este método se encuentra en un estado que permite su aplicación real, siendo posible encontrar algunos de ello.
- NP-Kim [255]: Al igual que nuestra propuesta, se trata de un método no panorámico para la detección de movimiento. Como ha ocurrido con ViBe, los autores han colaborado activamente con nosotros, si bien en este caso sólo nos han proporcionado las salidas que obtiene su método en nuestras secuencias de prueba.
- FVS [267]: Es un método basado en el flujo óptico. A priori está diseñado para detectar movimiento en cualquier tipo de situación. El método se divide en una serie de etapas durante las cuales la segmentación va siendo refinada. La parte novedosa radica en la primera etapa en donde se estiman qué píxeles caen dentro de los objetos, para ello utiliza información de los bordes en fotogramas consecutivos. Es posible obtener una versión ejecutable de este método en la Web[4].
- MoSeg [268]: También es un método basado en el flujo óptico y, a diferencia de nuestra propuesta y de FVS, utiliza información de una ventana de tiempo grande, del orden de los cientos de fotogramas. Al igual que con FVS, es posible obtener una versión ejecutable en la Web [5].

La implementación de nuestra propuesta ha sido realizada utilizando Matlab en combinación con ficheros MEX para aquellas partes más demandantes de tiempo de CPU. No se ha aplicado ninguna etapa de post-procesado. Hemos utilizado la versión 2.4.8 de la biblioteca OpenCV y la 4.2 de TBB (*Threading Building Blocks*).

[2] http://www.vibeinmotion.com/Home.aspx
[3] http://www2.ulg.ac.be/telecom/research/vibe/
[4] http://groups.inf.ed.ac.uk/calvin/FastVideoSegmentation/
[5] http://lmb.informatik.uni-freiburg.de/resources/binaries/

### 7.3.2. Secuencias

A la hora de segmentar secuencias grabadas con cámaras de videovigilancia es habitual que dichas cámaras experimenten pequeños movimientos bruscos en el plano horizontal y vertical, que si bien sólo desplazan la escena unos pocos píxeles, lo hacen en muy poco tiempo. Nosotros los denominaremos vibraciones, en la literatura suelen denominarse *jitter*. Esta problemática no se circunscribe al ámbito de la cámaras en movimiento ya que es normal que lo sufran las cámaras de videovigilancia de exteriores incluso aunque estas sean estáticas. Por ejemplo, las cámaras que graban calles o autovías pueden verse afectas por ráfagas de viento como consecuencia de las condiciones climatológicas o del paso de vehículos a gran velocidad.

Para estudiar este tipo de situaciones hemos utilizado cuatro secuencias ampliamente conocidas en el ámbito de la segmentación de fondo debido a que están incluidas en el repositorio Changedetection.net [149]. Las características más destacadas de dichas secuencias son las siguientes:

- Traffic: Muestra una carretera donde van apareciendo vehículos que se desplazan por la escena a gran velocidad. Esto evita algunas dificultades habituales como las que producen los objetos del plano parados, sin embargo es una escena muy compleja ya que los movimientos de cámara son muy fuertes y numerosos, tanto es así que en ocasiones la imagen queda desenfocada.
- Sidewalk y boulevard: Son secuencias similares, ambas representan avenidas en donde van apareciendo personas y vehículos. En sidewalk se define una región de interés que sólo recoge la parte inferior izquierda de la escena. Una de las dificultades características de esta secuencia consiste en que hay una persona que permanece prácticamente parada durante la mayor parte de la escena, lo cual ocasiona problemas a los métodos de detección.
- Badminton: Es un vídeo que recoge un partido de bádminton en donde la cámara vibra ligeramente y los jugadores no se mueven especialmente rápido. Las líneas de la pista pueden ser problemáticas para algunos métodos ya que tienen un color muy diferente al del resto de la cancha.

Si bien las vibraciones en las cámaras de videovigilancia son un problema frecuente, nuestro método ha sido diseñado con un objetivo más amplio. En concreto es capaz de segmentar secuencias grabadas con cámaras con libertad de movimiento. Para probar este extremo hemos considerado secuencias grabadas con cámaras PTZ y portátiles. En este sentido se han probado cinco secuencias que describimos a continuación:

- Campus1 y campus2: Pertenecen al repositorio de seguimiento de la Universidad de California (*UC Santa Barbara tracking repository*) [269] y en él se denominan *user19_2* y *bren_user07-02*. Están disponibles en la Web[6]. Campus1 es especialmente compleja debido a que en ella se producen cambios de

[6]http://www.lcc.uma.es/ %7Eezeqlr/nonpan/nonpan.html

iluminación abruptos y el fondo a veces es uniforme, lo que puede dificultar la búsqueda de puntos característicos entre fotogramas. Ambas secuencias han sido grabadas con cámaras PTZ y aparecen objetos del primer plano tanto lejos como cerca de la cámara.

- Campus3: Se denomina *seqD* en el repositorio BoBoT - *Bonn Benchmark on Tracking* [270, 271] y puede descargarse del sitio web donde se aloja el repositorio[7]. Es un ejemplo de secuencia con objetos cercanos y problemas de camuflaje. En ella aparece una persona que se va moviendo hacia un lado de la escena y la cámara lo va siguiendo.
- Tennis: Presenta una escena donde una persona está jugando al tenis y se caracteriza por el uso de zoom. Está disponible en el repositorio *Freiburg-Berkeley Motion Segmentation Dataset*[8] [272, 273].
- Woman: Es un ejemplo de vídeo casual subido a YouTube. Ha sido grabado con una cámara portátil y se producen desplazamientos de cámara en el plano horizontal. Está disponible en la Web [9].

Se han empleado los fotogramas *ground truth* que proporcionan los repositorios originales en el caso de las secuencias de vibraciones de cámara. Por otro lado, en el segundo conjunto de secuencias hemos tenido que segmentar manualmente dichos fotogramas ya que no hay ninguno publicado. Para ser lo más justos posible, hemos usado una frecuencia de muestreo constante en cada secuencia a la hora de seleccionar los fotogramas *ground truth*. Dichos fotogramas están disponibles en la Web [10].

### 7.3.3. Selección de parámetros

La cantidad de valores probados en cada uno de los parámetros considerados ha sido similar. Sin embargo en el caso de NP-Kim no hemos probado ninguna configuración porque no disponemos de código fuente ni ejecutable. En todos los demás casos se han probado los valores recomendados tanto por la documentación como por los artículos donde han sido publicados. También han sido elegidos aquellos valores que han demostrado buenos resultados experimentalmente. De este modo el conjunto de configuraciones probadas para cada método se muestra en el Cuadro 7.1.

El número máximo de iteraciones del paso de refinamiento del método FVS es 4, tal y como recomienda la documentación. No obstante, en algunas secuencias este método termina antes de la cuarta iteración debido a un fallo durante la factorización de Cholesky causado por matrices no definidas positivas.

[7] http://www.iai.uni-bonn.de/~kleind/tracking/
[8] http://lmb.informatik.uni-freiburg.de/resources/datasets/sequences.en.html
[9] http://www.lcc.uma.es/%7Eezeqlr/nonpan/panning_example.mp4
[10] http://www.lcc.uma.es/%7Eezeqlr/nonpan/nonpan.html

| Método | Parámetros |
|---|---|
| ViBe | Muestras Nb, $n = \{20, 40, 60\}$ |
| | Umbral de emparejamiento, $t = \{10, 20, 30\}$ |
| | Número de emparejamiento, $s = \{1, 2, 4\}$ |
| | Tasa de submuestreo, $f = \{8, 16, 32\}$ |
| FVS | Fundido, $F = \{0.0005, 0.0001, 0.00005\}$ |
| | Pesos espaciales, $w_S = \{1000, 5000, 10000, 15000, 20000\}$ |
| | Pesos temporales,$w_T = \{500, 2000, 4000, 7000, 10000\}$ |
| MoSeg | Muestras, $S$=$\{4, 8, 16\}$ |
| | Peso de las trayectorias, $T = \{40, 60, 80\}$ |
| Propuesta | Tamaño del paso,$\varepsilon = \{0.002, 0.01, 0.02, 0.03\}$ |
| | Umbral, $\tau = \{0.999, 0.9995, 0.9999\}$ |
| | Rasgos, $\mathcal{F} = \left\{ \begin{array}{c} \{19, 20, 22\}, \{3, 20, 21, 22\}, \\ \{5, 19, 20, 21, 22\} \end{array} \right\}$ |

**Cuadro 7.1:** Valores seleccionados para cada uno de los parámetros considerados, la combinación de todos ellos conforma el conjunto de todas las configuraciones probadas.

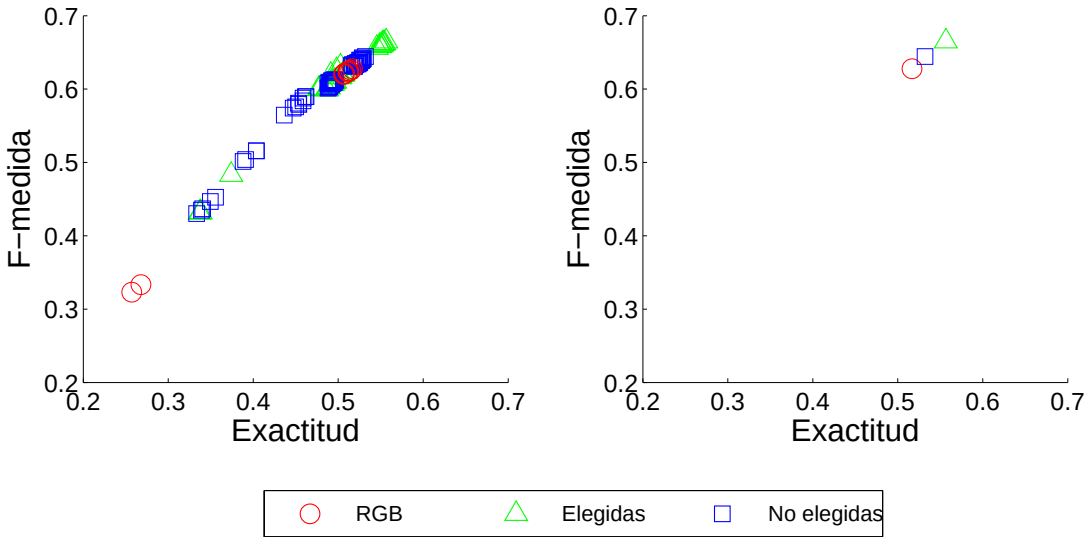



**Figura 7.3:** Rendimiento medio de los conjuntos de rasgos en términos de F-medida frente a exactitud. La gráfica de la izquierda representa el rendimiento medio de cada configuración utilizando todas las combinaciones de $\varepsilon$ y $\tau$ mostradas en el Cuadro 7.1. La gráfica de la derecha muestra aquellas configuraciones que se encuentran en el frente de Pareto. Los círculos rojos representan los rasgos RGB $\{1, 2, 3\}$. Los triángulos verdes corresponden a los rasgos elegidos (véase el Cuadro 7.1). Los cuadrados azules corresponden a los rasgos considerados en el Capítulo 5, pero que no han sido empleados en los experimentos de este capítulo.

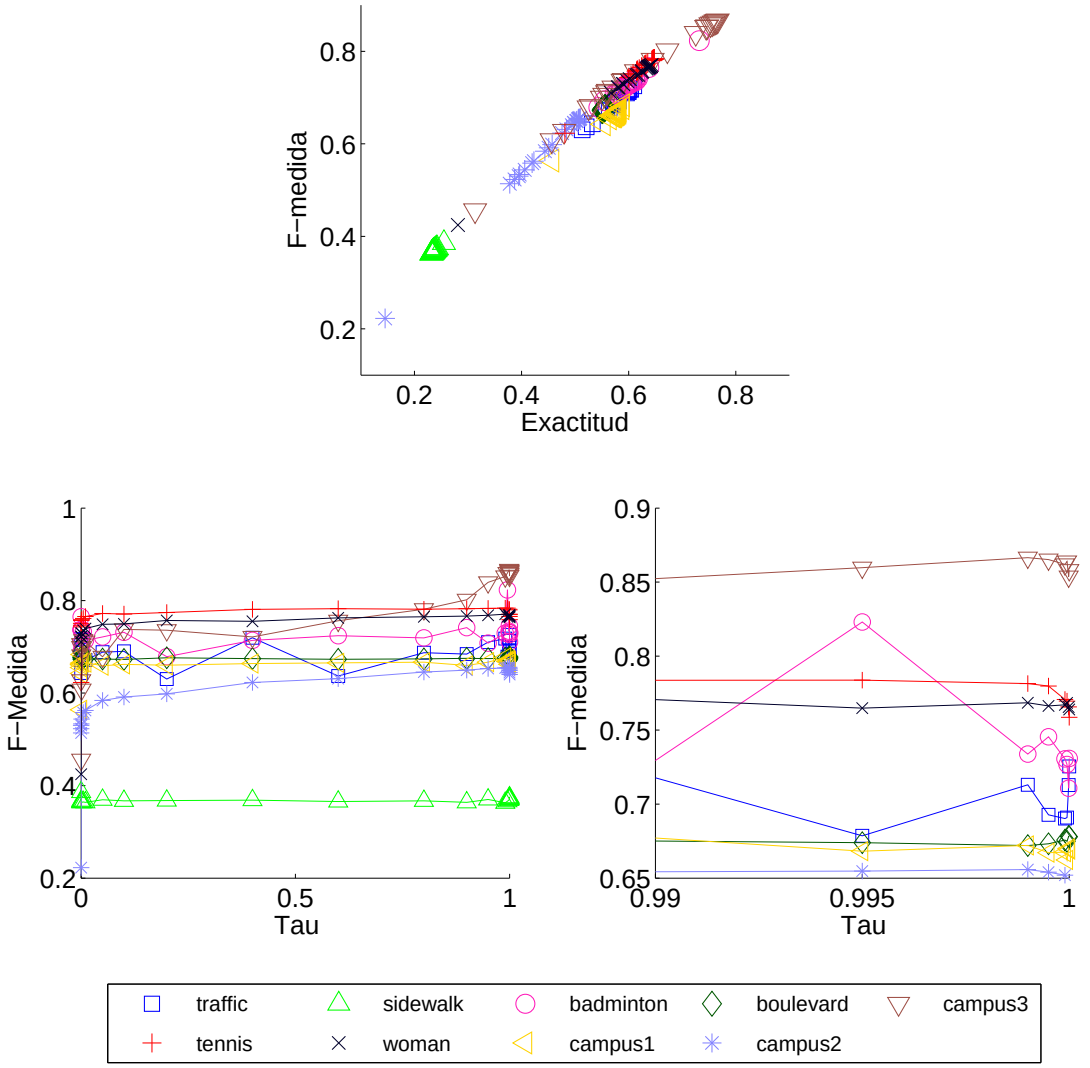



**Figura 7.4:** Rendimiento de nuestra propuesta manteniendo constantes $\varepsilon$ y $\mathcal{F}$ en sus valores óptimos para cada secuencia y variando $\tau$. La primera gráfica muestra el rendimiento en términos de F-medida frente a exactitud para los tres valores de $\tau$ probados. La segunda gráfica representa el rendimiento en términos de F-medida frente a $\tau$. La última gráfica es un detalle de la segunda.

Para probar nuestro método sólo hemos utilizado los tres conjuntos de rasgos que mejor resultado han obtenido en el Capítulo 5. De este modo hemos buscado que el número de valores del parámetro "Rasgos" sea igual o menor que el de los competidores. En la Figura 7.3 podemos ver el rendimiento medio de cada una de las configuraciones consideradas originalmente en el Capítulo 5 al usarlas con el método propuesto. Cada punto representado en esta figura corresponde al rendimiento medio de una configuración en las nueve secuencias probadas. En color verde se han pintado los resultados de los tres mejores rasgos previamente mencionados, en rojo

los correspondientes a RGB y en azul el resto. Como puede observarse el rendimiento que se obtiene al segmentar con los rasgos RGB es menor que el de los demás. Además, los conjuntos de rasgos elegidos superan a los que no han sido elegidos.

En cuanto al tamaño del paso, debemos hacer notar que en el caso del modelo RGB únicamente hemos usado $\varepsilon = 0.2$ ya que es suficientemente grande como para adaptarse al movimiento de la cámara sin emborronar el modelo.

La importancia de $\tau$ se ilustra en la Figura 7.4. El valor $\tau$ no modifica el rendimiento del método en la mayoría de las secuencias con vibraciones, esto se debe a que la proporción de fondo desconocido es pequeña comparado con el de otras secuencias. En las secuencias PTZ y con cámaras portátiles el valor de $\tau$ sí es importante. Teniendo en cuenta que $\tau = 0$ es equivalente a copiar los modelos del borde a ciegas, mientras que $\tau = 1$ significa que no se reutiliza ningún modelo de fondo; la figura viene a poner de relieve la importancia de la extrapolación del fondo, ya que valores bajos $\tau$ se asocian a rendimientos menores y los valores altos hacen que el método obtenga mejores resultados. Como vemos en el detalle que muestra la última gráfica, si fijamos un valor de $\tau = 1$ el rendimiento decrece significativamente, de modo que lo beneficioso es utilizar valores ligeramente inferiores a 1. Esto significa que se debe replicar el borde únicamente si los rasgos son muy parecidos, pero dejando un pequeño margen para la variedad de rasgos. Así pues en los experimentos hemos usado tres valores cercanos a 1, como muestra el Cuadro 7.1.

### 7.3.4. Resultados cualitativos

En esta subsección se lleva a cabo un análisis cualitativo de los resultados. Para ello haremos uso de las Figuras 7.5-7.9 donde se muestran dos fotogramas representativos de cada secuencia. Adicionalmente, en la Web[11] se pueden visualizar los vídeos de salida completos de cada uno de los métodos. Tanto en las figuras como en los vídeos mencionados se ha utilizado la mejor configuración en términos de F-medida para cada una de las secuencias.

Las Figuras 7.5-7.6 están destinadas a las secuencias con vibraciones. A continuación vamos a analizar las ventajas e inconvenientes que tiene cada método en este tipo de situaciones:

- Traffic (primera y segunda columnas de la Figura 7.5): Como mencionamos previamente, la secuencia se caracteriza por unas vibraciones de cámara muy fuertes. En estos casos el método NP-Kim produce una gran cantidad de falsos positivos, como vemos en el fotograma 998 (primera columna). También sufre problemas de camuflaje en las ventanillas de los vehículos porque su color es similar al del asfalto. ViBe comete algunos falsos positivos debido a los bordes de las texturas, pero el problema principal son los falsos negativos que genera al no ser capaz de rellenar los objetos del primer plano. FVS obtiene muchos

[11] http://www.lcc.uma.es/ %7Eezeqlr/nonpan/nonpan.html

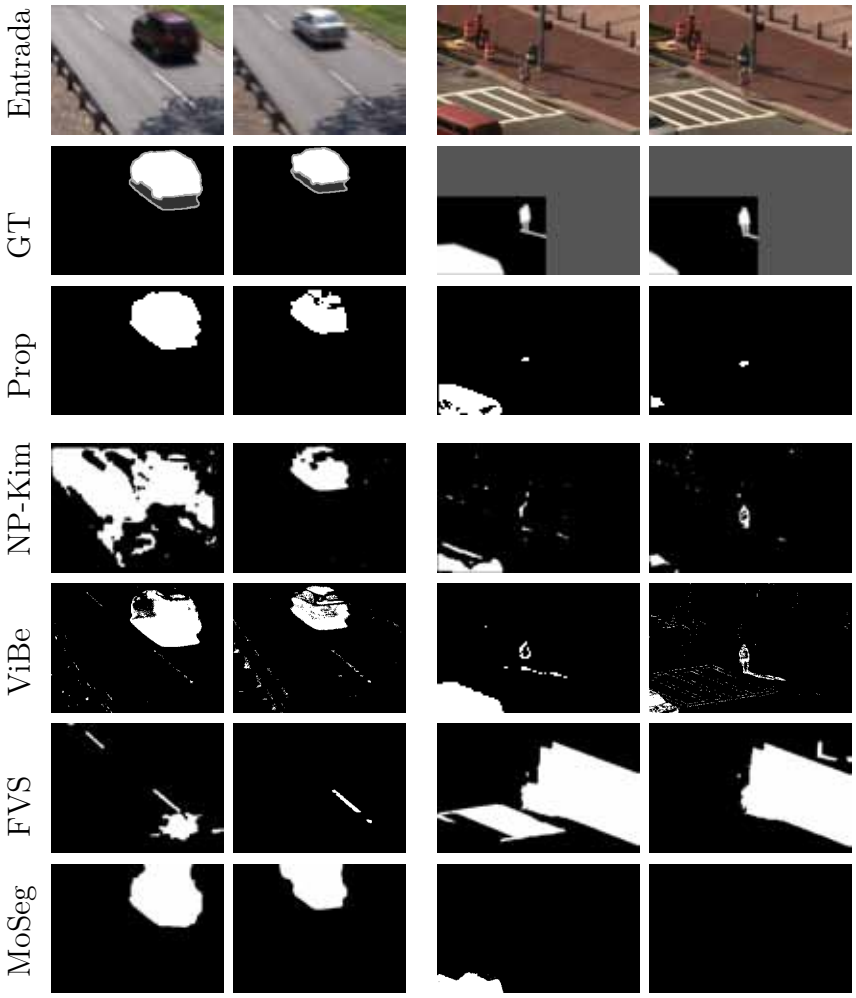



**Figura 7.5:** Salidas obtenidas al aplicar los métodos en las secuencias traffic y sidewalk utilizando las mejores configuraciones en términos de F-medida. De arriba a abajo: fotograma original, ground truth, método propuesto, NP-Kim, ViBe, FVS y MoSeg. De izquierda a derecha: fotogramas 998 y 1385 de la secuencia traffic; fotogramas 830 y 1014 de sidewalk.

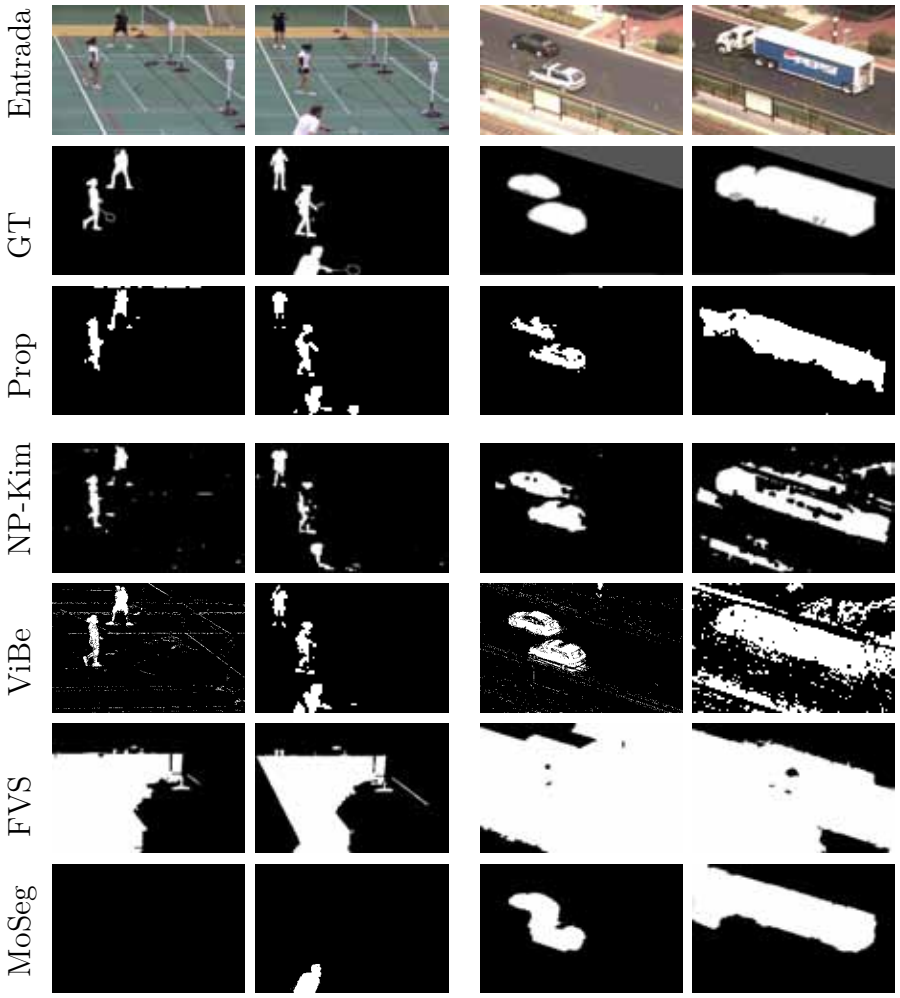



**Figura 7.6:** Salidas obtenidas al aplicar los métodos en las secuencias badminton y boulevard utilizando las mejores configuraciones en términos de F-medida. De arriba a abajo: fotograma original, ground truth, método propuesto, NP-Kim, ViBe, FVS y MoSeg. De izquierda a derecha: fotogramas 916 y 1024 de la secuencia badminton; fotogramas 895 y 2156 de boulevard.

falsos positivos y negativos. Por otro lado, MoSeg rinde adecuadamente, pero produce una cantidad elevada de falsos positivos. En contraste con esto, nuestra propuesta es capaz de obtener buenos resultados en ambos fotogramas, siendo suficientemente robusto como para evitar los problemas de camuflaje y adaptarse a los movimientos bruscos de la cámara

- Sidewalk (columnas tres y cuatro de la Figura 7.5): Si nos centramos en la región de interés definida en la secuencia observamos que la problemática que genera la persona que se detiene al borde del paso de cebra afecta de manera particular tanto a nuestra propuesta como a NP-Kim. Pasado un tiempo ambos métodos subsumen como parte del fondo los píxeles correspondientes a esta persona. Sin embargo, los métodos FVS y ViBe son prácticamente inmunes a esto, si bien FVS comete una gran cantidad de falsos positivos. ViBe también comete el mismo tipo de fallo en los bordes de las texturas, pero es menos acusado que en el caso de FVS. Por su parte, MoSeg es incapaz de detectar la persona y además segmenta pobremente los vehículos que aparecen en escena.
- Badminton (primera y segunda columnas de la Figura 7.6): Nuestra propuesta obtiene muy pocos falsos positivos y sólo algunos falsos negativos. NP-Kim produce grandes cantidades de falsos negativos, como puede observarse tanto en la persona de en medio como en la de abajo. ViBe tiende a cometer falsos positivos en las líneas que hay pintadas en la pista. Por otro lado, FVS es claramente la peor alternativa ya que obtiene demasiados falsos positivos. En cuanto a MoSeg observamos que es incapaz de detectar a la mayoría de personas que aparecen en la escena, si bien su resultado es aparentemente mejor que el de FVS.
- Boulevard (columnas tres y cuatro de la Figura 7.6): En esta secuencia nuestro método genera más falsos negativos que los competidores a causa de los efectos de camuflaje, en particular en el fotograma 895 (tercera columna). No obstante es capaz de producir menos falsos positivos que ViBe y FVS en situaciones con objetos de primer plano grandes, como en el fotograma 2156 (cuarta columna), donde NP-Kim también rinde pobremente. MoSeg logra buenos resultados, pero a cambio de un gran número de falsos positivos, como le ocurría en la secuencia traffic.

A continuación vamos a estudiar las salidas de los métodos en las secuencias grabadas con cámaras PTZ y portátiles. Como podemos ver en las Figuras 7.7-7.9, la situación es muy distinta y las diferencias son mayores que en el caso de las secuencias con vibraciones. ViBe produce una gran cantidad de falsos positivos y negativos, por lo que puede descartarse como opción válida para este tipo de secuencias. Los métodos basados en el flujo óptico tienen un rendimiento irregular que, al igual que el del resto de métodos, analizaremos bajo estas líneas:

- Campus1 (primera y segunda columnas de la Figura 7.7): En general es una secuencia compleja, fundamentalmente a causa de los cambios bruscos de iluminación y a los giros rápidos de cámara. NP-Kim sufre estos dos inconvenien-

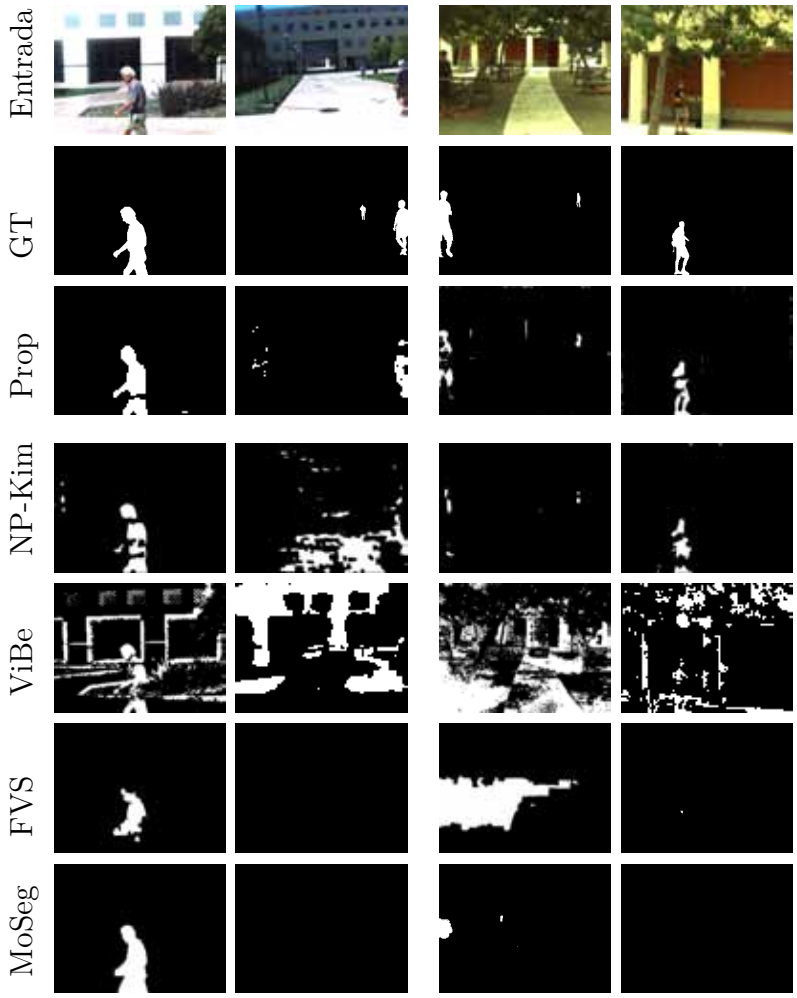



**Figura 7.7:** Salidas obtenidas al aplicar los métodos en las secuencias campus1 y campus2 utilizando las mejores configuraciones en términos de F-medida. De arriba a abajo: fotograma original, ground truth, método propuesto, NP-Kim, ViBe, FVS y MoSeg. De izquierda a derecha: fotogramas 60 y 195 de la secuencia campus1; fotogramas 1498 y 1888 de campus2.

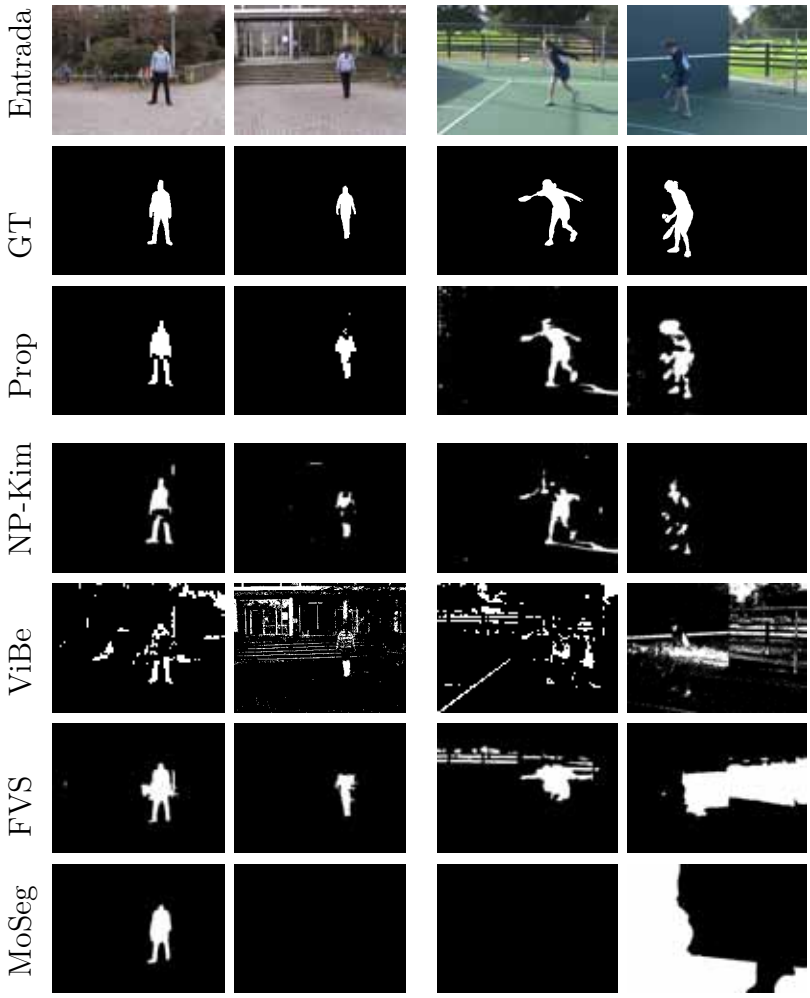



**Figura 7.8:** Salidas obtenidas al aplicar los métodos en las secuencias campus3 y tennis utilizando las mejores configuraciones en términos de F-medida. De arriba a abajo: fotograma original, ground truth, método propuesto, NP-Kim, ViBe, FVS y MoSeg. De izquierda a derecha: fotogramas 576 y 901 de la secuencia campus3; fotogramas 590 y 878 de tennis.

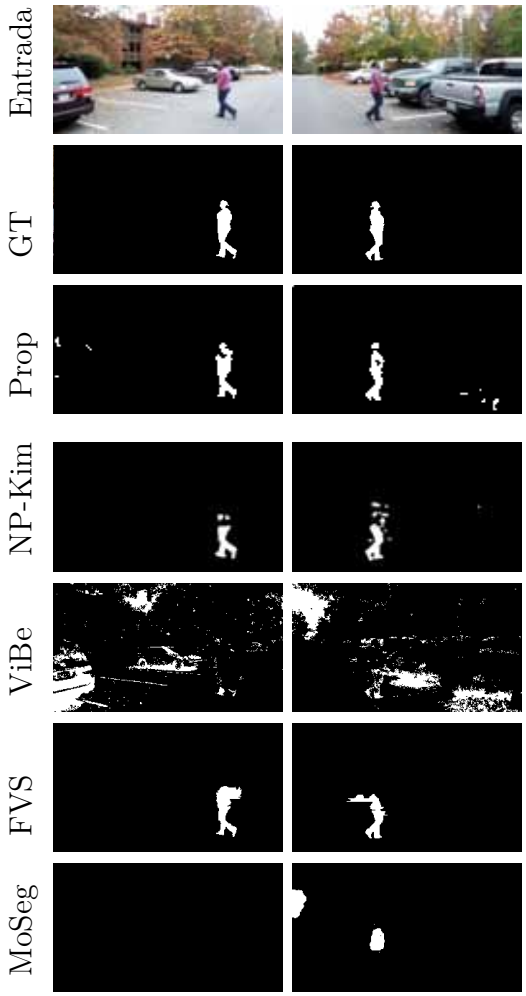



**Figura 7.9:** Salidas obtenidas al aplicar los métodos en la secuencia woman utilizando las mejores configuraciones en términos de F-medida. De arriba a abajo: fotograma original, ground truth, método propuesto, NP-Kim, ViBe, FVS y MoSeg. De izquierda a derecha: fotogramas 250 y 550 de la secuencia woman.

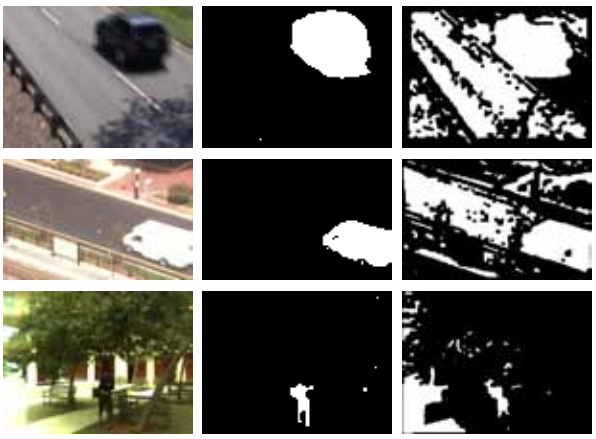



**Figura 7.10:** Ejemplos de situaciones en las que se producen grandes áreas de falsos positivos al utilizar NP-Kim. De arriba a abajo: fotograma 1453 de la secuencia traffic, fotograma 1190 de boulevard y fotograma 1474 de campus2.

tes y como resultado obtiene una suma significativa de falsos positivos, como en el fotograma 195 (segunda columna). Además este método tiene problemas a la hora de rellenar los objetos del primer plano en el fotograma 60 (primera columna). FVS y MoSeg segmentan bien el fotograma 60, pero fallan en el 195, el cual se caracteriza por un movimiento rápido de cámara. A pesar de la complejidad de la secuencia, nuestra propuesta logra un buen resultado en ambos fotogramas.

- Campus2 (tercera y cuarta columnas de la Figura 7.7): Tanto NP-Kim como nuestro método segmentan adecuadamente los objetos distantes, sin embargo NP-Kim obtiene más falsos negativos que nuestra propuesta, particularmente en los objetos cercanos. FVS y MoSeg tienen un rendimiento muy pobre, en ambos casos la suma de falsos negativos y positivos es muy alta.
- Campus3 (primera y segunda columnas de la Figura 7.8): Como ya le ocurría en otras secuencias, NP-Kim es incapaz de superar los problemas de camuflaje. Falla al segmentar el interior de la persona, tanto sus zonas oscuras como las claras. Por otro lado, FVS segmenta bien la escena, comete algunos falsos positivos y negativos, pero el rendimiento es mejor que el que logra en otras secuencias. MoSeg funciona bien en el fotograma 576 (primera columna), pero en el 901 (segunda columna) no es capaz de detectar el primer plano.
- Tennis (tercera y cuarta columnas de la Figura 7.8): Es un vídeo que exhibe una situación muy dinámica con giros de cámara y zoom. NP-Kim lleva a cabo una segmentación pobre debido al elevado número de falsos negativos. Los métodos basados en el flujo óptico obtienen un resultado aún peor. Sin embargo nuestra propuesta es capaz de obtener muy buen resultado, lo que la convierte en la mejor alternativa.

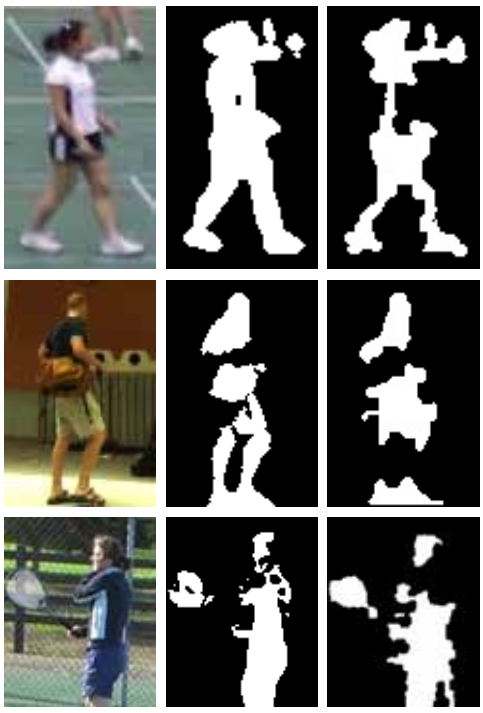



**Figura 7.11:** Detalles de fotogramas en donde los objetos que son segmentados con NP-Kim tienen bordes no suaves. De arriba a abajo: fotograma 927 de la secuencia badminton, fotograma 1888 de campus2 y fotograma 771 de tennis.

- Woman (Figura 7.9): Por último, esta secuencia se caracteriza por el uso de una cámara portátil. NP-Kim y FVS son incapaces de segmentar la cabeza de la persona que aparece en la escena, mientras que nuestra propuesta no tiene grandes dificultades, salvo una pequeña cantidad de falsos positivos que no es significativa.

En resumen, ViBe tiende a cometer un número elevado de falsos positivos tanto en el borde de las texturas como en las secuencias grabadas con cámaras PTZ y portátiles. Además ViBe obtiene muchos falsos negativos al no ser capaz de rellenar adecuadamente los objetos del primer plano. NP-Kim produce grandes regiones de falsos positivos en algunas situaciones complejas, como se pone de manifiesto en la Figura 7.10. En algunas secuencias no es capaz de obtener bordes suaves en los objetos del primer plano, creando bordes que no guardan relación con los objetos reales, en la Figura 7.11 podemos ver algunos ejemplos de este efecto. No obstante, su mayor problema son los falsos negativos, en gran medida debido a los efectos de camuflaje. Las propuestas basadas en el flujo óptico presentan un rendimiento irregular. MoSeg parece funcionar mejor en las secuencias con vibraciones que en las PTZ. Por su lado, FVS es el peor método en las secuencias con vibraciones, pero funciona aceptablemente bien en algunas secuencias grabadas con cámaras PTZ y portátiles. Nuestro método produce algunos falsos positivos, pero lo compensa con

un número muy reducido de falsos negativos. Por todo ello, nuestra propuesta parece superar a los competidores en la mayoría de los vídeos probados. En la siguiente subsección se confirmará este extremo de forma cuantitativa.

### 7.3.5. Resultados cuantitativos

En esta subsección exponemos los resultados desde el punto de vista cuantitativo. Las Figuras 7.12 y 7.13 muestran el rendimiento de cada algoritmo fotograma a fotograma, en la primera figura se ha empleado la mejor configuración en términos de exactitud mientras que en la segunda, la F-medida ha sido el criterio. La Figura 7.14 muestra el rendimiento medio de cada una de las configuraciones consideradas en el Cuadro 7.1. El Cuadro 7.2 lista los valores medios de las mejores configuraciones de acuerdo con la F-medida y la exactitud.

| Secuencia | Método | F-medida | Exactitud | Secuencia | F-medida | Exactitud |
|---|---|---|---|---|---|---|
| traffic | Propuesta | **0.71 ± 0.25** | **0.60 ± 0.24** | campus2 | **0.66 ± 0.17** | **0.51 ± 0.17** |
| | NP-Kim | 0.57 ± 0.26 | 0.44 ± 0.23 | | 0.53 ± 0.21 | 0.39 ± 0.19 |
| | ViBe | 0.68 ± 0.18 | 0.54 ± 0.18 | | 0.08 ± 0.05 | 0.04 ± 0.03 |
| | FVS | 0.32 ± 0.34 | 0.24 ± 0.28 | | 0.24 ± 0.25 | 0.16 ± 0.19 |
| | MoSeg | 0.47 ± 0.41 | 0.40 ± 0.36 | | 0.19 ± 0.33 | 0.16 ± 0.27 |
| sidewalk | Propuesta | 0.37 ± 0.14 | 0.24 ± 0.13 | campus3 | **0.87 ± 0.04** | **0.77 ± 0.06** |
| | NP-Kim | **0.45 ± 0.20** | **0.32 ± 0.18** | | 0.78 ± 0.08 | 0.64 ± 0.10 |
| | ViBe | 0.34 ± 0.21 | 0.23 ± 0.19 | | 0.27 ± 0.09 | 0.16 ± 0.06 |
| | FVS | 0.17 ± 0.12 | 0.10 ± 0.08 | | 0.81 ± 0.06 | 0.69 ± 0.08 |
| | MoSeg | 0.06 ± 0.22 | 0.05 ± 0.19 | | 0.43 ± 0.37 | 0.35 ± 0.32 |
| badminton | Propuesta | **0.81 ± 0.17** | **0.71 ± 0.18** | tennis | **0.78 ± 0.05** | **0.64 ± 0.07** |
| | NP-Kim | 0.68 ± 0.21 | 0.55 ± 0.20 | | 0.67 ± 0.12 | 0.51 ± 0.13 |
| | ViBe | 0.80 ± 0.16 | 0.70 ± 0.17 | | 0.21 ± 0.05 | 0.12 ± 0.03 |
| | FVS | 0.14 ± 0.08 | 0.07 ± 0.04 | | 0.31 ± 0.16 | 0.20 ± 0.11 |
| | MoSeg | 0.22 ± 0.26 | 0.15 ± 0.18 | | 0.23 ± 0.26 | 0.16 ± 0.20 |
| boulevard | Propuesta | 0.68 ± 0.24 | 0.56 ± 0.27 | woman | **0.77 ± 0.14** | **0.64 ± 0.16** |
| | NP-Kim | **0.78 ± 0.18** | **0.67 ± 0.22** | | 0.70 ± 0.15 | 0.56 ± 0.17 |
| | ViBe | 0.65 ± 0.19 | 0.51 ± 0.20 | | 0.10 ± 0.12 | 0.06 ± 0.07 |
| | FVS | 0.14 ± 0.11 | 0.08 ± 0.07 | | 0.66 ± 0.24 | 0.54 ± 0.26 |
| | MoSeg | 0.76 ± 0.32 | 0.70 ± 0.32 | | 0.15 ± 0.23 | 0.11 ± 0.17 |
| campus1 | Propuesta | **0.68 ± 0.33** | **0.59 ± 0.31** | | | |
| | NP-Kim | 0.56 ± 0.32 | 0.46 ± 0.29 | | | |
| | ViBe | 0.23 ± 0.15 | 0.14 ± 0.10 | | | |
| | FVS | 0.46 ± 0.35 | 0.37 ± 0.30 | | | |
| | MoSeg | 0.59 ± 0.40 | 0.52 ± 0.38 | | | |

**Cuadro 7.2:** Resultados de las mejores configuraciones en términos de F-medida y exactitud. La primera y quinta columnas denotan el nombre de la secuencia, la segunda columna indica el nombre del método, la tercera y sexta columnas son los resultados según la F-medida, y por último, la cuarta y séptima columnas son los resultados medidos usando exactitud. El mejor resultado para cada vídeo y medida se ha marcado en negrita.

En primer lugar vamos a analizar el rendimiento de los métodos fotograma por fotograma, es decir, las Figuras 7.12 y 7.13. Como puede observarse, dentro de la categoría de secuencias en las que la cámara experimenta vibraciones, la secuencia traffic es particularmente difícil ya que el rendimiento de todos los métodos en este

| Secuencia | Propuesta | NP-Kim | ViBe | FVS | MoSeg |
|---|---|---|---|---|---|
| traffic | 603408 | 603277 | 585806 | **301097** | 387451 |
| sidewalk | 4094 | 2803 | 8928 | **19** | 1482 |

**Cuadro 7.3:** Error en las sombras (SE) de las mejores configuraciones según F-medida. La primera columna denota el nombre de la secuencia, el resto corresponden al SE obtenido por cada método.

| Método | Exactitud | F-medida |
|---|---|---|
| Propuesta | **0.57*** | **0.69*** |
| NP-Kim | N/A | N/A |
| ViBe | 0.27 | 0.36 |
| FVS | 0.26 | 0.35 |
| MoSeg | 0.20 | 0.24 |

**Cuadro 7.4:** Resultados medios de las cinco mejores configuraciones de cada secuencia y método. La negrita representa que es el mejor método en media y el asterisco que la diferencia es significativa con un nivel de confianza del 99 %.

caso es altamente inestable. De hecho es muy difícil decir cuál es el mejor método. En el caso de badminton aparentemente nuestra propuesta es de las mejores, sin embargo es arriesgado afirmar que se la mejor. Por último, tanto en boulevard como en sidewalk es aún más complicado decantarse por un método, lo único que es evidente es que FVS exhibe un rendimiento muy pobre en ambos casos.

Para tener una idea exacta de qué método es mejor debemos observar el Cuadro 7.2, en él se pone de manifiesto que nuestra propuesta consigue los mejores resultados tanto en traffic como en badminton. En los otros dos casos, es decir, en boulevard y en sidewalk, NP-Kim es la mejor alternativa. No obstante hay que señalar que en sidewalk ningún método consigue un resultado aceptable, ya que todos están por debajo de 0.5 en F-medida. El método ViBe muestra un buen rendimiento en este tipo de secuencias, pero ligeramente peor que el de nuestra propuesta. Por último, ninguno de los métodos basados en flujo óptico destaca por su buen rendimiento. De hecho, MoSeg es el único que tiene un rendimiento razonable en boulevard y traffic, pero no es una alternativa competitiva ya que sus resultados son muy pobres en las otras dos secuencias.

Así pues, es difícil decir cuál es el mejor método en los vídeos donde la cámara vibra. En todo caso, NP-Kim y nuestra propuesta son las dos alternativas que mejores resultados obtienen, ambos son los mejores en dos de las tres secuencias analizadas. No obstante, debemos hacer notar que nuestro método es el único que supera al resto en dos vídeos, quedando en segunda y tercera posición en los otros dos, todo lo cual hace de ella una alternativa muy competitiva.

El Cuadro 7.3 realiza una comparación del número de errores producidos por las sombras (SE) en los dos vídeos cuyos *ground truth* proporcionan información acerca

de las sombras. Según esta medida de rendimiento, el método FVS es el mejor, a pesar de que segmenta pobremente la escena y no es capaz de detectar algunos de los objetos del primer plano ni tampoco las sombras. En este sentido resulta ilustrativo ver las secuencias subidas a la Web[12]. Estos resultados ponen de manifiesto que tanto la F-medida como la exactitud son medidas de rendimiento más adecuadas para comparar el desempeño de los métodos de segmentación. De hecho, el error debido a las sombras ya está incluido en ambas medidas, como podemos ver en sus definiciones en la Subsección 2.4.2.

El análisis cuantitativo del segundo tipo de secuencias, es decir, cámaras PTZ y portátiles, proporciona una idea más clara de las diferencias entre los métodos. Nuestra propuesta supera a los dos competidores principales en la mayoría de las secuencias, como arrojan los resultados fotograma a fotograma de la Figura 7.13. También es interesante destacar que, salvo en campus3, hay fragmentos de las secuencias en los que el rendimiento de NP-Kim y de nuestra propuesta caen de manera significativa, por ejemplo, campus1 GT: 14-19; campus2 GT: 6-10; tennis GT: 15-21 y woman GT: 10-15. No obstante, la robustez de nuestra propuesta es mayor que la de NP-Kim ya que en estos casos, suele conseguir mejores resultados.

De nuevo, si prestamos atención al Cuadro 7.2, en cada secuencia el rendimiento de nuestra propuesta supera en seis o más puntos FM y AC al del mejor competidor. Como apuntamos en la subsección anterior, el método ViBe es el que peores resultados obtiene, siendo incapaz de obtener un rendimiento aceptable en ninguna de las secuencias consideradas. En cuanto a los métodos basados en flujo óptico los resultados no son particularmente buenos, si bien FVS es competitivo en algunas secuencias como campus1, campus3 y woman.

La Figura 7.14 complementa los resultados vistos hasta ahora, cada punto representa el rendimiento medio en las nueve secuencias de una configuración. Con esta figura podemos comprender cuales son las limitaciones de cada método. En la primera fila (FPR frente a FNR) pone de relieve que las diferentes configuraciones de ViBe mejoran FNR a cambio de empeorar su FPR y viceversa. NP-Kim y nuestro método obtiene resultados similares, pero la ventaja de nuestra propuesta consiste en que es capaz de reducir su FNR sin empeorar de forma significativa en FPR. La mayor dificultad de los métodos basado en el flujo óptico son los falsos negativos. MoSeg tiene una tasa excesivamente alta de falsos negativos. FVS mejora en este aspecto, pero empeora su FPR notablemente. En la segunda fija (RC frente a PR) se muestra que ViBe es capaz de conseguir buenas marcas en RC, pero su precisión es baja. Al igual que antes, NP-Kim y nuestra propuesta están muy cerca, aunque hay algunas de nuestras configuraciones que proporcionan un rendimiento más equilibrado entre ambas medidas, acercándose más al punto óptimo $(1, 1)$. En la última fila (FM frente a AC) se confirma que nuestro método es mejor que los competidores considerados, en la segunda columna (frente de Pareto) se ilustra esto de manera más clara.

Para concluir esta subsección, en el Cuadro 7.4 se exponen los resultados medios

[12]http://www.lcc.uma.es/%7Eezeqlr/nonpan/nonpan.html

de las cinco mejores configuraciones de cada secuencia y método. Se ha marcado en negrita el mejor resultado y con un asterisco cuando además las diferencias son significativas con un nivel de confianza del 99 %. La significancia se determinado aplicando la prueba no paramétrica de Friedman con la correspondiente prueba de Dunn. No se incluye NP-Kim por no tener suficientes muestras con las que comparar ya que no proporcionan código ni ejecutable. Como podemos apreciar, nuestra propuesta logra los mejores resultados medios y además la diferencia con el segundo mejor método es estadísticamente significativa.

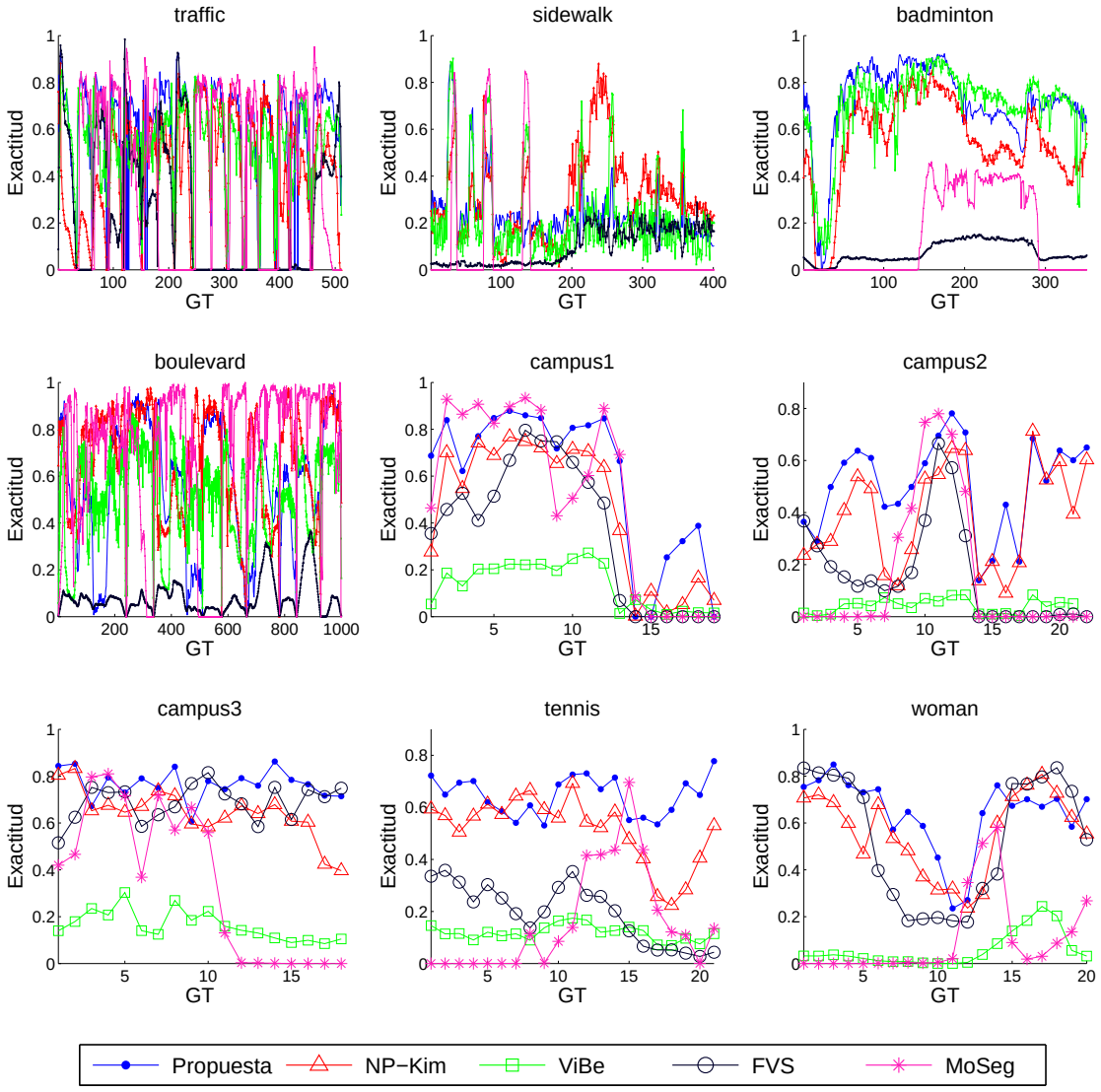



**Figura 7.12:** Rendimiento obtenido en cada fotograma por las mejores configuraciones en términos de exactitud. El eje horizontal muestra el número de ground truth y el vertical la exactitud.

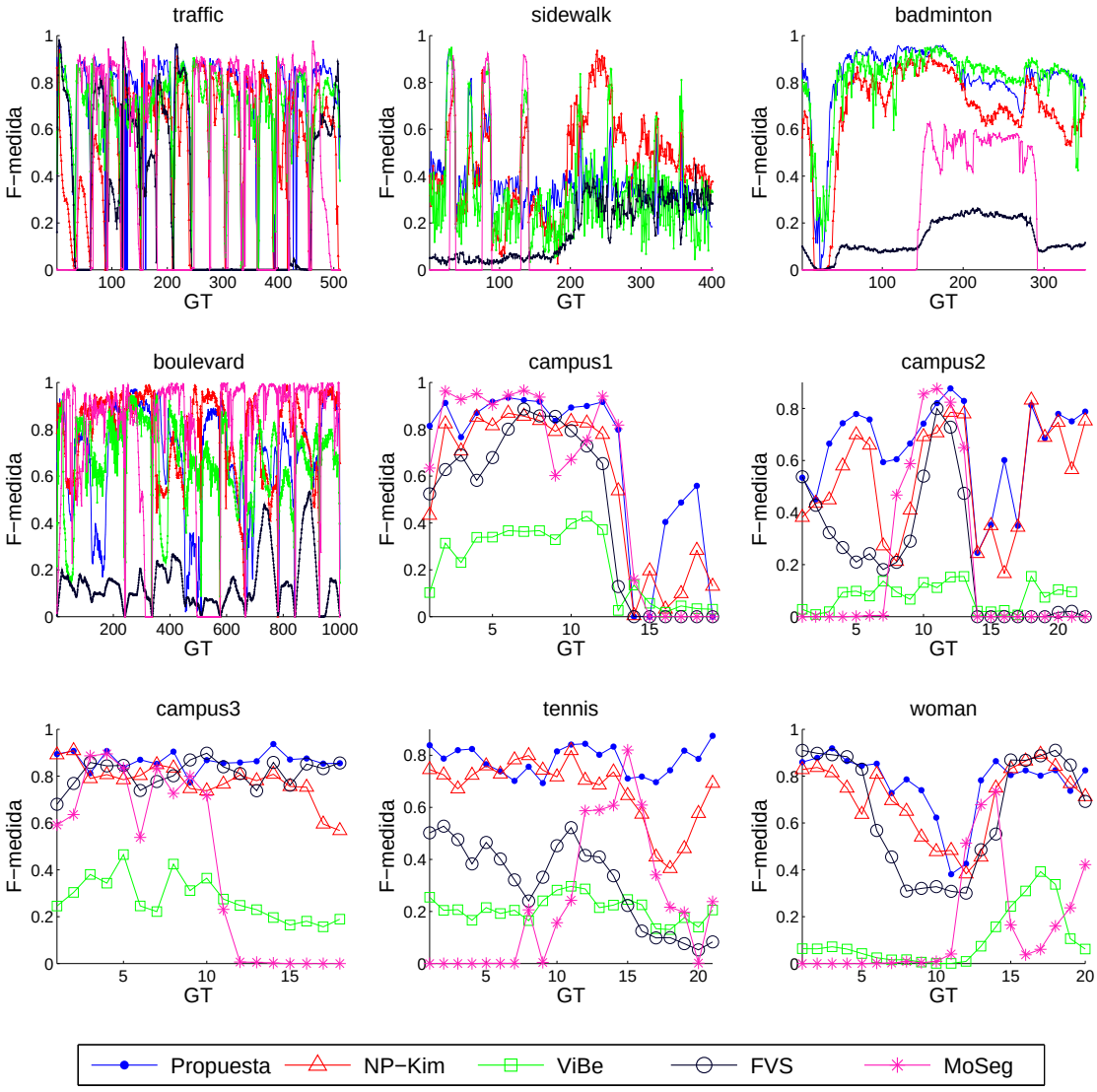



**Figura 7.13:** Rendimiento obtenido en cada fotograma por las mejores configuraciones en términos de F-medida. El eje horizontal muestra el número de ground truth y el vertical la F-medida.

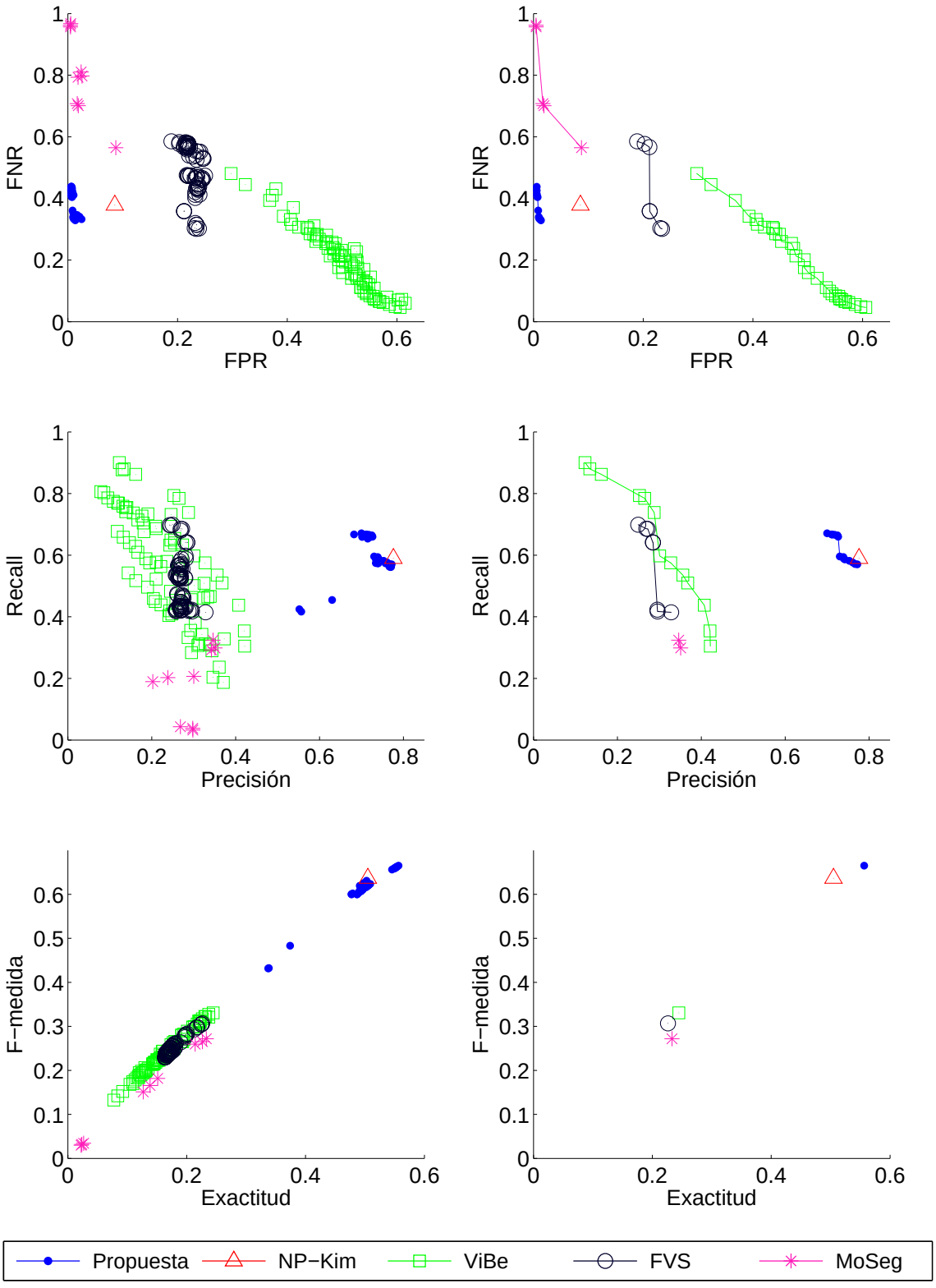



**Figura 7.14:** Rendimiento promedio en las nueve secuencias de todas las configuraciones. La primera columna muestra la tasa de falsos negativos frente a la tasa de falsos positivos, recall frente a precisión y F-medida frente a exactitud. La segunda columna muestra las configuraciones que están en el frente de Pareto.

## 7.4. Discusión

En esta sección vamos a discutir las características más relevantes de nuestra propuesta. En primer lugar hay que destacar que el modelo probabilístico de fondo es más realista que el de otras alternativas no panorámicas previas ya que contiene una matriz de covarianza completa para modelar el fondo y una distribución uniforme para el primer plano (ver Subsección 7.2.1 para más detalles). Por ejemplo, el método publicado en [255] considera una matriz de covarianza esférica, es decir, todas las componentes comparten la misma varianza y no se modela la correlación entre las diferentes componentes de entrada. Como ya se expuso en la Sección 2.5, este tipo de modelo está lejos de ser óptimo, entre otras cosas porque las componentes de entrada suelen estar correladas.

Seguir el movimiento de la cámara y detectar los objetos del primer plano son problemas distintos con sus propias particularidades. En este capítulo proponemos tener dos modelos, uno para cada tarea de modo que se pueden ajustar de forma independiente para lograr sus objetivos específicos (Subsección 7.2). El uso de dos modelos de fondo para el problema de las cámaras en movimiento ya ha sido propuesto en [274], sin embargo en esa propuesta la idea consiste en utilizar un modelo como detector del primer plano y otro como respaldo para reemplazar al primero en caso de que éste sea corrompido. Así pues, no existe división del trabajo que es lo que hacemos en nuestro modelo.

El procedimiento para transformar e interpolar las matrices de covarianza del modelo de fondo en situaciones con movimiento de cámara (Subsección 7.2.2) permite el uso de gaussianas con covarianza completa. Esto es una novedad fundamental de nuestro método, ya que hemos comprobado experimentalmente que la interpolación directa de los elementos de las matrices de covarianza lleva a un rendimiento extremadamente bajo. En este sentido, las ideas aquí presentadas pueden ser extendidas fácilmente a otros modelos de fondo que implementen matrices de covarianza completas.

Las regiones desconocidas que son similares a partes de la escena ya vistas son modeladas adecuadamente mediante extrapolación del fondo (Subsección 7.2.3). Este procedimiento es particularmente relevante en un modelo no panorámico, debido a que la aparición de este tipo de regiones es inherente al movimiento de la cámara, por lo que es ineludible adaptar el modelo a ellas.

Por último hay que hacer notar que nuestra propuesta consigue unos resultados sobresalientes en los vídeos grabados con cámaras PTZ y portátiles, donde los demás métodos estudiados tienen un desempeño pobre. Además es el método más robusto en las secuencias con vibraciones de cámara.

## 7.5. Conclusiones

En este capitulo se ha propuesto un método no panorámico para la detección de los objetos del primer plano en secuencias grabadas con cámaras en movimiento. Emplea un marco de trabajo basado en la aproximación estocástica e incluye un modelo del fondo para cada píxel en el que se usan matrices de covarianza completas, lo cual es más realista que otras aproximaciones anteriores. Se ha desarrollado un procedimiento para transformar las matrices de covarianza en situaciones con movimientos de cámara y otro para extrapolar el modelo del fotograma anterior al actual de forma que se reutilice la mayor información posible en las regiones desconocidas. Ambos procedimientos pueden adaptarse para ser utilizados en otros modelos de fondo.

Se han puesto a prueba un total de nueve secuencias con el fin de comparar nuestra propuesta con otros cuatro métodos pertenecientes al estado del arte. Las secuencias elegidas representan situaciones reales que pueden clasificarse en dos grupos: cámaras con vibraciones y cámaras PTZ o portátiles sin cambios abruptos en el fondo. En siete de los casos nuestra propuesta es la que mejor rendimiento obtiene, mientras que en los otros dos restantes obtienen la segunda y tercera posición respectivamente. En consecuencia nuestra propuesta ha demostrado su potencial como alternativa a otros métodos de segmentación existentes.

# 8 Conclusiones

En este capítulo vamos a exponer las conclusiones finales de la presente tesis doctoral. Antes de esto debemos señalar que el trabajo desarrollado y expuesto a lo largo de la presente tesis ha servido para realizar varias publicaciones. En concreto se han publicado cuatro artículos en revistas indexadas en el *Journal Citation Reports*.

Esta tesis comienza estudiando las características de la entrada para comprender cómo afecta a la detección de movimiento. Esto se lleva a cabo fundamentalmente en los Capítulos 3 y 4.

La importancia del ruido en el vídeo de entrada se constata en el Capítulo 3. Los resultados evidencian que el ruido uniforme es mucho más perjudicial que el ruido gaussiano, incluso con niveles de ruido relativamente bajos. La razón reside en la propia naturaleza de ambos ruidos. El primero destruye toda la información del color en aquellos píxeles a los que afecta, mientras que el segundo preserva parte de dicha información. No obstante, la evolución del rendimiento frente a los diferentes niveles de ruido estudiados varía en función del método considerado, esto hace que existan algunos métodos más robustos que otros. Por ejemplo, el método FuzzyElBaf tiene la respuesta más lineal en ambos tipos de ruido de entre todos los métodos considerados. La explicación es que el esquema de actualización que utiliza protege al modelo de actualizarse cuando la entrada es muy ruidosa. Sin embargo, su rendimiento en ausencia de ruido no es tan bueno como el de otros métodos. Por lo tanto, la elección de uno u otro método depende en gran medida del entorno en el que se vaya a realizar la detección. Por ejemplo, si sabemos que el entorno puede ser ruidoso, la robustez frente al ruido es relevante, sin embargo si sabemos que el ruido va a ser nulo o al menos reducido, existen otras alternativas potencialmente mejores.

Otro resultado interesante es el hecho de que corromper la entrada con una pequeña cantidad de ruido gaussiano puede ser beneficioso para superar algunas limitaciones que padecen ciertos métodos de detección. Esto se pone de manifiesto en dos casos distintos. En primer lugar, cuando ElgammalKDE reduce su cantidad de falsos positivos al añadir ruido gaussiano, está evidenciando un efecto análogo a la regularización de matrices de covarianza. En ausencia de ruido el método se ajusta excesivamente a las características del fondo, por lo que cualquier pequeña imperfección como las que produce la propia compresión de vídeo se traduce en un falso positivo, pero al añadir ruido hacemos que el modelo sea más flexible y se adapte a estos pequeños cambios sin que los detecte como parte del primer plano. En segundo lugar, GrimsonGMM y Sakbot tienden a cometer menos falsos negativos cuando se

añade algo de ruido gaussiano. Esto puede verse como un efecto similar al de la resonancia estocástica, es decir, el color del píxel original de un objeto del primer plano no es suficientemente diferente del que almacena el modelo de fondo, pero al añadir algo de ruido esta diferencia aumenta, con lo que se detecta como primer plano.

El estudio de las características de la entrada continúa en el Capítulo 4 [275]. A la vista de los resultados obtenidos podemos decir que, con el método de detección considerado, los espacios de color que se usan habitualmente como entrada tienen componentes poco relevantes de cara a la detección, mientras que otras son muy importantes. Así pues, asignar el mismo peso a todas las componentes no parece una buena opción. No obstante, los mejores resultados siempre se obtienen utilizando tres componentes, por lo que tampoco sería una buena idea anular alguna componente, ya que todas contribuyen en mayor o menos medida. También es interesante destacar que la componente más relevante es la que codifica la luminancia.

Fruto del conocimiento adquirido en los dos capítulos comentados, se hace evidente la importancia de los rasgos característicos en los métodos de detección. La información que se codifica en cada componente de la imagen de entrada es crucial. De ahí que en el Capítulo 5 [276] se diseñe un método que, más allá de usar como entrada la imagen real codificada con un espacio tridimensional de color, es capaz de aceptar cualquier tipo de entrada en forma de imagen multicanal. Además, dadas las limitaciones de las componentes de color habituales, se proponen otros rasgos específicamente diseñados para la detección de movimiento. En concreto se observa buen rendimiento de los canales de color normalizados y los rasgos basados en el filtro mediana. Los resultados obtenidos al utilizar este método ponen de manifiesto que utilizar más rasgos es beneficioso, si bien más de cinco componentes de entrada no es útil, al menos con el método y los rasgos considerados.

Al lo largo de esta tesis se observa que los métodos existentes tienen una serie de limitaciones entre las que destacan dos porque aparecen con gran frecuencia: los efectos de camuflaje y los cambios de iluminación. El Capítulo 6 [277] aborda la problemática asociada a los cambios de iluminación. Los resultados obtenidos justifican los beneficios de integrar nuestro método de detección de cambios de iluminación en métodos ya existentes que han demostrado su buen funcionamiento en general, pero que no gestionan adecuadamente dichos cambios. Además se prueba que nuestra propuesta modela mejor los cambios de iluminación que otras alternativas diseñadas específicamente para detectar movimiento en este tipo de situaciones.

Dado el buen rendimiento del modelo propuesto en el Capítulo 5 y la relevancia que comienza a tener la detección de movimiento en cámara no estáticas, se considera usarlo como base en el diseño de un nuevo método que sea capaz de realizar esta tarea. Esta idea se materializa en el Capítulo 7 [278]. Los resultados no sólo confirman que el método propuesto es mejor que otras alternativas actuales, sino que el procedimiento de interpolación de matrices de covarianza completas es útil, el cual no se había aplicado hasta ahora a la detección de movimiento. Así mismo, el proce-

dimiento propuesto para extrapolar el modelo a las regiones desconocidas demuestra un buen rendimiento. Si bien nuestro método no está diseñado específicamente para este fin, los resultados preliminares apuntan a que es posible abordar el problema del egomovimiento. En este sentido, como nuestra propuesta independiza el seguimiento y la detección al utilizar un modelo para cada tarea, algo inédito hasta el momento, es posible ajustar el modelo de seguimiento en el caso de que sea necesario.

## 8.1. Trabajos futuros

A continuación vamos a esbozar las posibles líneas de investigación futuras que se derivan de la presente tesis:

- Ampliar el estudio sobre los efectos del ruido en la entrada. En este sentido tenemos tres líneas a seguir, la primera consiste en comprobar si el comportamiento demostrado por los métodos analizados también se da en otros métodos actuales, en particular los efectos de resonancia estocástica y de regularización de matrices de covarianza. En segundo lugar, ampliar el estudio con más secuencias de prueba. Y por último, probar a distorsionar la entrada con ruido de Poisson-Gauss, este tipo de ruido modela algunos tipos distorsiones de la entrada que no hemos llegado a probar [169, 279, 280] de modo que se completaría el estudio.
- Usar el marco de trabajo para ponderar los canales de color de entrada en otros métodos pertenecientes al estado del arte y así ratificar los resultados obtenidos por el método adaptado. Analizar el rendimiento que se consigue al aplicar dicho marco de trabajo a los rasgos de entrada propuestos en el Capítulo 5.
- Profundizar en la búsqueda de rasgos específicamente diseñados para la detección, bien proponiendo unos nuevos, bien recopilando otros propuestos en la bibliografía. Probar su rendimiento en el método propuesto en el Capítulo 5.
- Aplicar el conjunto de rasgos estudiados en el Capítulo 5 en otros modelos de la escena, como por ejemplo una red neuronal competitiva.
- Buscar rasgos que optimicen la tarea de seguimiento de la cámara en aquellas secuencias grabadas con cámaras en movimiento. Si bien los rasgos diseñados para detección no son particularmente buenos en la tarea de seguimiento, es posible que existan otros rasgos que sí lo sean. De este modo se lograría mayor robustez en las secuencias de egomovimiento.
- Diseñar un modelo específicamente orientado al problema del egomovimiento o bien ajustar para dicha tarea el modelo propuesto en el Capítulo 7.

# Apéndice

En las páginas siguientes se muestran los artículos publicados fruto de la investigación llevada a cabo a lo largo de esta tesis. A continuación se exponen varias tablas donde se indican las revistas donde se han publicado cada uno de ellos, así como diferentes medidas de calidad y la relación con los capítulos de la tesis.

| | |
|---|---|
| **Título** | *Selecting the color space for self-organizing map based foreground detection in video* [275] |
| **Autores** | Francisco J. López Rubio, Enrique Domínguez, Esteban J. Palomo, Ezequiel López Rubio, Rafael M Luque-Baena. |
| **Revista** | Neural Processing Letters |
| **Año** | 2015 |
| **Factor de impacto** | 1.448 |
| **Categoría JCR** | COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (Posición: 60/123 (Q2)) |
| **Capítulo** | 4 |

| | |
|---|---|
| **Título** | *Features for stochastic approximation based foreground detection* [276] |
| **Autores** | Francisco Javier López Rubio y Ezequiel López Rubio |
| **Revista** | Computer Vision and Image Understanding |
| **Año** | 2015 |
| **Factor de impacto** | 1.54 |
| **Categoría JCR** | ENGINEERING, ELECTRICAL & ELECTRONIC (Posición: 100/249 (Q2))<br>COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (Posición 56/123 (Q2)) |
| **Capítulo** | 5 |

| | |
|---|---|
| **Título** | *Local color transformation analysis for sudden illumination change detection* [277] |
| **Autores** | Francisco Javier López Rubio y Ezequiel López Rubio |
| **Revista** | Image and Vision Computing |
| **Año** | 2015 |
| **Factor de impacto** | 1.587 |
| **Categoría JCR** | OPTICS (Posición: 41/87 (Q2))<br>COMPUTER SCIENCE, THEORY & METHODS (Posición 22/102 (Q1))<br>COMPUTER SCIENCE, SOFTWARE ENGINEERING (Posición 22/104 (Q1))<br>COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (Posición 52/123 (Q2))<br>ENGINEERING, ELECTRICAL & ELECTRONIC (Posición 96/249 (Q2)) |
| **Capítulo** | 6 |

| | |
|---|---|
| **Título** | *Foreground detection for moving cameras with stochastic approximation* [278] |
| **Autores** | Francisco Javier López Rubio y Ezequiel López Rubio |
| **Revista** | Pattern Recognition Letters |
| **Año** | 2015 |
| **Factor de impacto** | 1.551 |
| **Categoría JCR** | COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE (Posición: 55/123 (Q2)) |
| **Capítulo** | 7 |

Neural Process Lett (2016) 43:345–361
DOI 10.1007/s11063-015-9431-8

# Selecting the Color Space for Self-Organizing Map Based Foreground Detection in Video

**Francisco J. López-Rubio[1] · Enrique Domínguez[1] · Esteban J. Palomo[1] · Ezequiel López-Rubio[1] · Rafael M. Luque-Baena[2]**

Published online: 10 May 2015


**Abstract** Detecting foreground objects on scenes is a fundamental task in computer vision and the used color space is an important election for this task. In many situations, especially on dynamic backgrounds, neither grayscale nor RGB color spaces represent the best solution to detect foreground objects. Other standard color spaces, such as YCbCr or HSV, have been proposed for background modeling in the literature; although the best results have been achieved using diverse color spaces according to the application, scene, algorithm, etc. In this work, a color space and a color component weighting selection process are proposed to detect foreground objects in video sequences using self-organizing maps. Experimental results are also provided using well known benchmark videos.

**Keywords** Probabilistic self-organising maps · Unsupervised learning · Video segmentation · Foreground detection · Color space

✉ Enrique Domínguez
enriqued@lcc.uma.es

Francisco J. López-Rubio
xavierprof@hotmail.com

Esteban J. Palomo
ejpalomo@lcc.uma.es

Ezequiel López-Rubio
ezeqlr@lcc.uma.es

Rafael M. Luque-Baena
rmluque@unex.es

1 Department of Computer Languages and Computer Science, University of Málaga, 29071 Málaga, Spain

2 Department of Computer Systems and Telematics Engineering, University of Extremadura, 06800 Mérida, Spain

## 1 Introduction

The selection of the color space is essential for the good performance of a foreground detection method. In [14], the HSV color space is used for background modeling, which is combined with moving object segmentation based on fuzzy clustering to extract objects of interest from frames. The accurate description of the HSV color space is able to restore the background and then the moving object segmentation is used to distinguish the moving area and noise area. The election of an appropriate color space is further investigated in [6], where a hybrid color space constituted by the three significant color components is determined. This approach is used for color image segmentation in a soccer video with a non-static background. Another approach [5] uses the YCbCr color space to introduce a vehicle shadow segmentation algorithm for moving vehicle detection in a traffic monitoring system. This approach proposes a background subtraction method based on binary discrete wavelet transforms (BDWT). The BDWT is used together with the shadow segmentation algorithm to obtain the motion area in the Y component and then segments the shadow in the YCbCr color space. In [4], a Hybrid Cone-Cylinder Codebook (HC3) model is introduced, which combines an adaptive background model with HSV-color space shadow suppression. The background subtraction problem when there is a non-stationary background is also addressed in [1], where a Gaussian Mixture Model is used for background modeling. Moreover, a different color space named Lab2000HL is used in addition to usual color spaces, which has a linear hue band.

In an earlier work [10] a background model based on probabilistic self-organizing maps was proposed. A fundamental limitation of this approach is that it considers spherical covariance matrices, which means that all the input dimensions are weighted equally. Moreover, only the standard RGB color space was considered. Here we aim to address these issues by selecting the most appropriate color space and color component weighting for a given scene.

The structure of this paper is as follows. First the basic background model is reviewed (Sect. 2). Then our new proposal about color spaces and component weighting selection is presented in Sect. 3. Finally, Sects. 4 and 5 are devoted to experimental results and conclusions, respectively.

## 2 Background Model Review

In this section the basic foreground detection system is reviewed. It is based on a probabilistic background model which employs a self-organizing map to represent the background pixel color. An online learning process rooted on stochastic approximation is used to train the model.

The model receives the incoming video frames and processes their pixels as training samples. It it aimed to build a probabilistic representation of the background of the scene, which is used to determine which pixels belong to the foreground at each time step. As we will see, many color spaces can be used [3], but in all cases the input space dimension is $D = 3$, i.e. tristimulus color values are considered. The probability distribution of the pixel color value $\mathbf{t}$ at pixel position $\mathbf{x}$ is modeled by a mixture with two components, namely $Back$ for the background and $Fore$ for the foreground. The associated probability density function is given by:

$$p_{\mathbf{x}}(\mathbf{t}) = \pi_{Back,\mathbf{x}} p_{\mathbf{x}}(\mathbf{t} \mid Back) + \pi_{Fore,\mathbf{x}} p_{\mathbf{x}}(\mathbf{t} \mid Fore) \tag{1}$$

$$\pi_{Back,\mathbf{x}} + \pi_{Fore,\mathbf{x}} = 1 \tag{2}$$

It must be highlighted that for each pixel location $\mathbf{x}$ a probabilistic mixture is defined in (1). This way the model is able to adapt to the specific characteristics of every pixel of the scene. In general terms it can be assumed that foreground objects can have any color. This calls for a uniform distribution over the color space to model the foreground:

$$p_{\mathbf{x}}(\mathbf{t} \mid Fore) = U(\mathbf{t}) \tag{3}$$

$$U(\mathbf{t}) = \begin{cases} 1/Vol(\mathcal{S}) & \text{iff} \quad \mathbf{t} \in \mathcal{S} \\ 0 & \text{iff} \quad \mathbf{t} \notin \mathcal{S} \end{cases} \tag{4}$$

where $\mathcal{S}$ is the overall color space and $Vol(\mathcal{S})$ is the three dimensional volume of $\mathcal{S}$. This way to model the foreground ensures that all incoming objects are treated the same way, irrespective of their color.

The distribution of the background color values at a certain position $\mathbf{x}$ depends on the features of the scene. For example, waving trees and other dynamic background objects lead to background distributions which are multimodal. A probabilistic self-organizing map can cope with this kind of distribution, since each neuron can specialize on one cluster of the input dataset:

$$p_{\mathbf{x}}(\mathbf{t} \mid Back) = \frac{1}{H} \sum_{i=1}^{H} p_{\mathbf{x}}(\mathbf{t} \mid i) \tag{5}$$

where $H$ is the number of mixture components (neurons) of the self-organizing map, and the prior probabilities or mixing proportions are assumed to be equal. Now it is necessary to define a *topological distance* $d(i, j)$ for each pair of neurons $(i, j)$ of the map. The standard choice has been used here: a rectangular grid to place the units, together with the Euclidean topological distance:

$$d(i, j) = \left\| \mathbf{r}_i - \mathbf{r}_j \right\| \tag{6}$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the positions of units $i$ and $j$ in the rectangular grid, respectively.

The computational load of the probabilistic model should be as small as possible, since there is one self-organizing map for each position $\mathbf{x}$ in the video frame. The simplest option is to consider a spherical Gaussian probability density for each mixture component $i \in \{1, \ldots, H\}$ of the map [2,18,19]:

$$p_{\mathbf{x}}(\mathbf{t} \mid i) = (2\pi)^{-D/2} \left(\sigma_{i,\mathbf{x}}^2\right)^{-D/2} \exp\left( -\frac{1}{2\sigma_{i,\mathbf{x}}^2} \left(\mathbf{t} - \boldsymbol{\mu}_{i,\mathbf{x}}\right) \left(\mathbf{t} - \boldsymbol{\mu}_{i,\mathbf{x}}\right)^T \right) \tag{7}$$

where $\boldsymbol{\mu}_{i,\mathbf{x}}$ and $\sigma_{i,\mathbf{x}}^2$ are the mean vector and the variance of mixture component $i$, respectively:

$$\boldsymbol{\mu}_{i,\mathbf{x}} = E[\mathbf{t} \mid i, \mathbf{x}] \tag{8}$$

$$\sigma_{i,\mathbf{x}}^2 = E\left[ \frac{1}{D} \left(\mathbf{t} - \boldsymbol{\mu}_{i,\mathbf{x}}\right)^T \left(\mathbf{t} - \boldsymbol{\mu}_{i,\mathbf{x}}\right) \mid i, \mathbf{x} \right] \tag{9}$$

In order to decide whether an observed pixel belongs to the background, a Bayesian classification procedure is carried out. The probability that the observed data (pixel color value) $\mathbf{t}$ is background is given by

$$P_{Back,\mathbf{x}}(\mathbf{t}) = \frac{\pi_{Back,\mathbf{x}} p_{\mathbf{x}}(\mathbf{t} \mid Back)}{\pi_{Back,\mathbf{x}} p_{\mathbf{x}}(\mathbf{t} \mid Back) + \pi_{Fore,\mathbf{x}} p_{\mathbf{x}}(\mathbf{t} \mid Fore)} \tag{10}$$

and the corresponding probability of the foreground is the complementary event:

$$P_{Fore,\mathbf{x}}(\mathbf{t}) = 1 - P_{Back,\mathbf{x}}(\mathbf{t}) \tag{11}$$

There are many undesirable effects that introduce noise in $P_{Back,\mathbf{x}}(\mathbf{t})$ and $P_{Fore,\mathbf{x}}(\mathbf{t})$. For example, camouflage effects (foreground objects whose color is similar to that of the background), camera imperfections and video compression artifacts. In order to alleviate this problem, the information from the 8-neighbors of a given pixel $\mathbf{x}$ can be used to reduce the noise. A simple approach would be to weight all neighbors equally (low pass filter), but this would neglect the fact that many neighboring pixels are not related. For example, this can happen on the bank of a river: the pixels outside the water (where no waves occur) are almost independent from those inside the river (where the water current changes the surface) despite of their proximity. A principled way to measure the correlation of pairs of pixels is Pearson's correlation [15] $\rho_{\mathbf{x},\mathbf{y}}$ between the random variables $P_{Fore,\mathbf{x}}$ and $P_{Fore,\mathbf{y}}$ corresponding to each pair of 8-neighboring pixels $\mathbf{x}$ and $\mathbf{y}$.

If two neighboring pixels are usually assigned to the same class, i.e. either both are background or both are foreground, then the correlation between them $\rho_{\mathbf{x},\mathbf{y}}$ is large and positive. On the other hand, if both pixels are independent, then we have $\rho_{\mathbf{x},\mathbf{y}} = 0$. A negative correlation corresponds to a pair of pixels which are usually assigned to opposite classes. This is rather unlikely, but there are some cases where small negative values are obtained due to noise.

The noise in $P_{Fore,\mathbf{x}}(\mathbf{t})$ can be reduced with the help of the correlations $\rho_{\mathbf{x},\mathbf{y}}$, so that the 8-neighbors of $\mathbf{x}$ with the highest positive correlations are given more importance:

$$\widetilde{P}_{Fore,\mathbf{x}}(\mathbf{t}) = \text{trunc}\left(\frac{1}{9}\sum_{\mathbf{y}\in Neigh(\mathbf{x})}\rho_{\mathbf{x},\mathbf{y}}P_{Fore,\mathbf{y}}(\mathbf{t})\right) \tag{12}$$

where $Neigh(\mathbf{x})$ contains the pixel $\mathbf{x}$ and its 8-neighbours, and

$$\rho_{\mathbf{x},\mathbf{x}} = 1 \tag{13}$$

$$\text{trunc}(z) = \begin{cases} z & \text{iff} \quad z \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$\widetilde{P}_{Fore,\mathbf{x}}(\mathbf{t}) \in [0, 1] \tag{15}$$

The role of the trunc function in (12) is to deactivate the parameter learning by setting $\widetilde{P}_{Fore,\mathbf{x}}(\mathbf{t}) = 0$ when a spike of noisy negative correlations occurs. It must be pointed out that the final result of this procedure is postprocessed by filling holes of size one pixel; then the objects with less than 10 pixels in size are removed.

Stochastic approximation is used to train the above model [7]. It has been used before to develop online mode learning procedures for probabilistic self-organizing maps [8,11]. One of its main advantages is that its associated computational complexity $O(HD)$ is quite low, which is of paramount importance for background models based on self-organizing maps [12]. More details about the training algorithm are given in [10].

As in any other application, the probabilistic mixture learning method can fall into a suboptimal solution, i.e. one which does not represent the input distribution faithfully. In order to tackle both problems at a time two thresholds are defined, $\sigma^2_{max}$ and $\sigma^2_{min}$, and it is required that each unit $i$ satisfies the following condition:

$$\sigma^2_{min} \leq \sigma^2_{i,\mathbf{x}} \leq \sigma^2_{max} \tag{16}$$

This is enforced by setting $\sigma^2_{i,\mathbf{x}}$ to $\sigma^2_{max}$ or $\sigma^2_{min}$ every time that a new value of $\sigma^2_{i,\mathbf{x}}$ is produced that does not fulfill (16).

Until now, the nature of the three dimensional inputs $\mathbf{t}$ has not been specified. In the following section, the role of the color space and the weighting of the color components is studied, so that a proper choice of the input information to the background model is carried out.

## 3 Color Spaces

In this section our new proposal about component weighting in color spaces for foreground detection is presented. In most commercial video cameras the pixels are given in the RGB color space with 8-bit precision values. If the raw color information from a camera of this kind was used, then we would have:

$$\mathcal{S}_{RGB} = \{(t_R, t_G, t_B) \,|t_R, t_G, t_B \in [0, 255]\} \tag{17}$$

$$Vol\left(\mathcal{S}_{RGB}\right) = 255^3 \tag{18}$$

However, it is well known that the RGB color space does not reflect the true similarities among colors [1]. Moreover, depending on the scene one color component could be more informative than the others, so it should be given more importance.

Let us consider a general color space $\mathcal{S}$, with colors $\mathbf{t} = (t_1, t_2, t_3) \in \mathcal{S}$. The difference between two colors $\mathbf{t}_A, \mathbf{t}_B \in \mathcal{S}$, which is used in (7) to adapt the probabilistic model, is the Euclidean distance in $S$:

$$\delta\left(\mathbf{t}_A, \mathbf{t}_B\right) = \|\mathbf{t}_A - \mathbf{t}_B\|^2 \tag{19}$$

Under this basic setting, the three color components have the same importance in the distance computation. Now, it is possible that a different weighting of the components yields a better performance:

$$\mathbf{t}' = (\alpha t_1, \beta t_2, \gamma t_3) = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{pmatrix} \mathbf{t} \tag{20}$$

$$\delta\left(\mathbf{t}'_A, \mathbf{t}'_B\right) = \alpha^2 \left(t_{A1} - t_{B1}\right)^2 + \beta^2 \left(t_{A2} - t_{B2}\right)^2 + \gamma^2 \left(t_{A3} - t_{B3}\right)^2 \tag{21}$$

where the scale factors are non negative:

$$\alpha, \beta, \gamma \geq 0 \tag{22}$$

Next, we must take into account that spherical covariance Gaussians are equivariant with respect to homogeneous scalings, i.e. if all the dimensions are scaled by the same factor $\xi$, then the learned model parameters are scaled accordingly [9]. In order to see the consequences of this fact, let us note $\mathcal{S}'$ the transformed color space after a homogeneous scaling transformation with common scaling factor $\xi$. Then the volume of the transformed color space is scaled accordingly:

$$Vol\left(\mathcal{S}'\right) = \xi^3 Vol\left(\mathcal{S}\right) \tag{23}$$

which in turn affects the uniform density distribution employed to model the foreground objects:

$$U\left(\mathbf{t}'\right) = \xi^{-3} U\left(\mathbf{t}\right) \tag{24}$$

At this point we may compare the pixel classification given by a background model $\mathcal{M}$ and its transformed counterpart $\mathcal{M}'$ after a homogeneous scaling with scaling factor $\xi$. The mean vectors and the variances transform as follows:

$$\boldsymbol{\mu}'_{i,\mathbf{x}} = \xi \boldsymbol{\mu}_{i,\mathbf{x}} \tag{25}$$

$$\left(\sigma^2_{i,\mathbf{x}}\right)' = \xi^2 \sigma^2_{i,\mathbf{x}} \tag{26}$$

Consequently the transformed mixture component probabilities are:

$$\begin{aligned} p_{\mathbf{x}}\left(\mathbf{t}' \mid i\right) &= (2\pi)^{-3/2} \left(\left(\sigma^2_{i,\mathbf{x}}\right)'\right)^{-3/2} \exp\left(-\frac{1}{2\left(\sigma^2_{i,\mathbf{x}}\right)'} \left(\mathbf{t}' - \boldsymbol{\mu}'_{i,\mathbf{x}}\right)\left(\mathbf{t}' - \boldsymbol{\mu}'_{i,\mathbf{x}}\right)^T\right) \\ &= (2\pi)^{-3/2} \xi^{-3} \left(\sigma^2_{i,\mathbf{x}}\right)^{-3/2} \exp\left(-\frac{1}{2\sigma^2_{i,\mathbf{x}}} \left(\mathbf{t} - \boldsymbol{\mu}_{i,\mathbf{x}}\right)\left(\mathbf{t} - \boldsymbol{\mu}_{i,\mathbf{x}}\right)^T\right) \\ &= \xi^{-3} p_{\mathbf{x}}\left(\mathbf{t}' \mid i\right) \end{aligned} \tag{27}$$

Then from (24), (27), (1) and (10) it follows that:

$$P_{Back,\mathbf{x}}\left(\mathbf{t}'\right) = P_{Back,\mathbf{x}}\left(\mathbf{t}\right) \tag{28}$$

Equation (28) means that the pixel classification probabilities are not affected by homogeneous scalings which multiply all three color dimensions by the same factor $\xi$. This implies that there are only two degrees of freedom in the choice of the scale factors, since we may normalize the transformed components by using $\xi = \alpha + \beta + \gamma$ to have:

$$\alpha + \beta + \gamma = 1 \tag{29}$$

Please note that (22) and (29) are equivalent to:

$$\alpha, \beta, \gamma \in [0, 1] \tag{30}$$

$$\gamma = 1 - \alpha - \beta \tag{31}$$

Consequently an optimization process can be carried out on $\alpha$ and $\beta$ according to (30) and (31) in order to choose the best scaling factors for the background modeling task at hand.

To weight each component of the color spaces taken into account, we test several different configurations by varying the $\alpha$, $\beta$ and $\gamma$ values. These values are given in Table 1, resulting in 55 different weighting configurations.

## 4 Experimental Results

In this section, we present the tests which have been conducted to justify our conclusions based on the reported results.

Besides the RGB space, five well known color spaces were elected for the study: Lab, Luv, HSV, HSL and YCbCr. Both the CIELAB space, named as Lab, and the Luv were established in 1976 by the Commission Internationale de l'clairage (CIE). HSV (Hue, Saturation, Value) was developed in the 1970s for computer graphics applications and it is one of the most common cylindrical models. Similar color spaces, like HSB (B for Brightness), HSL (L for Lightness) or HSI (I for Intensity) were also developed to balance the advantages and disadvantages of the previous one. These cylindrical models are useful because they are more intuitive than the RGB. The last elected color space YCbCr, also written as $YC_BC_R$ and sometimes abbreviated YCC, is commonly used in video and photography systems and it is composed by the luma component (Y) and the blue and red difference chroma component (Cb and Cr, respectively).

**Table 1** The 55 different configurations of $\alpha$, $\beta$ and $\gamma$ values used to weight each component of a color space

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 0 |
| $\beta$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 |
| $\gamma$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.9 |
| $\alpha$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0 | 0.1 | 0.2 |
| $\beta$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 |
| $\gamma$ | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.8 | 0.7 | 0.6 |
| $\alpha$ | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| $\beta$ | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| $\gamma$ | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 |
| $\alpha$ | 0.6 | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0 | 0.1 | 0.2 | 0.3 |
| $\beta$ | 0.3 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.5 | 0.5 | 0.5 | 0.5 |
| $\gamma$ | 0.1 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.5 | 0.4 | 0.3 | 0.2 |
| $\alpha$ | 0.4 | 0 | 0.1 | 0.2 | 0.3 | 0 | 0.1 | 0.2 | 0 | 0.1 | 0 |
| $\beta$ | 0.5 | 0.6 | 0.6 | 0.6 | 0.6 | 0.7 | 0.7 | 0.7 | 0.8 | 0.8 | 0.9 |
| $\gamma$ | 0.1 | 0.4 | 0.3 | 0.2 | 0.1 | 0.3 | 0.2 | 0.1 | 0.2 | 0.1 | 0.1 |

**Table 2** List of sequences used for the experiments

| Sequence name | Source | Description |
|---|---|---|
| Video2 | VSSN 2006 | 3D objects are artificially inserted to easily detect the foreground |
| Video4 | | |
| Campus | Institute for Infocomm Research | Complicated situations with camouflage, cast shadows and iluminance changes are showed |
| Meeting room | | |
| Subway station | | |
| Water surface | | |
| Lobby | | |
| Fountain | | |
| One shop one wait 1cor | CAVIAR dataset | Busy corridor is presented |
| Level crossing | Sheikh & Shah [16] | Presence of nominal camera motion and dynamic textures |
| Light switch | Laboratory for Image and Media Understanding | Light switch on/off in indoor scene |

A set of 11 sequences that represent real situations (Table 2), such as indoor and outdoor environments, were employed in order to make an exhaustive study as fair as possible. All tested sequences have been used in other studies [12–14,16,17] and are publicly accessible. Note that in most of the cases, a manual segmentation was performed in order to obtain the quantitative results.

A preliminary study was carried on using the set of sequences for each standard color space (RGB, Lab, Luv, HSV, HSL, YCbCr). Previously, the proposed SOM model was adjusted according to each color space, mainly the learning rate $l$, the step size $\varepsilon_0$ and the number of neurons $M$. To this end, simulations were run with varying values of the three parameters

**Table 3** Set of parameters used by the SOM model in each color cpace

| Color space | $l$ | $\varepsilon_0$ | $H$ |
|---|---|---|---|
| RGB | 0.050 | 0.050 | 18 |
| Lab | 0.001 | 0.010 | 18 |
| Luv | 0.005 | 0.001 | 6 |
| HSV | 0.010 | 0.050 | 6 |
| YCbCr | 0.005 | 0.010 | 12 |

$l$, $\varepsilon_0$ and $M$, and the best performing configuration according to F-measure was selected. The ranges of the parameter values were as follows: $l \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$, $\epsilon_0 \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$, $M \in \{3, 6, 12, 18\}$. The higher the learning rate $l$ and the step size $\epsilon_0$, the faster the adaptation of the SOM models to the variations in the pixel colors of the scene. However, a faster adaptation can also lead to worse detection results if these color variations are due to foreground objects. On the other hand, the number of neurons $M$ must be high enough to represent the variability of the background color properly, but if it is too high then some neurons could erroneously associate to the color of some foreground objects. The resulting optimum parameters are shown in Table 3.

Figures 1 and 2 show the segmentation results achieved by the best configuration of the proposed SOM model using several standard color spaces.

It is generally observed that both RGB and YCbCr are the best performing color spaces. The most frequently observed artifact is camouflage, which occurs when the background and foreground colors are so similar that segmentation algorithms take as background pixels some that are actually foreground.

In order to quantify the performance of each of the alternatives, we employed a total of six measures. First we define $A$ as the set of pixels corresponding to foreground and $B$ as the set of pixels classified as foreground by the segmentation method:

$$A = \{\mathbf{t} \mid \chi(\mathbf{t}) = 1\} \tag{32}$$

$$B = \{\mathbf{t} \mid \widetilde{\chi}(\mathbf{t}) = 1\} \tag{33}$$

Two basic measures used in our study are the false negative (FN) and false positive (FP) rates, which are defined as follow (lower is better):

$$FN = \frac{\text{card}\left(A \cap \overline{B}\right)}{\text{card}\left(A \cup B\right)} \tag{34}$$

$$FP = \frac{\text{card}\left(\overline{A} \cap B\right)}{\text{card}\left(A \cup B\right)} \tag{35}$$

where 'card' stands for the number of elements of a set. Another pair of measures we have used in this study are precision (PR) and recall (RC) (higher is better):

$$PR = \frac{\text{card}\left(A \cap B\right)}{\text{card}\left(B\right)} \tag{36}$$

$$RC = \frac{\text{card}\left(A \cap B\right)}{\text{card}\left(A\right)} \tag{37}$$

$$PR,\ RC \in [0, 1] \tag{38}$$

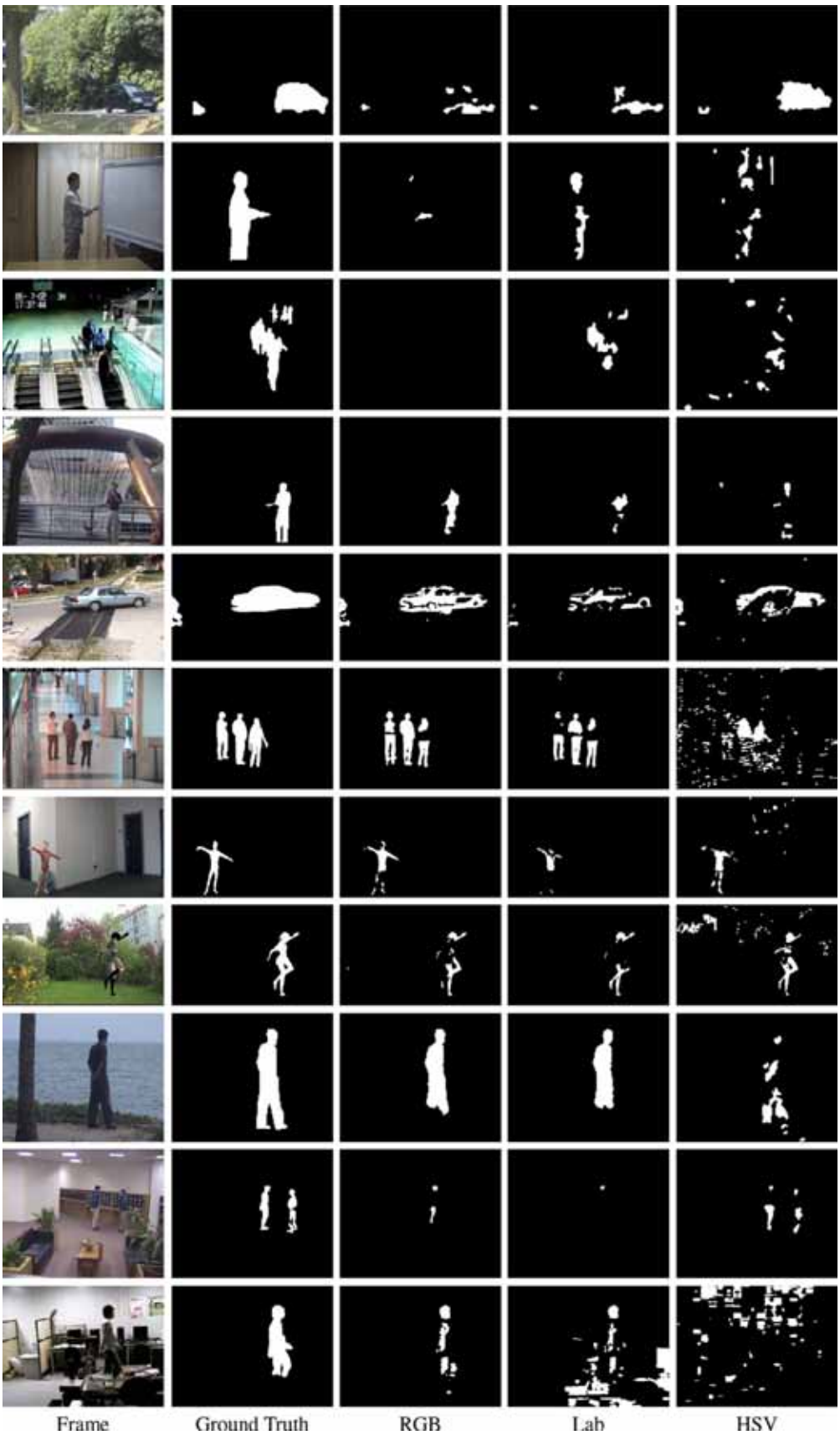



**Fig. 1** Segmentation results by the best performing configuration for each color space. Rows from top to bottom: Campus (frame 2348), Meeting Room (23835), Subway Station (4787), Fountain (1489), LevelCrossing (440), Corridor (370), Video2 (550), Video4 (690), WaterSurface (1559), Lobby (2440) and LightSwitch (1880). The first two columns show original frame and Ground Truth, the last three columns show the tested color spaces: RGB, Lab and HSV

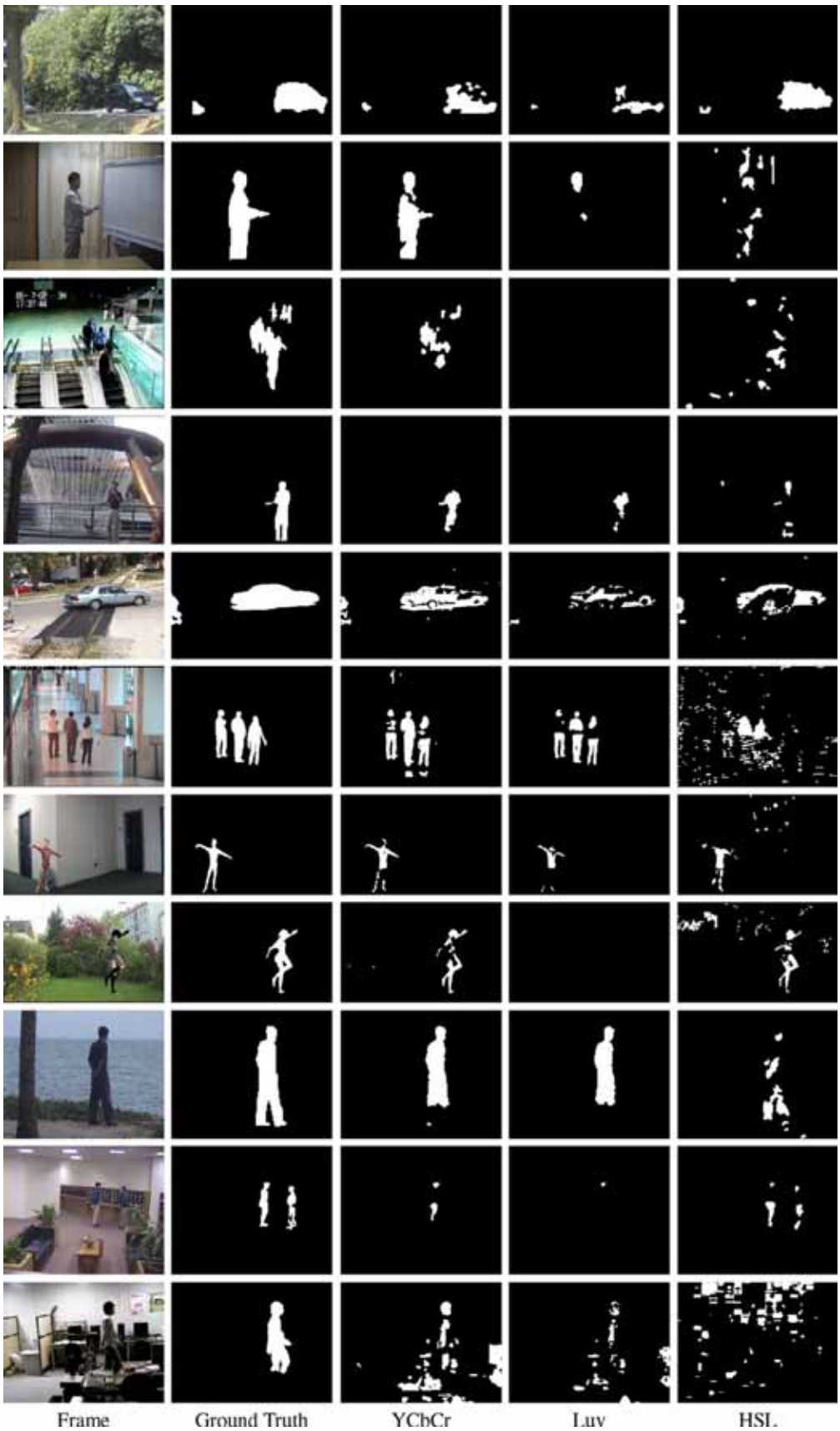
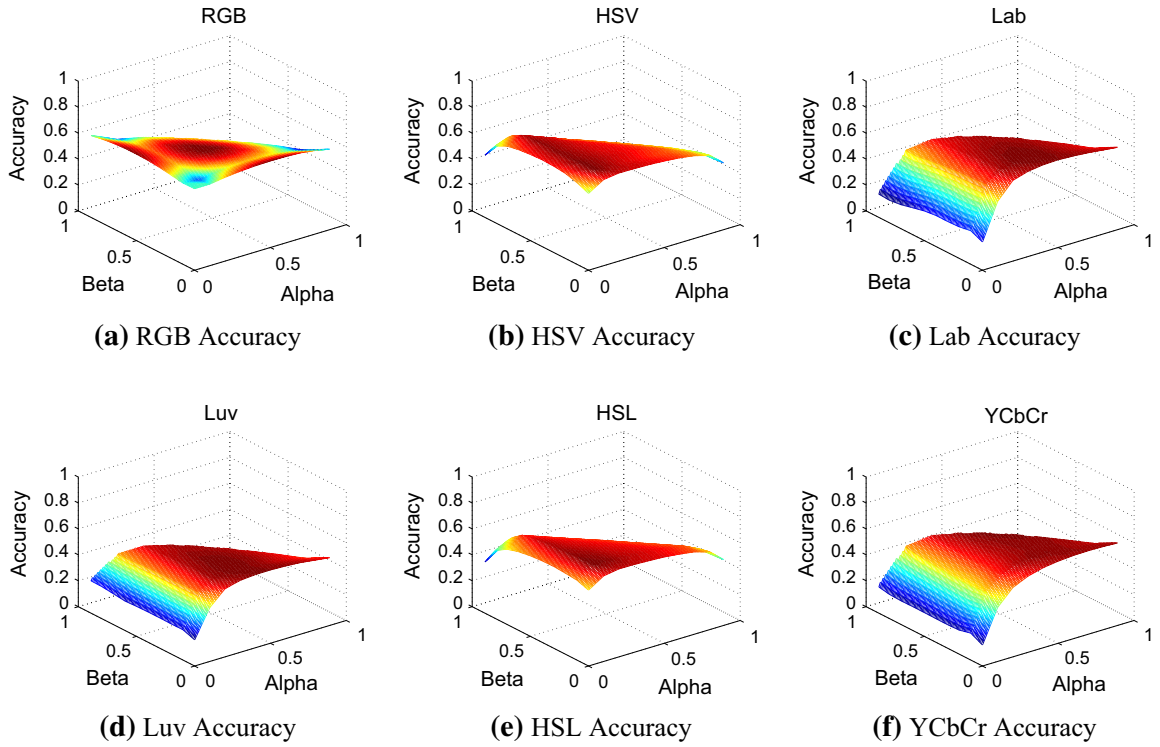


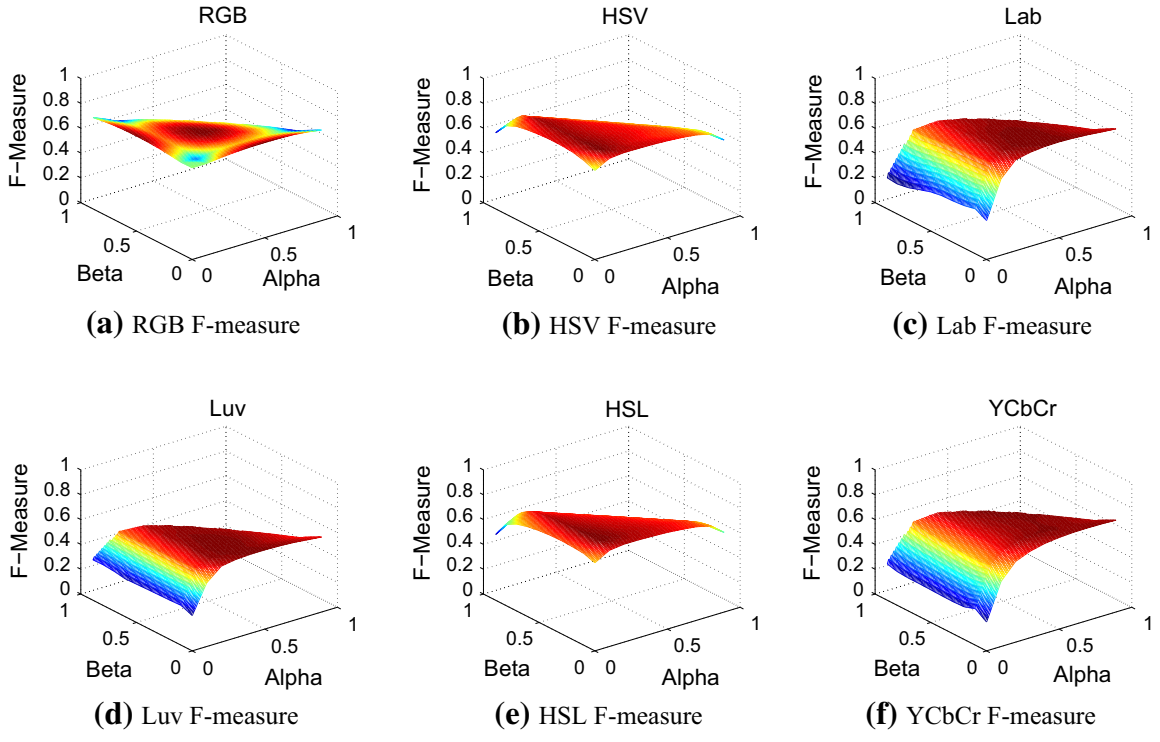

**Fig. 2** Segmentation results by the best performing configuration for each color space. Rows from top to bottom: Campus (frame 2348), Meeting Room (23835), Subway Station (4787), Fountain (1489), LevelCrossing (440), Corridor (370), Video2 (550), Video4 (690), WaterSurface (1559), Lobby (2440) and LightSwitch (1880). The first two columns show original frame and Ground Truth, the last three columns show the tested color spaces: YCbCr, Luv and HSL




**(a)** RGB Accuracy **(b)** HSV Accuracy **(c)** Lab Accuracy

**(d)** Luv Accuracy **(e)** HSL Accuracy **(f)** YCbCr Accuracy

**Fig. 3** Accuracy achieved for each color space using different weighting configurations. The X and Y axis represent the $\alpha$ and $\beta$ weights, respectively. Note that $\gamma = 1 - \alpha - \beta$




**(a)** RGB F-measure **(b)** HSV F-measure **(c)** Lab F-measure

**(d)** Luv F-measure **(e)** HSL F-measure **(f)** YCbCr F-measure

**Fig. 4** F-measure achieved for each color space using different weighting configurations. The X and Y axis represent the $\alpha$ and $\beta$ weights, respectively. Note that $\gamma = 1 - \alpha - \beta$

In many cases the optima of $PR$ and $RC$ are not attained at the same system configuration. Consequently it is hard to choose the best performing configuration according to both $PR$ and $RC$. Under these conditions the F-measure can be employed, which combines $PR$ and $RC$ in a single measure (higher is better):

**Table 4** Best accuracy, F-measure and weighting configuration $(\alpha, \beta, \gamma)$ for each color space and video sequence

| Video sequence | | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|---|
| Campus | Accuracy | *0.77 (0.09)* | *0.77 (0.09)* | 0.74 (0.07) | 0.75 (0.07) | 0.64 (0.13) | 0.72 (0.07) |
| | F-measure | *0.87 (0.06)* | *0.86 (0.06)* | 0.85 (0.05) | 0.85 (0.05) | 0.78 (0.10) | 0.84 (0.05) |
| | Weighting | *(0.2,0.2,0.6)* | *(0.2,0.1,0.7)* | (0.2,0.3,0.5) | (0.1,0.4,0.5) | (0.4,0.0,0.6) | (0.2,0.7,0.1) |
| Meeting Room | Accuracy | 0.75 (0.07) | 0.75 (0.08) | 0.80 (0.06) | *0.81 (0.07)* | *0.81 (0.07)* | 0.80 (0.08) |
| | F-measure | 0.86 (0.04) | 0.86 (0.05) | 0.89 (0.04) | *0.89 (0.05)* | *0.89 (0.05)* | 0.88 (0.05) |
| | Weighting | (0.1,0.1,0.8) | (0.2,0.1,0.7) | (0.4,0.1,0.5) | *(0.4,0.1,0.5)* | *(0.6,0.0,0.4)* | (0.5,0.1,0.4) |
| Subway station | Accuracy | 0.61 (0.10) | 0.55 (0.12) | 0.56 (0.14) | 0.03 (0.06) | *0.62 (0.11)* | 0.53 (0.15) |
| | F-measure | 0.75 (0.09) | 0.70 (0.11) | 0.71 (0.13) | 0.04 (0.10) | *0.76 (0.10)* | 0.68 (0.14) |
| | Weighting | (0.1,0.1,0.8) | (0.0,0.3,0.7) | (0.4,0.5,0.1) | (0.5,0.4,0.1) | *(0.2,0.5,0.3)* | (0.7,0.1,0.2) |
| Fountain | Accuracy | *0.76 (0.06)* | *0.76 (0.06)* | 0.66 (0.09) | 0.67 (0.07) | 0.65 (0.05) | 0.66 (0.09) |
| | F-measure | *0.86 (0.04)* | *0.86 (0.04)* | 0.79 (0.07) | 0.80 (0.05) | 0.79 (0.04) | 0.79 (0.07) |
| | Weighting | *(0.5,0.1,0.4)* | *(0.4,0.2,0.4)* | (0.5,0.0,0.5) | (0.3,0.6,0.1) | (0.5,0.0,0.5) | (0.6,0.0,0.4) |
| Level crossing | Accuracy | 0.89 (0.04) | 0.89 (0.04) | *0.91 (0.03)* | 0.89 (0.03) | *0.91 (0.03)* | 0.90 (0.03) |
| | F-measure | 0.94 (0.02) | 0.94 (0.02) | *0.95 (0.02)* | 0.94 (0.02) | *0.95 (0.02)* | 0.95 (0.02) |
| | Weighting | (0.1,0.4,0.5) | (0.1,0.3,0.6) | *(0.4,0.1,0.5)* | (0.4,0.1,0.5) | *(0.4,0.3,0.3)* | (0.4,0.5,0.1) |
| Corridor | Accuracy | 0.77 (0.03) | 0.80 (0.05) | 0.82 (0.04) | *0.84 (0.04)* | 0.63 (0.02) | *0.84 (0.04)* |
| | F-measure | 0.87 (0.02) | 0.89 (0.03) | 0.90 (0.03) | *0.92 (0.03)* | 0.77 (0.02) | *0.91 (0.03)* |
| | Weighting | (0.0,0.6,0.4) | (0.1,0.8,0.1) | (0.1,0.4,0.5) | *(0.1,0.6,0.3)* | (0.5,0.0,0.5) | *(0.1,0.1,0.8)* |
| Video2 | Accuracy | 0.89 (0.05) | 0.90 (0.03) | *0.94 (0.02)* | *0.94 (0.02)* | 0.93 (0.02) | 0.94 (0.03) |
| | F-measure | 0.94 (0.03) | 0.94 (0.02) | *0.97 (0.01)* | *0.97 (0.01)* | 0.96 (0.01) | 0.97 (0.01) |
| | Weighting | (0.0,0.2,0.8) | (0.0,0.2,0.8) | *(0.8,0.1,0.1)* | *(0.8,0.1,0.1)* | (0.2,0.7,0.1) | (0.8,0.1,0.1) |
| Video4 | Accuracy | *0.79 (0.05)* | 0.79 (0.10) | 0.76 (0.11) | 0.00 (0.01) | 0.75 (0.07) | 0.73 (0.11) |
| | F-measure | *0.88 (0.03)* | 0.88 (0.07) | 0.86 (0.08) | 0.01 (0.01) | 0.85 (0.05) | 0.84 (0.07) |
| | Weighting | *(0.2,0.0,0.8)* | (0.2,0.1,0.7) | (0.4,0.4,0.2) | (0.1,0.5,0.4) | (0.0,0.0,1.0) | (0.6,0.3,0.1) |
| Water surface | Accuracy | 0.89 (0.03) | 0.91 (0.03) | *0.91 (0.02)* | *0.91 (0.02)* | 0.90 (0.02) | 0.90 (0.02) |
| | F-measure | 0.94 (0.02) | 0.95 (0.02) | *0.95 (0.01)* | *0.95 (0.01)* | 0.95 (0.01) | 0.95 (0.01) |
| | Weighting | (0.2,0.5,0.3) | (0.1,0.6,0.3) | *(0.4,0.1,0.5)* | *(0.4,0.1,0.5)* | (0.1,0.7,0.2) | (0.8,0.1,0.1) |
| Lobby | Accuracy | 0.50 (0.15) | 0.54 (0.16) | 0.61 (0.26) | 0.62 (0.27) | *0.64 (0.27)* | 0.56 (0.24) |
| | F-measure | 0.65 (0.13) | 0.69 (0.14) | 0.71 (0.29) | 0.72 (0.30) | *0.73 (0.30)* | 0.67 (0.28) |
| | Weighting | (0.4,0.1,0.5) | (0.1,0.5,0.4) | (0.5,0.1,0.4) | (0.5,0.1,0.4) | *(0.4,0.1,0.5)* | (0.8,0.1,0.1) |
| Light switch | Accuracy | 0.51 (0.17) | *0.55 (0.16)* | 0.27 (0.17) | 0.19 (0.12) | 0.12 (0.21) | 0.23 (0.13) |
| | F-measure | 0.66 (0.18) | *0.70 (0.17)* | 0.39 (0.22) | 0.31 (0.16) | 0.18 (0.23) | 0.35 (0.17) |
| | Weighting | (0.0,0.3,0.7) | *(0.0,0.2,0.8)* | (0.1,0.1,0.8) | (0.1,0.2,0.7) | (0.5,0.0,0.5) | (0.1,0.8,0.1) |

Best results for each video sequence are in italic

$$FM = \frac{2 \times PR \times RC}{PR + RC} \tag{39}$$

Another interesting measure is accuracy (AC), which is defined as follows (higher is better):

$$AC = \frac{\text{card}\,(A \cap B)}{\text{card}\,(A \cup B)} \tag{40}$$

**Table 5** Accuracy and F-measure of the uniform weighting configuration ($\alpha, \beta, \gamma = 1/3$) for each color space and video sequence

| Video sequence | | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|---|
| Campus | Accuracy | *0.76 (0.10)* | 0.74 (0.14) | 0.71 (0.11) | 0.73 (0.09) | 0.62 (0.16) | 0.60 (0.15) |
| | F-measure | *0.86 (0.07)* | 0.84 (0.11) | 0.82 (0.08) | 0.84 (0.06) | 0.76 (0.13) | 0.74 (0.12) |
| Meeting room | Accuracy | 0.61 (0.16) | 0.68 (0.14) | 0.77 (0.08) | *0.79 (0.07)* | *0.79 (0.07)* | 0.76 (0.08) |
| | F-measure | 0.75 (0.12) | 0.80 (0.10) | 0.87 (0.05) | *0.88 (0.05)* | *0.88 (0.05)* | 0.86 (0.05) |
| Subway station | Accuracy | 0.46 (0.10) | 0.34 (0.10) | 0.52 (0.15) | 0.02 (0.06) | *0.62 (0.11)* | 0.43 (0.16) |
| | F-measure | 0.62 (0.10) | 0.50 (0.12) | 0.68 (0.14) | 0.04 (0.09) | *0.76 (0.09)* | 0.59 (0.16) |
| Fountain | Accuracy | 0.73 (0.07) | *0.75 (0.06)* | 0.64 (0.10) | 0.65 (0.09) | 0.61 (0.07) | 0.61 (0.12) |
| | F-measure | 0.84 (0.05) | *0.86 (0.04)* | 0.78 (0.08) | 0.79 (0.07) | 0.76 (0.05) | 0.75 (0.10) |
| Level crossing | Accuracy | 0.88 (0.05) | 0.88 (0.05) | 0.89 (0.03) | 0.89 (0.03) | *0.91 (0.03)* | 0.88 (0.03) |
| | F-measure | 0.93 (0.03) | 0.94 (0.03) | 0.94 (0.02) | 0.94 (0.02) | *0.95 (0.02)* | 0.94 (0.02) |
| Corridor | Accuracy | *0.74 (0.05)* | 0.73 (0.05) | 0.69 (0.04) | 0.70 (0.04) | 0.60 (0.02) | 0.70 (0.04) |
| | F-measure | *0.85 (0.03)* | 0.85 (0.03) | 0.81 (0.03) | 0.82 (0.03) | 0.75 (0.02) | 0.82 (0.03) |
| Video2 | Accuracy | 0.80 (0.07) | 0.79 (0.08) | 0.92 (0.03) | 0.91 (0.02) | *0.92 (0.02)* | 0.91 (0.03) |
| | F-measure | 0.89 (0.05) | 0.88 (0.05) | 0.96 (0.01) | 0.96 (0.01) | *0.96 (0.01)* | 0.95 (0.02) |
| Video4 | Accuracy | 0.71 (0.08) | 0.74 (0.11) | *0.76 (0.11)* | 0.00 (0.01) | 0.66 (0.12) | 0.70 (0.09) |
| | F-measure | 0.83 (0.05) | 0.84 (0.08) | *0.86 (0.08)* | 0.01 (0.01) | 0.79 (0.09) | 0.82 (0.07) |
| Water surface | Accuracy | 0.88 (0.03) | *0.90 (0.02)* | 0.89 (0.03) | 0.90 (0.03) | 0.89 (0.03) | 0.85 (0.04) |
| | F-measure | 0.93 (0.01) | *0.95 (0.01)* | 0.94 (0.02) | 0.94 (0.02) | 0.94 (0.02) | 0.92 (0.02) |
| Lobby | Accuracy | 0.47 (0.16) | 0.54 (0.16) | 0.50 (0.23) | 0.55 (0.25) | *0.62 (0.27)* | 0.41 (0.22) |
| | F-measure | 0.62 (0.15) | 0.69 (0.14) | 0.62 (0.26) | 0.67 (0.28) | *0.72 (0.30)* | 0.54 (0.25) |
| Light switch | Accuracy | *0.32 (0.15)* | *0.32 (0.15)* | 0.12 (0.20) | 0.10 (0.15) | 0.12 (0.21) | 0.12 (0.19) |
| | F-measure | *0.46 (0.18)* | *0.47 (0.18)* | 0.18 (0.22) | 0.15 (0.19) | 0.18 (0.23) | 0.18 (0.21) |

Best results for each video sequence are in italic

The accuracy and F-measure results for each color space and weighting configuration are shown in Figs. 3 and 4, respectively. In these figures, the X axis shows the $\alpha$ weighting, the Y axis the $\beta$ weighting and the vertical axis the accuracy or F-measure results. Each weighting configuration point is computed as the mean of the accuracies or F-measures corresponding to the 11 video sequences. Note that for each color space, the accuracy and F-measure results are similar since these two measures are correlated with each other. For this reason, the accuracy is used to decide the best color component weighting for each color space. Also, by comparing the resulting shapes three groups of color spaces can be identified, which are more similar among them: Lab, Luv and YCbCr; HSV and HSL; and RGB. The highest accuracies and F-measures are achieved by the following color spaces in this order: HSL, HSV, Lab, RGB, YCbCr and Luv. This confirms that RGB is not the most appropriate color space for foreground detection in video sequences.

In order to compute the best weighting configuration for each color space, all the weighting configurations were tested on the set of 11 video sequences for each color space. The best accuracy, F-measure and weighting configuration for each color space and video sequence is given in Table 4, where the best results for each video sequence are in bold. Here, the maximum accuracy value was used to select the weighting configuration, since accuracy and F-measure are correlated. In general terms, the best color space depends largely on the

**Table 6** Mean of the accuracies achieved with the 11 video squences for each color space and weighting configuration

| Weigting | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|
| (0.0,0.0,1.0) | 0.59 | 0.59 | 0.22 | 0.20 | 0.63 | 0.16 |
| (0.1,0.0,0.9) | 0.66 | 0.66 | 0.50 | 0.42 | 0.63 | 0.42 |
| (0.2,0.0,0.8) | 0.68 | 0.68 | 0.62 | 0.53 | 0.65 | 0.54 |
| (0.3,0.0,0.7) | 0.68 | 0.68 | 0.66 | 0.56 | 0.66 | 0.59 |
| (0.4,0.0,0.6) | 0.67 | 0.68 | 0.67 | 0.57 | 0.67 | 0.62 |
| (0.5,0.0,0.5) | 0.67 | 0.67 | 0.68 | 0.57 | 0.67 | 0.63 |
| (0.6,0.0,0.4) | 0.65 | 0.66 | 0.67 | 0.56 | 0.67 | 0.64 |
| (0.7,0.0,0.3) | 0.64 | 0.64 | 0.66 | 0.55 | 0.66 | 0.64 |
| (0.8,0.0,0.2) | 0.60 | 0.60 | 0.65 | 0.54 | 0.64 | 0.64 |
| (0.9,0.0,0.1) | 0.50 | 0.51 | 0.63 | 0.52 | 0.62 | 0.64 |
| (0.0,0.1,0.9) | 0.62 | 0.63 | 0.26 | 0.23 | 0.63 | 0.21 |
| *(0.1,0.1,0.8)* | *0.70* | *0.70* | 0.58 | 0.49 | 0.61 | 0.49 |
| (0.2,0.1,0.7) | 0.70 | 0.69 | 0.66 | 0.56 | 0.63 | 0.59 |
| (0.3,0.1,0.6) | 0.68 | 0.68 | 0.68 | 0.57 | 0.64 | 0.63 |
| *(0.4,0.1,0.5)* | 0.68 | 0.67 | *0.68* | 0.57 | 0.65 | 0.65 |
| (0.5,0.1,0.4) | 0.67 | 0.66 | 0.68 | 0.56 | 0.65 | 0.65 |
| (0.6,0.1,0.3) | 0.65 | 0.65 | 0.66 | 0.55 | 0.64 | 0.65 |
| (0.7,0.1,0.2) | 0.63 | 0.63 | 0.64 | 0.53 | 0.63 | 0.64 |
| (0.8,0.1,0.1) | 0.58 | 0.58 | 0.62 | 0.51 | 0.60 | 0.62 |
| (0.0,0.2,0.8) | 0.63 | 0.66 | 0.24 | 0.23 | 0.64 | 0.19 |
| (0.1,0.2,0.7) | 0.70 | 0.70 | 0.58 | 0.48 | 0.64 | 0.48 |
| (0.2,0.2,0.6) | 0.69 | 0.69 | 0.66 | 0.56 | 0.65 | 0.58 |
| *(0.3,0.2,0.5)* | 0.68 | 0.68 | 0.68 | *0.57* | 0.66 | 0.62 |
| (0.4,0.2,0.4) | 0.67 | 0.67 | 0.68 | 0.57 | 0.66 | 0.64 |
| (0.5,0.2,0.3) | 0.66 | 0.66 | 0.67 | 0.56 | 0.66 | 0.65 |
| (0.6,0.2,0.2) | 0.64 | 0.65 | 0.66 | 0.55 | 0.65 | 0.65 |
| (0.7,0.2,0.1) | 0.58 | 0.61 | 0.64 | 0.53 | 0.62 | 0.64 |
| (0.0,0.3,0.7) | 0.63 | 0.67 | 0.24 | 0.23 | 0.66 | 0.18 |
| (0.1,0.3,0.6) | 0.69 | 0.69 | 0.57 | 0.48 | 0.65 | 0.48 |
| (0.2,0.3,0.5) | 0.68 | 0.69 | 0.65 | 0.55 | 0.66 | 0.58 |
| *(0.3,0.3,0.4)* | 0.67 | 0.68 | 0.67 | 0.57 | *0.67* | 0.62 |
| (0.4,0.3,0.3) | 0.66 | 0.67 | 0.68 | 0.57 | 0.67 | 0.64 |
| (0.5,0.3,0.2) | 0.64 | 0.66 | 0.67 | 0.56 | 0.66 | 0.65 |
| (0.6,0.3,0.1) | 0.59 | 0.61 | 0.65 | 0.54 | 0.64 | 0.65 |
| (0.0,0.4,0.6) | 0.63 | 0.67 | 0.24 | 0.23 | 0.66 | 0.18 |
| (0.1,0.4,0.5) | 0.69 | 0.69 | 0.56 | 0.48 | 0.65 | 0.48 |
| (0.2,0.4,0.4) | 0.68 | 0.68 | 0.64 | 0.55 | 0.67 | 0.59 |
| (0.3,0.4,0.3) | 0.66 | 0.67 | 0.66 | 0.56 | 0.67 | 0.63 |
| (0.4,0.4,0.2) | 0.65 | 0.66 | 0.67 | 0.56 | 0.66 | 0.65 |
| (0.5,0.4,0.1) | 0.59 | 0.62 | 0.66 | 0.55 | 0.64 | 0.66 |
| (0.0,0.5,0.5) | 0.62 | 0.67 | 0.23 | 0.23 | 0.67 | 0.18 |
| (0.1,0.5,0.4) | 0.68 | 0.69 | 0.55 | 0.48 | 0.65 | 0.49 |
| (0.2,0.5,0.3) | 0.67 | 0.68 | 0.63 | 0.55 | 0.66 | 0.59 |

**Table 6** continued

| Weigting | HSL | HSV | Lab | Luv | RGB | YCbCr |
|---|---|---|---|---|---|---|
| (0.3,0.5,0.2) | 0.65 | 0.67 | 0.65 | 0.56 | 0.66 | 0.64 |
| *(0.4,0.5,0.1)* | 0.60 | 0.63 | 0.67 | 0.56 | 0.64 | *0.66* |
| (0.0,0.6,0.4) | 0.61 | 0.66 | 0.21 | 0.23 | 0.67 | 0.18 |
| (0.1,0.6,0.3) | 0.67 | 0.69 | 0.54 | 0.47 | 0.65 | 0.50 |
| (0.2,0.6,0.2) | 0.65 | 0.68 | 0.62 | 0.54 | 0.66 | 0.60 |
| (0.3,0.6,0.1) | 0.59 | 0.63 | 0.65 | 0.56 | 0.64 | 0.65 |
| (0.0,0.7,0.3) | 0.58 | 0.64 | 0.19 | 0.24 | 0.66 | 0.18 |
| (0.1,0.7,0.2) | 0.65 | 0.68 | 0.52 | 0.46 | 0.64 | 0.51 |
| (0.2,0.7,0.1) | 0.59 | 0.63 | 0.62 | 0.54 | 0.64 | 0.62 |
| (0.0,0.8,0.2) | 0.52 | 0.59 | 0.17 | 0.25 | 0.65 | 0.19 |
| (0.1,0.8,0.1) | 0.58 | 0.64 | 0.52 | 0.46 | 0.61 | 0.52 |
| (0.0,0.9,0.1) | 0.38 | 0.47 | 0.18 | 0.25 | 0.63 | 0.20 |

Best results for each color space are in italic

video sequence analyzed. Thus, the cylindrical models (HSL and HSV) are the best option for sequences with high variability on the background, such as Campus and Video4, where the movement of the vegetation should be handled as part of the background, or the Fountain scene, where the water in the fountain should not be considered as motion. Additionally, the sequence LightSwitch with sudden illumination changes, improves significantly with HSL and HSV spaces, probably because the cylindrical form of these models helps to cope with the structure or range of the color variation. Both Lab and Luv color spaces present also good segmentation results, especially in sequences with low variability in the background (Video2, LevelCrossing or WaterSurface).

We have also compared these results with those achieved by the uniform weighting configuration ($\alpha, \beta, \gamma = 1/3$), where all the components of the color space are equally used. These results are shown in Table 5. Note that both the accuracy and F-measure of all color spaces are improved using a non-uniform weighting configuration. Thus, the importance of weighting color component is demonstrated for foreground detection.

Next, we aim to compute the best weighting configuration for each color space independently from the video sequence. First for each weighting configuration and color space, we compute the mean of the accuracies achieved with the 11 video sequences. Then the weighting configuration with the maximum value of the 55 resulting means is chosen as the best weighting configuration for that color space as shown in Table 6.

This table shows a good snapshot of the most valuable component of each color space for foreground detection. For instance, the luminance (first component) and the chromatic color from blue to yellow (third component) are the most important in both Lab and Luv color spaces. The significance of the third component is due to background color of the used sequences. That is, the most prominent background color of the used sequences is ranged between blue to yellow. Luminance is also the most valuable component of the cylindrical models (HSL ans HSV), where the significance is greater. However, both the hue and the saturation are certainly necessary, since the accuracy is decreased when these components are null (e.g. see the first configuration). Finally, the components of the RGB color space are equally valuables, since the luminance is uniformly distributed between the components.

## 5 Conclusions

A common framework to choose the optimal color space and color component scaling for foreground object detection has been presented. This framework has been applied to a probabilistic background model which is based on a self-organizing neural network. The performance of the proposal has been tested with several well known benchmark videos, and it has been found that the luminance color components are the most relevant ones for this task. This work opens the way to tune other background models with the proposed framework to achieve better object detection performance.

**Acknowledgments** This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2014-53465-R, project name Video surveillance by active search of anomalous events. It is also partially supported by the Autonomous Government of Extremadura (Spain) under the project IB13113 and by the Autonomous Government of Andalusia (Spain) under projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga.

## References

1. Balcilar M, Karabiber F, Sonmez A (2013) Performance analysis of Lab2000HL color space for background subtraction. In: 2013 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) (2013)
2. Bishop CM, Svenson M (1998) The generative topographic mapping. Neural Comput 10(1):215–234
3. Brainard DH (2003) The science of color. In: Shevell SK (ed) Color appearance and color difference specification. Elsevier, Oxford, pp 191–216
4. Doshi A, Trivedi M (2007) Satellite imagery based adaptive background models and shadow suppression. Signal Image Video Process 1(2):119–132
5. Gao T, Liu ZG, Yue SH, Mei JQ, Zhang J (2009) Traffic video-based moving vehicle detection and tracking in the complex environment. Cybern Syst 40(7):569–588
6. Jlassi M, Douik A, Messaoud H (2010) Color images segmentation algorithms during a sports meeting: application to soccer video images. J Circuits Syst Comput 19(6):1307–1332
7. Kushner HJ, Yin GG (2003) Stochastic approximation and recursive algorithms and applications. Springer, New York
8. López-Rubio E (2009) Multivariate student-*t* self-organizing maps. Neural Netw 22(10):1432–1447
9. López-Rubio E (2009) Robust location and spread measures for nonparametric probability density function estimation. Int J Neural Syst 19(5):345–357
10. López-Rubio E, Luque-Baena RM, Dominguez E (2011) Foreground detection in video sequences with probabilistic self-organizing maps. Int J Neural Syst 21(3):225–246
11. López-Rubio E, Ortiz-de-Lazcano-Lobato JM, López-Rodríguez D (2009) Probabilistic PCA self-organizing maps. IEEE Trans Neural Netw 20(9):1474–1489
12. Maddalena L, Petrosino A (2008) A self-organizing approach to background subtraction for visual surveillance applications. IEEE Trans Image Process 17(7):1168–1177. doi:10.1109/TIP.2008.924285
13. Ming Y, Jiang J (2008) Background modeling and moving-objects detection based on Cauchy distribution for video sequence. Acta Optica Sinica 28(3):587–592
14. Ning J, Yang Y, Zhu F (2013) Background modeling and fuzzy clustering for motion detection from video. J Multimed 8(5):626–631
15. Rodgers JL, Nicewander WA (1988) Thirteen ways to look at the correlation coefficient. Am Stat 42(1):59–66
16. Sheikh Y, Shah M (2005) Bayesian modeling of dynamic scenes for object detection. Pattern Anal Mach Intell IEEE Trans 27(11):1778–1792. doi:10.1109/TPAMI.2005.213
17. Shimada A, Taniguchi R (2010) Hybrid background modeling for long-term and short-term illumination changes. IEEJ Trans Electron Inf Syst 130(9):1524–1529

18. Van Hulle M (2005) Maximum likelihood topographic map formation. Neural Comput 17(3):503–513
19. Van Hulle MM (2002) Kernel-based topographic map formation by local density modeling. Neural Comput 14(7):1561–1573

Computer Vision and Image Understanding 133 (2015) 30–50

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

# Features for stochastic approximation based foreground detection ☆

Francisco Javier López-Rubio, Ezequiel López-Rubio *

*Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur, 35, 29071 Málaga, Spain*

ARTICLE INFO

*Article history:*
Received 13 May 2014
Accepted 24 December 2014
Available online 31 December 2014

*Keywords:*
Background modeling
Foreground detection
Probabilistic mixture model
Background features
Pareto front

ABSTRACT

Foreground detection algorithms have sometimes relied on rather ad hoc procedures, even when probabilistic mixture models are defined. Moreover, the fact that the input features have different variances and that they are not independent from each other is often neglected, which hampers performance. Here we aim to obtain a background model which is not tied to any particular choice of features, and that accounts for the variability and the dependences among features. It is based on the stochastic approximation framework. A possible set of features is presented, and their suitability for this problem is assessed. Finally, the proposed procedure is compared with several state-of-the-art alternatives, with satisfactory results.



## 1. Introduction

The proliferation of video vigilance systems has given rise to an ever increasing need of preprocessing, analyzing, indexing and searching over huge quantities of video data. These tasks cannot be efficiently carried out by humans, since the volume of data to be processed is too large for their capabilities. Hence, activity analysis intelligent systems are an emergent research field with multiple applications, both in the private and public sectors [36].

Under a modular view of computer vision systems, the separation of moving objects from the background is one of the earliest stages. This is an essential part of any surveillance system, since its performance deeply influences the higher level stages which carry out the object detection. Consequently, much effort has been devoted to this complex problem, which includes challenges such as dynamic backgrounds, shadows, objects which integrate into the background, sudden illumination changes, color similarity (camouflage) and many others [4].

Most approaches are based on building a model of the background which is based on some statistics obtained from the previous video frames. We might classify most of them into four classes: median based, kernel density estimation based, subspace based, and probabilistic mixture based [15]. The first class computes the median of the pixel values over the last frames in order to obtain a robust estimation of the background image. These approaches exhibit a good resilience against noise and artifacts [8], but they are also computationally expensive if the frame window is large, a problem which can be alleviated by fast methods to approximate the median [29,33]. The second kind of approaches estimates the probability density function (pdf) of the pixel values, but it does not assume any particular probability distribution for them, i.e. non parametric methods are used. Gaussian kernels are commonly chosen for this purpose [9,14]; the fundamental parameters to be tuned in this case are the number of kernels and their bandwidth. There is also a need to reduce the inherent computational load of kernel density estimation, which can be done by dropping irrelevant features. Subspace based methods try to find a subspace of the space of all possible images where the background of the scene lies, so that departures from that subspace can be detected as foreground objects [31,46,44,42]. Finally, the fourth group of methods assumes that the pixel values follow certain probability distribution, usually a mixture of Gaussians, and then it tries to estimate its parameters; this means that they are parametric methods. They tend to have less memory and time requirements than non parametric ones, since the number of parameters is relatively small. This is one of the causes of their popularity [45,40,51,17], along with the possibility to introduce specific mechanisms to tackle the above mentioned challenges [4].

Even though there is a large number of background subtraction algorithms based on probabilistic mixtures, a majority of them stick to a set of simplifications which can reduce their performance. On one hand, most proposals use the RGB pixel values as inputs [34]. On the other hand, some well established and frequently used background modeling algorithms use the same variance for all the input variables [17,41,51], although there is no

☆ This paper has been recommended for acceptance by Andrea Prati.

* Corresponding author.
*E-mail addresses:* xavierprof@hotmail.com (F.J. López-Rubio), ezeqlr@lcc.uma.es (E. López-Rubio).
*URL:* http://www.lcc.uma.es/%7Eezeqlr/index-en.html (E. López-Rubio).

http://dx.doi.org/10.1016/j.cviu.2014.12.007
1077-3142/

fundamental reason not to use the full covariance matrix. In other words, the use of full covariances is an option which exists in theory but is very rarely implemented in practice. A spherical covariance matrix might not be the optimal choice because the variance of the input variables can be different, which means that those models do not adapt to the specific dispersion of each variable. For example, the variabilities of median filtered features are expected to be much lower than those of non median filtered features. Also, features based on edge information (high pass filters) tend to vary more than features based on low pass filters. Moreover, the above mentioned models do not consider the covariances among the variables, so they treat them as if they were independent, which is not the case. For example, the red, green and blue color components of a pixel will typically grow or diminish together as the lighting increases or decreases, respectively. This means that RGB color components are strongly correlated. Another example is edge features in the horizontal and vertical directions, since textured objects will have high values of the features in both directions, while homogeneous objects will have low values of the features in both directions.

Our aim here is to develop a method which overcomes the limitations we have just outlined, along with a set of relevant features that yields adequate results. Our proposal defines a probabilistic model which handles any number of pixel features. It also accounts for the correlations among the features, so that a more realistic model is obtained.

The structure of this paper is as follows. First of all, a review of previous work about probabilistic background models and feature selection is done in Section 2. Then the proposed probabilistic mixture model and the corresponding learning algorithm are considered in Section 3. The set of pixel features that we have chosen are defined and studied in Section 4. Section 5 is devoted to the experimental results, with comparisons with state-of-the-art approaches. Finally, Sections 6 and 7 deal with the discussion of the most relevant characteristics of our proposal and the conclusions, respectively.

## 2. Related work

As outlined above, the design of a realistic mixture model for foreground detection can be decomposed into two decisions: the choice of the probability model for each mixture component and the selection of the number and kind of input features. Next a review of previous literature on these two topics is carried out, with indications of possible enhancements.

Most methods rely on the assumption that all the variables have the same variance, and that all of them are independent. This translates to Gaussian mixture components with spherical covariance matrices, i.e. matrices of the form $\sigma^2\mathbf{I}$, where $\sigma^2$ is the common variance and $\mathbf{I}$ is the identity matrix. It has long been known that the assumption of a spherical covariance matrix does not conform to real scenes. In [12] it is found that the estimation of the mean vector is reliable, but the covariance matrix does not conform to a spherical model. The distribution of values for a background pixel in the standard RGB space conforms to a cylinder rather than a sphere, which renders the spherical model unrealistic [19,5]. The cylindrical model is proposed to overcome this problem. Under this model the major axis of the cylinder extends to the black point, and the position along this axis corresponds to luminance. The distance of a point to the major axis determines the radius of the cylinder, which is called the color distortion. The cylindrical model has the disadvantage that it is not associated to any probability density, since it only defines a distance measure of how close a color is to the model. Hence it is not possible to assign a probability that a pixel belongs to the background, and consequently the amount of learning of the background model of a pixel cannot be adapted to the likelihood that the observed color corresponds to the background, which affects the learning process negatively. On the other hand, a Gaussian with a full covariance matrix can model an elongated shape with any position, thickness and orientation, and it produces the probability that a pixel belongs to the background, so it is a viable alternative to fit the data.

A commonly used alternative is to use several spherical Gaussians for the background [40,49,6], rather than one [45]. Comparative studies show that more than one spherical Gaussians perform better than a single spherical Gaussian [11]. Nevertheless, a mixture of spherical Gaussians can only give a rough approximation of an elongated cluster of data unless the number of Gaussians grows considerably, because the points which are midway between the mean vectors of two successive Gaussian components have a lower probability density than those closer to a mean vector. Again, a Gaussian with a full covariance matrix does not have this problem, since the probability density is smooth through the elongated cluster. This is confirmed by the study in [3], which points out that a single Gaussian with a full covariance matrix outperforms a model with multiple spherical Gaussians in some situations. It is reported that this is because the covariances are able to adapt to background instabilities.

Last but not the least Gaussian mixture components cannot model the foreground adequately unless a large amount of them is used. This is because incoming foreground objects can have any aspect, so any mixture component used to model the foreground should have a substantially flat profile, i.e. no prominent modes. However, most algorithms neglect this fact and assign a small number of spherical Gaussians to the foreground [18,6]. This leads to very large variances in the foreground Gaussians as they try to adapt to heterogeneous foreground input samples. Moreover, a large part of the probability mass of these Gaussians could be out of the support of the real input distribution, since the Gaussians must adapt to the input samples which lie near the borders of the support. That is, they are modeling regions where no inputs can exist. Of course this is also true for the Gaussians assigned to the background, but in this case the effect is not so serious because the background samples are expected to be more concentrated, so that the spread of the Gaussians is smaller and little probability mass is out of the support of the real input. On the other hand, the assignment of a Gaussian to the background or the foreground is a difficult problem on its own. This situation calls for an uninformative prior for the foreground, i.e. a single flat mixture component such as a uniform distribution [35,13], which is the approach that we advocate here.

The second fundamental decision to be made when a foreground detection algorithm is developed deals with the set of input features. The most straightforward option is to use the raw RGB information from the camera, and it is chosen by many proposals. It has the advantage of its speed and ease of implementation, but it is widely acknowledged that it suffers from limitations due to illumination changes, among other artifacts. Separation of luminance and chrominance can help to reduce these undesirable effects, as done in [45], where the YUV color space is employed. The Y channel conveys the luminance information, while U and V channels carry chrominance. A further development is to remove the luminance information completely, which is commonly done by dividing the RGB values by their sum to yield the normalized RGB set of features [38,28,2,9]. This attains certain resilience to illumination changes, but the differing optical characteristics of objects imply that normalized RGB data are still subject to variation when lights are switched on or off. A way to remove the dependence from color and lighting is to use texture features [48,16], although they are

not useful for scenes which contain objects or background with little or no texture.

Combinations of features of different kinds have proved to perform adequately in many situations. Spectral, spatial and temporal features are combined under a Bayesian classification framework in [21]. It is also possible to choose the features by means of a boosting algorithm [32], although a warmup period with no foreground objects is required. It is worth noting that these works have fundamental differences with our approach, since [21] estimates the background and foreground probabilities by multivariate histograms, while [32] considers kernel density estimators with spherical Gaussians, which has the disadvantages that have been explained above.

In principle, any foreground detection algorithm which is based on a probabilistic mixture which includes spherical Gaussians could be adapted to accommodate full covariance matrices with any number of variables. However, each algorithm makes its own simplifying assumptions which do not hold for full covariance matrices. Hence, each of them would have to be reformulated from scratch. Consequently, we restrict our attention here to the adaptation of our earlier model [24], which was oriented to the RGB color space, in order to accommodate for any number and kind of pixel features.

## 3. Background model

In this section we present a probabilistic mixture model for the foreground detection problem. As mentioned in Section 2, it is an improvement of the model [24]. The new model is defined in Section 3.1, and the learning algorithm is detailed in Section 3.2. It is necessary to employ specific procedures to manage quantization noise (Section 3.3) and sudden background changes (Section 3.4). The initialization is outlined in Section 3.5. Finally, the implementation of the model is detailed in Section 3.6.

### 3.1. Definition

We propose to use the stochastic approximation framework to train a mixture which models the distribution of pixel feature values $\mathbf{t}(\mathbf{x}) \in \mathbb{R}^D$ at frame coordinates $\mathbf{x} = (j,k)$, where $D$ is the number of pixel features of interest. There is one Gaussian component for the background and one uniform component for the foreground. This leads to the following probabilistic model for the distribution of feature values at any given position, where we drop the position coordinates $\mathbf{x}$ for the sake of simplicity:

$$\begin{aligned} p(\mathbf{t}) &= \pi_{Back} p(\mathbf{t}|Back) + \pi_{Fore} p(\mathbf{t}|Fore) \\ &= \pi_{Back} K\left(\mathbf{t}|\boldsymbol{\mu}_{Back}, \mathbf{C}_{Back} + \boldsymbol{\Psi}\right) + \pi_{Fore} U(\mathbf{t}) \end{aligned} \tag{1}$$

We have:

$$K(\mathbf{t}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-D/2} \det(\boldsymbol{\Sigma})^{-1/2} \exp\left((\mathbf{t}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{t}-\boldsymbol{\mu})\right) \tag{2}$$

$$U(\mathbf{t}) = \begin{cases} 1/Vol(H) & \text{iff } \mathbf{t} \in H \\ 0 & \text{iff } \mathbf{t} \notin H \end{cases} \tag{3}$$

$$\boldsymbol{\mu}_{Back} = E[\mathbf{t}|Back] \tag{4}$$

$$\boldsymbol{\mu}_{Fore} = E[\mathbf{t}|Fore] \tag{5}$$

$$\mathbf{C}_{Back} = E\left[(\mathbf{t}-\boldsymbol{\mu}_{Back})(\mathbf{t}-\boldsymbol{\mu}_{Back})^T | Back\right] \tag{6}$$

where $H$ is the support of the uniform pdf, $Vol(H)$ is the $D$-dimensional volume of $H$, and $\boldsymbol{\Psi}$ is a constant diagonal matrix which accounts for the quantization and compression noise (see Section 3.3). Let us note $d_h, e_h$ the minimum and maximum values of pixel feature $h \in \{1,2,\ldots,D\}$, respectively. Then we have:

$$H = [d_1, e_1] \times [d_2, e_2] \times \ldots \times [d_D, e_D] \tag{7}$$

$$Vol(H) = \prod_{h=1}^{D} (d_h - e_h) \tag{8}$$

It must be highlighted that $\boldsymbol{\mu}_{Fore}$ is not used to compute the probability densities in any way. It is only needed when a sudden background change is detected, as explained in Section 3.4.

The Gaussian component $K(\mathbf{t}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is able to capture a wide range of dynamic backgrounds because $\mathbf{C}_{Back}$ (and hence $\mathbf{C}_{Back} + \boldsymbol{\Psi}$) are not restricted to be diagonal matrices. On the other hand, the uniform component $U(\mathbf{t})$ is able to model foreground objects of any kind equally well.

The class probabilities of the observed value $\mathbf{t}_n$ of a pixel at time step $n$ are given by:

$$\begin{aligned} &\forall i \in \{Back, Fore\},\ R_{ni} = P(i|\mathbf{t}_n) \\ &= \frac{\pi_i p(\mathbf{t}_n|i)}{\pi_{Back} p(\mathbf{t}_n|Back) + \pi_{Fore} p(\mathbf{t}_n|Fore)} \end{aligned} \tag{9}$$

That is, $R_{n,Back}$ and $R_{n,Fore}$ are the posterior probabilities that the pixel belongs to the background or the foreground, respectively.

Under the Bayesian framework, a pixel is declared to belong to the foreground at time step $n$ if and only if $R_{n,Fore} > R_{n,Back}$. However, it is sometimes advantageous to introduce a probability threshold $\rho \in [0,1]$ so that it is decided that a pixel belongs to the foreground when the following condition holds:

$$R_{n,Fore} > \rho \tag{10}$$

Unless otherwise noticed we will make Bayesian decisions, i.e. $\rho = 0.5$.

### 3.2. Learning

Several previous approaches to the background modeling problem have used algorithms which can be related to the stochastic approximation framework. For example, the parameter update scheme of the adaptive Gaussian mixture model in [40,18,30] can be seen as a stochastic approximation learning algorithm for a mixture of Gaussians. From the theoretical point of view this is because stochastic approximation applied to a mixture of Gaussians is a recursive (online) method for the maximization of the likelihood [50], which has led to the algorithm in [51]. This framework can also be applied to probabilistic self-organizing maps, as demonstrated in [25].

When applied to the probabilistic model given in the previous section, the Robbins-Monro stochastic approximation algorithm yields the following update equations at time step $n$ for an observed pixel value $\mathbf{t}_n$ (see [24] for details):

$$\forall i \in \{Back, Fore\},\ \pi_i(n) = (1-\epsilon)\pi_i(n-1) + \epsilon R_{ni} \tag{11}$$

$$\forall i \in \{Back, Fore\},\ \mathbf{m}_i(n) = (1-\epsilon)\mathbf{m}_i(n-1) + \epsilon R_{ni} \mathbf{t}_n \tag{12}$$

$$\forall i \in \{Back, Fore\},\ \boldsymbol{\mu}_i(n) = \frac{\mathbf{m}_i(n)}{\pi_i(n)} \tag{13}$$

$$\begin{aligned} \mathbf{M}_{Back}(n) &= (1-\epsilon)\mathbf{M}_{Back}(n-1) \\ &\quad + \epsilon R_{n,Back} (\mathbf{t}_n - \boldsymbol{\mu}_{Back}(n))(\mathbf{t}_n - \boldsymbol{\mu}_{Back}(n))^T \end{aligned} \tag{14}$$

$$\mathbf{C}_{Back}(n) = \frac{\mathbf{M}_{Back}(n)}{\pi_{Back}(n)} \tag{15}$$

where $\epsilon$ is a constant step size, which we have set to $\epsilon = 0.01$ in all the experiments. The step size is constant (and not decaying) because the system is time varying, as considered in Section 3.2 of [20]. The larger $\epsilon$, the bigger the modifications of the model parameters. Eq. (11) is used to update the a priori probabilities $\pi_i$ of the foreground and background classes, depending on their posterior probabilities according to the observed data. The mean vectors $\boldsymbol{\mu}_i$ are modified through (13), where $\mathbf{m}_i$ are auxiliary variables which store a weighted averages of the observed data with the posterior probabilities being the weights (12). Finally, the covariance matrix of the background class $\mathbf{C}_{Back}$ is learned by (15), with $\mathbf{M}_{Back}$ playing the role of an auxiliary variable which stores a weighted average where the weights are the posterior probabilities (14).

### 3.3. Quantization noise estimation

The diagonal noise matrix $\boldsymbol{\Psi}$ models the effects of pixel quantization and compression, which are sources of background noise. This is estimated separately from $\mathbf{C}_{Back}$ because these effects appear and disappear suddenly from one frame to the next, so that they cannot be captured adequately by the stochastic approximation learning algorithm. Stochastic approximation filters out these abrupt changes and does not learn from them. In contrast to this, the elements of $\mathbf{C}_{Back}$ model the variability of the pixel data which is due to the true background color signal, which changes smoothly through time. This smooth variation can be learned by stochastic approximation, i.e. it is not filtered out by the update equations of Section 3.2. In other words, $\mathbf{C}_{Back}$ models the low frequency component of the background color signal while $\boldsymbol{\Psi}$ models the high frequency component. Hence, $\boldsymbol{\Psi}$ and $\mathbf{C}_{Back}$ model two fundamentally different processes, both of which affect the observed color data.

We assume that the quantization and compression effects are approximately equal for all the pixels, so that an offline estimation of a global and constant value of $\boldsymbol{\Psi}$ is carried out. The matrix is given by

$$\boldsymbol{\Psi} = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \sigma_D^2 \end{pmatrix} \tag{16}$$

The noise parameters may be estimated by comparing the original, uncompressed pixels $\boldsymbol{\tau}(\mathbf{x})$ with the observed (compressed) pixels $\mathbf{t}(\mathbf{x})$ for a selection of videos:

$$\forall i \in \{1, \dots, D\}\ \sigma_i^2 = E\left[\left(\tau_i(\mathbf{x}) - t_i(\mathbf{x})\right)^2\right] \tag{17}$$

where the expectation is computed by averaging over all the pixels in the video selection. Please note that this estimation is carried out offline (that is, $\boldsymbol{\Psi}$ is a constant throughout the online learning). In practice, it is often found that the original values $\boldsymbol{\tau}(\mathbf{x})$ cannot be obtained because the video is available in compressed format only. This problem is solved by estimating the original values by application of a rotationally symmetric Gaussian low pass filter of size $3 \times 3$ pixels with standard deviation 0.5. The filtered values are taken as estimations $\widehat{\tau}_i$ of the original values $\tau_i$:

$$\forall i \in \{1, \dots, D\}\ \sigma_i^2 \approx E\left[\left(\widehat{\tau}_i(\mathbf{x}) - t_i(\mathbf{x})\right)^2\right] \tag{18}$$

where the expectation is computed by averaging over all the pixels in the first frame of the video.

### 3.4. Sudden background change detection

Sometimes an object integrates into the background, for example when a car is parked. This produces a sudden change in the background distributions of many pixels, i.e. those which are inside the arriving object. The same effect happens when an object which was a part of the background departs. We must take care of these events separately [21]. As pointed out by Kushner & Yin in Section 3.2 of [20], stochastic approximation of time varying systems is only advisable when the rate of change of the estimated parameters is slow. Hence, these sudden changes must be managed in a different procedure which detects these situations that the stochastic approximation framework is unable to model.

The detection procedure needs a previous offline study. Let $\chi$ be the ground truth for a particular pixel:

$$\chi = \begin{cases} 1 & \text{iff the pixel belongs to the foreground} \\ 0 & \text{iff the pixel belongs to the background} \end{cases} \tag{19}$$

Then, let $\widetilde{\chi}$ be the estimation of $\chi$ carried out by the basic algorithm, i.e. without sudden background change detection. That is, $\widetilde{\chi} = 1$ iff the pixel is classified as foreground by the basic algorithm, and $\widetilde{\chi} = 0$ otherwise.

The basic algorithm works correctly if and only if $\widetilde{\chi} = \chi$. We define three disjoint random events: undetected background change (*Change*), correct foreground detection (*Good*) and other situations that we are not interested in (*Other*):

$$Change \equiv \left(\widetilde{\chi} = 1\right) \wedge \left(\chi = 0\right) \tag{20}$$

$$Good \equiv \left(\widetilde{\chi} = 1\right) \wedge \left(\chi = 1\right) \tag{21}$$

$$Other \equiv \widetilde{\chi} = 0 \tag{22}$$

At frame $n$, the number of frames $z(n)$ that the pixel has been continuously classified as foreground is defined as follows:

$$z(n) = \max\left\{N \mid \widetilde{\chi}(n) = \widetilde{\chi}(n-1) = \dots = \widetilde{\chi}(n-N) = 1\right\} \tag{23}$$

Please note that $z(n)$ is readily computed by a running counter which is incremented each frame that $\widetilde{\chi} = 1$ and reset to zero each time that $\widetilde{\chi} = 0$. The offline task consists in estimating the probabilities $P(Good \mid z, \widetilde{\chi} = 1)$ and $P(Change \mid z, \widetilde{\chi} = 1)$. This is accomplished by counting the number of times that *Change* and *Good* are verified for a certain value of $z$, where we take as input samples the values of $z(n)$ for all frames $n$ and all pixels such that $\widetilde{\chi} = 1$. These pixel counts are noisy functions of $z$ that are smoothed by kernel regression before estimating the probabilities.

Then we compute a threshold $Z$:

$$Z = \min\left\{z \mid P(Change|z, \widetilde{\chi} = 1) > \frac{1}{2}\right\} \tag{24}$$

The computation of $Z$ finishes the offline study. We have found by studying several videos that $Z = 250$ is a good compromise between the adaptation to sudden changes and the stability of the background model, so we have used that value in all the experiments.

When the online system is operating, in each pixel we maintain a running counter of the number of frames $z$. If we have at frame $n$ that a pixel is classified as foreground and $z(n) \geqslant Z$, then that pixel must be reset. The pixel reset procedure involves setting $\mathbf{C}_{Back} = \mathbf{0}$, which corresponds to setting $\boldsymbol{\Sigma} = \boldsymbol{\Psi}$ as the covariance matrix for the Gaussian mixture component in (2). Please note that it is $\boldsymbol{\Sigma}$ and not $\mathbf{C}_{Back}$ the matrix that must be invertible, and $\boldsymbol{\Psi}$ is never zero, so the matrix inversion in (2) is always possible. Additionally, we swap $\boldsymbol{\mu}_{Back}$ with $\boldsymbol{\mu}_{Fore}$, and $\mathbf{m}_{Back}$ with $\mathbf{m}_{Fore}$.

### 3.5. Initialization

Here we outline the initialization procedure to be carried out at the beginning of the execution of the method. The first $L$ frames of the analyzed video sequence are used for this purpose, with

$L = 100$ in our experiments. The background model of every pixel is set up according to the $L$ pixel value samples obtained from those first frames. Hence we have:

$$\forall i \in \{Back, Fore\},\ \pi_i(0) = \frac{1}{2} \tag{25}$$

$$\mathbf{m}_{Back}(0) = \frac{\pi_{Back}(0)}{L}\sum_{j=1}^{L}\mathbf{t}_j \tag{26}$$

$$\boldsymbol{\mu}_{Back}(0) = \frac{\mathbf{m}_{Back}(0)}{\pi_{Back}(0)} \tag{27}$$

$$\mathbf{M}_{Back}(0) = \frac{\pi_{Back}(0)}{L}\sum_{j=1}^{L}(\mathbf{t}_j - \boldsymbol{\mu}_{Back}(0))(\mathbf{t}_j - \boldsymbol{\mu}_{Back}(0))^T \tag{28}$$

$$\mathbf{C}_{Back}(0) = \frac{\mathbf{M}_{Back}(0)}{\pi_{Back}(0)} \tag{29}$$

On the other hand, the initialization of $\boldsymbol{\mu}_{Fore}$ is not critical, since its value is not used for the first $Z$ frames at least. We might start with the point at the center of $H$:

$$\boldsymbol{\mu}_{Fore}(0) = \left(\frac{1}{2}(d_1 + e_1), \ldots, \frac{1}{2}(d_D + e_D)\right)^T \tag{30}$$

$$\mathbf{m}_{Fore}(0) = \pi_{Fore}(0)\boldsymbol{\mu}_{Fore}(0) \tag{31}$$

Finally, the running counter is initialized with $z(0) = 0$.

### 3.6. Implementation

In this section the details of the implementation of the proposed method are discussed. From now on we will name it Multiple Feature Background Model (MFBM). Since the number of features $D$ is not restricted, a fast and numerically stable implementation is of paramount importance to the practical applicability of the procedure.

The slowest part of the method is Eq. (2), i.e. the computation of the Gaussian probability density. It is $O\left(D^3\right)$ due to the inversion of the covariance matrix $\boldsymbol{\Sigma}$ and the computation of its determinant. The other equations are $O\left(D^2\right)$ or lower. Moreover, matrix inversion is prone to numerical errors. Therefore, all efforts must be devoted to tune the implementation of (2).

The first step is to compute the Cholesky decomposition of $\boldsymbol{\Sigma}$, which is always possible because $\boldsymbol{\Sigma}$ is known to be positive definite:

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T \tag{32}$$

where $\mathbf{L}$ is a lower triangular matrix of size $D \times D$. Then we note:

$$\mathbf{y} = \mathbf{L}^{-1}(\mathbf{t} - \boldsymbol{\mu}) \tag{33}$$

which allows to obtain $\mathbf{y}$ as the solution of a lower triangular linear system of equations:

$$\mathbf{L}\mathbf{y} = \mathbf{t} - \boldsymbol{\mu} \tag{34}$$

Then we get:

$$(\mathbf{t} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{t} - \boldsymbol{\mu}) = \mathbf{y}^T\mathbf{y} \tag{35}$$

so that the squared Mahalanobis distance in (2) is obtained without computing $\boldsymbol{\Sigma}^{-1}$ explicitly. The described procedure is faster and more accurate than the explicit computation of $\boldsymbol{\Sigma}^{-1}$.

The final step is the computation of the determinant of $\boldsymbol{\Sigma}$:

$$\det(\boldsymbol{\Sigma}) = \det(\mathbf{L})\det\left(\mathbf{L}^T\right) = (\det(\mathbf{L}))^2 \tag{36}$$

where $\det(\mathbf{L})$ is obtained as the product of the elements of the diagonal of $\mathbf{L}$. Again, this is a fast and accurate way to compute $\det(\boldsymbol{\Sigma})$.

## 4. Background features

Now we present the set of background features we have considered (Section 4.1). Some of them are well known, while others are novel. We have adjusted all of them so that their range of values is the interval $[0, 1]$. Section 4.2 studies the properties of the features, which have guided their choice. The last part of this section tests the detection performance of the features over three benchmark videos in order to check their suitability for foreground object detection (Section 4.3).

### 4.1. Feature description

The first three features are the red, green and blue channels of the input frame. We note them as $\mathbf{F}_1, \mathbf{F}_2$ and $\mathbf{F}_3$, respectively, where $\mathbf{F}_i$ are matrices of size $N \times M$, i.e. the size of the input frames. Here we assume that the pixel values are real numbers in the interval $[0, 1]$.

As explained in Section 2, the normalized RGB channels are frequently used due to their robustness with respect to illumination changes [38,28,2,9]. They are obtained as follows:

$$\forall i \in \{4, 5, 6\},\ \mathbf{F}_i = \mathbf{F}_{i-3}\text{ø}\mathbf{G} \tag{37}$$

$$\mathbf{G} = \mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3 \tag{38}$$

where ø stands for the elementwise division operator and $\mathbf{G}$ is the sum of the three original channels. In case that some elements of $\mathbf{G}$ are zero, we define the result of the division to be zero for these elements.

Next we consider the six Haar-like features of size $9 \times 9$ pixels considered in [14], which are weak features that convey texture information [43]:

$$\forall i \in \{7, \ldots 12\},\ \mathbf{F}_i = \frac{1}{3\sum_{j,k=1}^{9} h_{i-6,j,k}}\mathbf{G} * \mathbf{H}_{i-6} \tag{39}$$

where $*$ stands for the convolution operator. The six filters $\mathbf{H}_i$ are defined as follows:

$$\mathbf{H}_1 = \begin{pmatrix}\mathbf{1}_{9,3} & \mathbf{0}_{9,3} & \mathbf{1}_{9,3}\end{pmatrix} \tag{40}$$

$$\mathbf{H}_2 = \mathbf{H}_1^T \tag{41}$$

$$\mathbf{H}_3 = \frac{1}{2}\begin{pmatrix}2\cdot\mathbf{1}_{4,4} & \mathbf{1}_{1,1} & \mathbf{0}_{4,4} \\ \mathbf{1}_{1,1} & \mathbf{1}_{1,1} & \mathbf{1}_{1,1} \\ \mathbf{0}_{4,4} & \mathbf{1}_{1,1} & 2\cdot\mathbf{1}_{4,4}\end{pmatrix} \tag{42}$$

$$\mathbf{H}_4 = \frac{1}{2}\begin{pmatrix}2\cdot\mathbf{1}_{9,4} & \mathbf{1}_{9,1} & \mathbf{0}_{9,4}\end{pmatrix} \tag{43}$$

$$\mathbf{H}_5 = \mathbf{H}_4^T \tag{44}$$

$$\mathbf{H}_6 = \begin{pmatrix}1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1\end{pmatrix} \otimes \mathbf{1}_{3,3} \tag{45}$$

where $\otimes$ stands for Kronecker product, $\mathbf{0}_{m,n}$ stands for a matrix of zeros of size $m \times n$, and $\mathbf{1}_{m,n}$ stands for a matrix of ones of size $m \times n$.

Gradient features are also useful because they are less affected by illumination changes, while they provide local texture information [21]. We estimate the gradient components with Sobel

operator, and then we scale and translate them to lie in the interval $[0,1]$. As mentioned before, the background model is not affected by these transformations:

$$\mathbf{F}_{13} = \frac{1}{2}\mathbf{1}_{M,N} + \frac{1}{3\cdot 8}\mathbf{G} * \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{46}$$

$$\mathbf{F}_{14} = \frac{1}{2}\mathbf{1}_{M,N} + \frac{1}{3\cdot 8}\mathbf{G} * \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{47}$$

Next we introduce two features which consider both color information and a pixel adjacent to the pixel at hand, as a local indication of color texture. The first one is the red channel of the current pixel, normalized with the current pixel and the pixel immediately to the left of the current one:

$$\mathbf{F}_{15} = \mathbf{F}_1 \oslash (\mathbf{G} + \mathbf{G}\mathbf{U}_n) \tag{48}$$

where $\mathbf{U}_h$ is the $h \times h$ upper shift matrix:

$$\forall j,k \in \{1,\ldots h\},\ U_{j,k} = \mathbb{I}(j+1=k) \tag{49}$$

with $\mathbb{I}$ standing for the indicator function.

The second one is the green channel of the pixel immediately to the lower right of the current one, normalized with the current pixel:

$$\mathbf{F}_{16} = (\mathbf{U}_m \mathbf{F}_2 \mathbf{Q}_n) \oslash \mathbf{G} \tag{50}$$

where $\mathbf{Q}_h$ is the $h \times h$ lower shift matrix:

$$\forall j,k \in \{1,\ldots h\},\ Q_{j,k} = \mathbb{I}(j=k+1) \tag{51}$$

In order to include some robust features, we add versions of features 1 to 6 processed with the bidimensional median filter of window size $5 \times 5$ pixels, that we note $\text{median}_{5,5}(\cdot)$. The filter size has been chosen as a balance between the robustness (which needs larger sizes) and the precision of the object detection (which calls for a smaller size). This way we have:

$$\forall i \in \{17,\ldots,22\},\ \mathbf{F}_i = \text{median}_{5,5}(\mathbf{F}_{i-16}) \tag{52}$$

Finally, we use two small filters of size $3 \times 3$ pixels to extract the texture information from the close vicinity of the pixel at hand:

$$\mathbf{F}_{23} = \frac{1}{3\cdot 6}\mathbf{G} * \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \tag{53}$$

$$\mathbf{F}_{24} = \frac{1}{3\cdot 8}\mathbf{G} * \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{54}$$

This completes a set of 24 features. Features 15 to 22 are novel, while the others are more or less known in previous literature. Given a set of selected features $\mathcal{F} \subset \{1,\ldots,24\}$, the input pattern for the pixel at frame coordinates $(j,k)$ is given by:

$$\mathbf{t}_{j,k} = \left(F_{i,j,k}\right)_{i\in\mathcal{F}} \tag{55}$$

where $\mathbf{t}_{j,k} \in [0,1]^D$ because we have added the necessary scalings and translations in the previous feature definitions (37)–(54).

### 4.2. Properties of the features

Next the properties of the above presented features that have guided their choice are studied. As mentioned in Section 4.1, one of the most typical difficulties encountered by foreground detection algorithms is that of illumination changes. In addition to this, quantization and compression artifacts are noise sources that can hamper the detection performance. In this section, a theoretical study of the robustness of the considered features with respect to these inconveniences is carried out. Furthermore, the size of the local pixel neighborhood which affects the value of a feature is also investigated.

To a first approximation, an illumination change can be expressed as a linear transform which scales the three RGB channels in the same way, so that the RGB features after an illumination change are given by:

$$\forall i \in \{1,2,3\},\ \bar{\mathbf{F}}_i = \lambda \mathbf{F}_i \tag{56}$$

where $\lambda$ is the scaling factor. This implies that features 1 to 3 are affected by illumination changes, which is undesirable. On the other hand the normalized features, i.e. those where the RGB values are divided by the sum of the three RGB values, satisfy that

$$\forall i \in \{4,5,6,15,16\},\ \bar{\mathbf{F}}_i = \mathbf{F}_i \tag{57}$$

which means that features 4, 5, 6, 15 and 16 are invariant to illumination variations to a first approximation. The Haar-like, gradient and small texture filter features behave the same way as the RGB features:

$$\forall i \in \{7,\ldots,14,23,24\},\ \bar{\mathbf{F}}_i = \lambda \mathbf{F}_i \tag{58}$$

where the translation of the gradient features (13 and 14) to make them lie in the interval $[0,1]$ is not affected by the $\lambda$ factor. The median filtered features (17 to 22) inherit the behavior of their original features (1 to 6).

Robustness under quantization and compression artifacts means that the value of a feature is not affected when a small fraction of pixels is corrupted by these noise sources. Out of the 24 chosen features, this desirable property is satisfied only by those that are based on robust statistics, i.e. the median filter based features (17 to 22). For the rest of the features, even a single corrupted pixel affects the final output.

Finally we study the radius of the local pixel neighborhood which contains all the pixels which affect the value of the feature at a given position. This is important because a large neighborhood means that the feature is less sensitive to small details, which can be beneficial if these details are irrelevant, but can also be a problem if these details correspond to portions of the foreground objects to be detected. The best radius will depend on the noise level, the size of the objects to be detected, and the robustness of the features. Feature 1 to 6 depend on the current pixel only, so the radius is zero. The Haar like features (7 to 12) are defined over $9 \times 9$ windows, so their radius is 4. The gradient and small texture features (13, 14, 23 and 24) are defined over $3 \times 3$ windows, so their radius is 1. The normalized texture features (15 and 16) also have radius 1 because they depend on the current pixel and on an adjacent one. Finally, the median filter features (17 to 22) have radius 2, because the filter window has size $5 \times 5$.

Table 1 summarizes the properties of the studied features. The 'Yes' entries correspond to desirable characteristics of the features, while it can be observed that we have chosen a wide range of radii so as to cover many possible situations.

### 4.3. Detection performance

Next we measure the detection performance of the selected features. We have used the standard benchmark sequences Campus, Fountain and Video4. The Campus and Fountain sequences created by Li et al. [21] are available in their website.[1] The Video4 sequence

[1] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html.

**Table 1**
Summary of the properties of the considered features. For the illumination column, 'Yes' means that the feature is invariant to illumination changes to a first approximation. For the Noise column, 'Yes' means that the feature is not affected by a small fraction of pixels corrupted by quantization or compression artifacts. For the Radius column, the distance to the farthest pixel which affects the value of the feature at the current pixel is given.

| Feature | Illumination | Noise | Radius |
|---|---|---|---|
| 1 | No | No | 0 |
| 2 | No | No | 0 |
| 3 | No | No | 0 |
| 4 | Yes | No | 0 |
| 5 | Yes | No | 0 |
| 6 | Yes | No | 0 |
| 7 | No | No | 4 |
| 8 | No | No | 4 |
| 9 | No | No | 4 |
| 10 | No | No | 4 |
| 11 | No | No | 4 |
| 12 | No | No | 4 |
| 13 | No | No | 4 |
| 14 | No | No | 4 |
| 15 | Yes | No | 1 |
| 16 | Yes | No | 1 |
| 17 | No | Yes | 2 |
| 18 | No | Yes | 2 |
| 19 | No | Yes | 2 |
| 20 | Yes | Yes | 2 |
| 21 | Yes | Yes | 2 |
| 22 | Yes | Yes | 2 |
| 23 | No | No | 1 |
| 24 | No | No | 1 |

was created for an international competition on background/foreground segmentation, and it is also publicly available.[2]

We have selected six quantitative performance measures to compare the segmentation methods we have considered. First of all, the accuracy of a method is obtained as follows (higher is better):

$$AC = \frac{\mathrm{card}(A \cap B)}{\mathrm{card}(A \cup B)} \tag{59}$$

where 'card' stands for the number of elements of a set, $A$ is the set of all pixels which belong to the foreground, and $B$ is the set of all pixels which are classified as foreground by the analyzed method:

$$A = \{\mathbf{t} \mid \chi(\mathbf{t}) = 1\} \tag{60}$$

$$B = \{\mathbf{t} \mid \widetilde{\chi}(\mathbf{t}) = 1\} \tag{61}$$

On the other hand, the proportions of false negatives and false positives (lower is better) are given by:

$$FN = \frac{\mathrm{card}(A \cap \overline{B})}{\mathrm{card}(A \cup B)} \tag{62}$$

$$FP = \frac{\mathrm{card}\left(\overline{A} \cap B\right)}{\mathrm{card}(A \cup B)} \tag{63}$$

From set theory we know that:

$$AC,\ FN,\ FP \in [0, 1] \tag{64}$$

$$AC + FN + FP = 1 \tag{65}$$

[2] http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC.

The optimal performance would be achieved for $AC = 1,\ FN = 0,\ FP = 0$. On the other hand, the worst possible performance corresponds to $AC = 0,\ FN + FP = 1$.

The precision and recall measures can be computed as follows (higher is better):

$$PR = \frac{\mathrm{card}(A \cap B)}{\mathrm{card}(B)} \tag{66}$$

$$RC = \frac{\mathrm{card}(A \cap B)}{\mathrm{card}(A)} \tag{67}$$

$$PR,\ RC \in [0, 1] \tag{68}$$

Often there is a tradeoff between precision and recall, i.e. we can make one of them grow as desired at the expense of diminishing the other.

Finally, the F-measure combines precision and recall into an overall performance measure:

$$FM = \frac{2 \cdot PR \cdot RC}{PR + RC} \tag{69}$$

The performance measures defined above are calculated for each sequence by averaging the yields obtained in each frame. However, we can also define these measures considering the complete sequence, where $A$ is the sum of all the pixels corresponding to the foreground throughout the sequence and B the sum of the pixels considered as foreground for the method. We denote as F-measure' the measure computed in this way.

We have made tests with all possible combinations of pairs of features and triples of features from those described in Section 4.1, and we have computed the six performance measures defined above, averaged over the three mentioned videos. It has been found that in general the best performance results are obtained by combining features from this set: $\{1, 2, \ldots, 6, 17, 18, \ldots, 24\}$. So, we have used this reduced set of 14 features to carry out tests with combinations of 4 and 5 features. Please note that it would have been unfeasible to carry out tests with all the combinations of 4 and 5 features from the original set of 24 features.

We must take into account that there are two pairs of measures which constitute tradeoffs: false positives versus false negatives and precision versus recall. In order to identify easily the best performing combinations of features under this multiobjective optimization context, we consider the Pareto front concept [22,1,23,37]. In particular, we compute the Pareto front of the set of tested combinations of $i$ features, for $i \in \{2, 3, 4, 5\}$ and for each of the following pairs of measures: $\{FN, FP\}$; $\{PR, RC\}$; and $\{AC, FM\}$. A combination of $i$ features belongs to the Pareto front of the set of combinations of $i$ features for a pair of performance measures if and only if there is no other combination of $i$ features that is better in both performance measures. The configurations (combinations of features) which belong to a Pareto front are the most outstanding ones.

Figs. 1–6 depict the obtained results. Each data point in the plots corresponds to the average performance measures obtained by a particular set of features, computed over the three benchmark videos mentioned above. We have marked the classic RGB features with a large black spot, i.e. the result for the set of features $\{1, 2, 3\}$. Several conclusions can be drawn from them:

- The standard RGB features yield rather poor results in all respects. Most of the other combinations outperform this one.
- The accuracy and the F-measure are strongly correlated (Fig. 5), and they can be regarded as global performance measures. On the other hand, there are tradeoffs among false positives versus false negatives and precision versus recall (Figs. 1–4), as mentioned before.

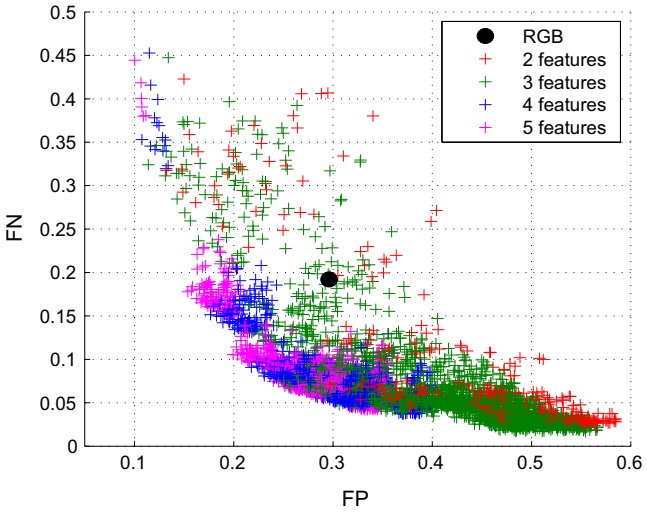



**Fig. 1.** False negatives versus false positives for all tested configurations.

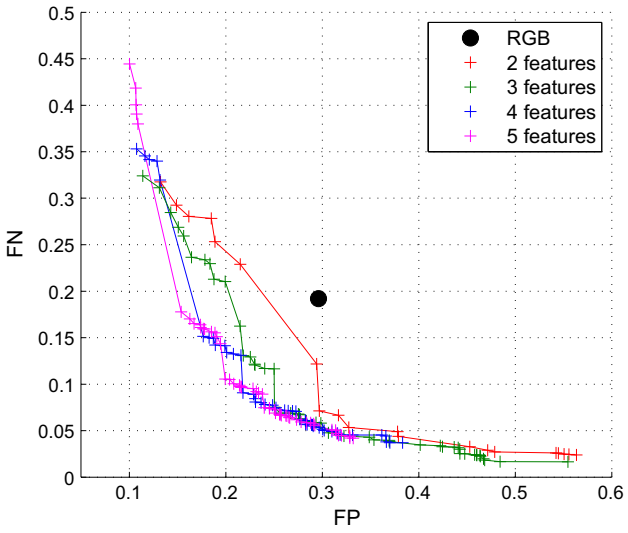



**Fig. 2.** False negatives versus false positives for Pareto front configurations.

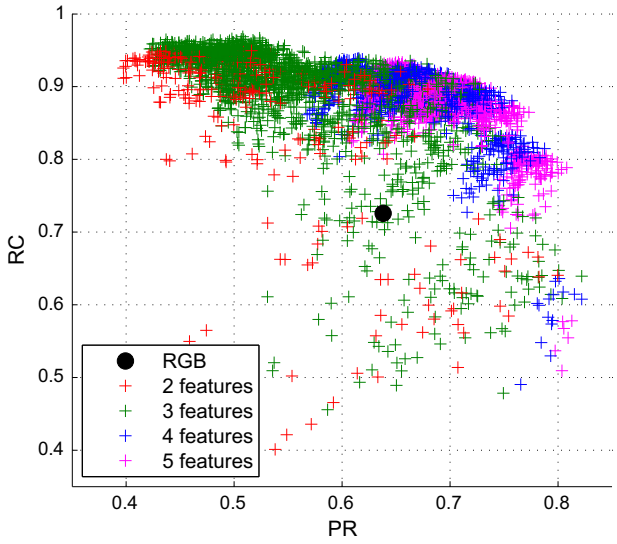



**Fig. 3.** Precision versus recall for all tested configurations.

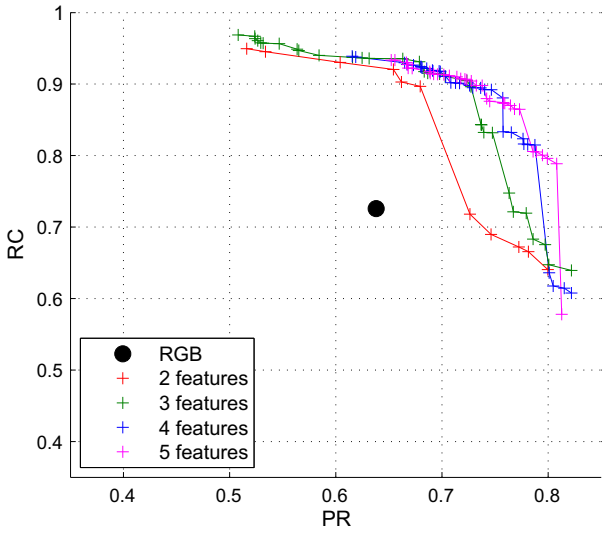



**Fig. 4.** Precision versus recall for Pareto front configurations.

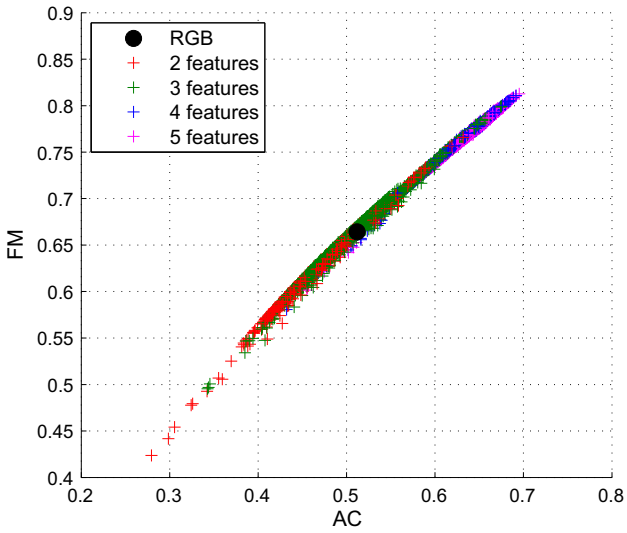



**Fig. 5.** Accuracy versus F-measure for all tested configurations.

- In general the performance improves as we add more features, but the improvement is progressively smaller (see Figs. 2, 4 and 6). Hence, we reach a point where it is not worthwhile to increase the computational complexity to gain little performance.
- For the false positives versus false negatives and precision versus recall plots (Figs. 1–4), the majority of the configurations are close to the Pareto fronts, with a minority of clearly bad ones. On the other hand, the accuracy versus F-measure plots are more discriminant (Figs. 5 and 6).

We have chosen 10 configurations which yield outstanding results, from those which belong to the Pareto fronts and those which give high values of *AC* and *FM*. They are listed on Table 2, along with their performance measures. From this table we can extract more useful information:

- All the configurations contain a majority of median filtered features (features 17 to 22). This means that median filtering is particularly effective at removing background noise.

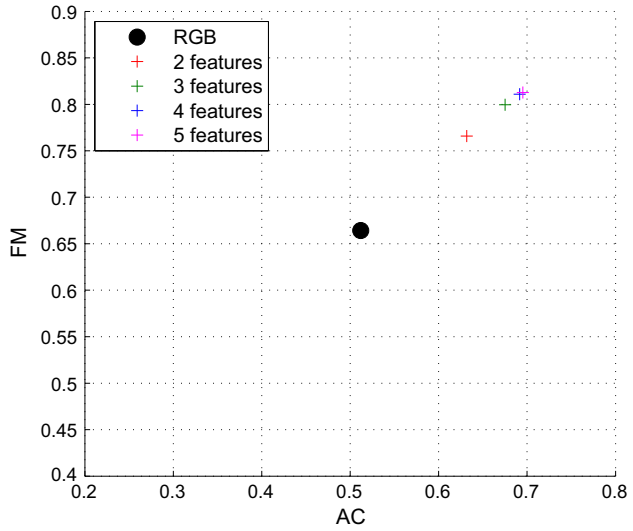



**Fig. 6.** Accuracy versus F-measure for Pareto front configurations.

**Table 2**
Some good performing configurations, sorted in lexicographical order of features. The best result for each criterion is marked in **bold**.

| Features | *FP* | *FN* | *PR* | *RC* | *AC* | *FM* |
|---|---|---|---|---|---|---|
| 3,20,21,22 | **0.1767** | 0.1515 | **0.7877** | 0.8149 | 0.6718 | 0.7941 |
| 4,19,20,21 | 0.2287 | 0.0890 | 0.7464 | 0.8819 | 0.6822 | 0.8047 |
| 4,19,20,21,22 | 0.2039 | 0.1050 | 0.7683 | 0.8652 | 0.6911 | 0.8102 |
| 5,6,19,20,22 | 0.2146 | 0.0975 | 0.7592 | 0.8732 | 0.6879 | 0.8085 |
| 5,19,20,22 | 0.2307 | 0.0808 | 0.7465 | 0.8921 | 0.6885 | 0.8087 |
| 5,19,21,22 | 0.2305 | 0.0845 | 0.7450 | 0.8866 | 0.6850 | 0.8056 |
| 5,19,20,21,22 | 0.1993 | 0.1054 | 0.7730 | 0.8648 | **0.6951** | **0.8126** |
| 6,19,20,21,22 | 0.2081 | 0.1005 | 0.7646 | 0.8701 | 0.6914 | 0.8104 |
| 19,20,22 | 0.2559 | **0.0690** | 0.7235 | **0.9048** | 0.6750 | 0.7996 |
| 19,20,21,22 | 0.2176 | 0.0907 | 0.7573 | 0.8807 | 0.6915 | 0.8108 |

- Among the median filtered features, those which appear the most often are features 20 to 22, i.e. the median filtered, normalized RGB channels. So, the resilience against illumination changes of normalization also improves the discrimination between background and foreground.
- Haar-like filters, gradient filters and color textures do not appear. Hence, they are non optimal for probabilistic mixture background modeling.

Now that we have studied the comparative performance of the 24 considered features, we are ready to compare our method with state-of-the-art proposals, which we do on Section 5.

## 5. Experimental results

In this section we are going to show the performance of our proposal against different state-of-the-art video segmentation methods.[3] For this study, we have tested several video sequences with various situations and a large number of configurations for each method. First of all, the competing methods and the tested sequences are described (Sections 5.1 and 5.2, respectively). Then the effect of using full covariance matrices in our proposal is assessed (Section 5.3). Section 5.4 is devoted to discuss the parameter choices for the tested approaches. Qualitative and quantitative results over standard benchmark videos are given in Sections 5.5 and 5.6, respectively. Finally two additional sets of experiments are reported, namely the Receiver Operating Characteristic (ROC) curves corresponding to different sets of features for our proposal (Section 5.7); and a comparison of the tested approaches for videos recorded under difficult conditions (Section 5.8).

### 5.1. Methods

Our method analyzes the scene pixel by pixel, so we have chosen five reference methods that are of this kind, i.e. the so-called pixel-level methods. The oldest we tested is Pfinder, proposed in [45], that we call WrenGA. It features a single Gaussian. We also analyze three Mixture of Gaussians approaches: GrimsonGMM, GMMV1 and GMMV2. GrimsonGMM is described in [40], while GMMV1 and GMMV2 are developed in [18,49] respectively. To complete the study we tested a fuzzy approach to a self-organizing background subtraction (SOBS) algorithm [26] named FASOM. In order to test these methods we have considered the implementation from version 1.3.0 of the BGS library, accessible from its website.[4] This library is written in C++ language and uses the free OpenCV Computer Vision library.[5] We must also highlight that the GMMV1 and GMMV2 are the current standard methods for the OpenCV library.

To prove our proposed method we have used an implementation in Matlab, in which we use MEX files for those more time-demanding parts and Matlab scripts for the rest.

In our aim to make a comparison as fair as possible, it has not been used any post-processing in any of the methods tested. All the experiments reported on this paper have been carried out on a 64-bit Personal Computer with a quad core 3.10 GHz CPU, 6 GB RAM and standard hardware. The implementation of our method does not use any GPU resources, so it does not require any specific graphics hardware.

### 5.2. Sequences

In video surveillance it is usual to find problematic situations as abrupt lighting changes, static or moving backgrounds, and foreground object's pixel features which may be subsumed by the modeled background. For this reason we have selected six standard videos with outdoor and indoor video sequences which exemplify these situations (Sections 5.5, 5.6, 5.7). We have also tested our proposal over four additional videos showing especially challenging situations (Section 5.8).

First of all the standard videos are described. Three videos have been taken from the dataset developed by Li et al. [21] available in his website[6]: an indoor video named moving curtain (MR) and an outdoor video named water surface (WS), whose main characteristic is that the background is constantly moving. Another test video is lobby (LB) whose peculiarity is the sudden changes of illumination.

A video featuring a 3D object moving through a real background scene has been added to the benchmarks. This kind of videos has the advantage that getting the ground truth is simple because, as the 3D object has been added by computer, we easily know which pixels belong to the 3D object in the sequence. The video named Video2 (V2) is an example of this and was created by the International Conference VSSN'06, on the occasion of an algorithm competition in Foreground/Background Segmentation.[7]

We have also selected other videos used in the literature, such as a video from CAVIAR dataset.[8] But only a sequence has been used

[3] The source code and several demos of our proposal are available at http://www.lcc.uma.es/%7Eezeqlr/backfeat/backfeat.html.

[4] https://github.com/andrewssobral/bgslibrary.
[5] http://opencv.org/.
[6] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html.
[7] http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC.
[8] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/.

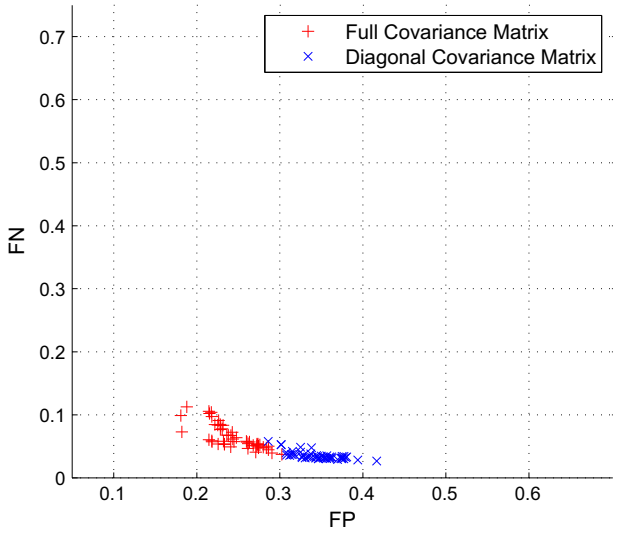



**Fig. 7.** False negatives versus false positives of each covariance matrix type for all tested configurations.

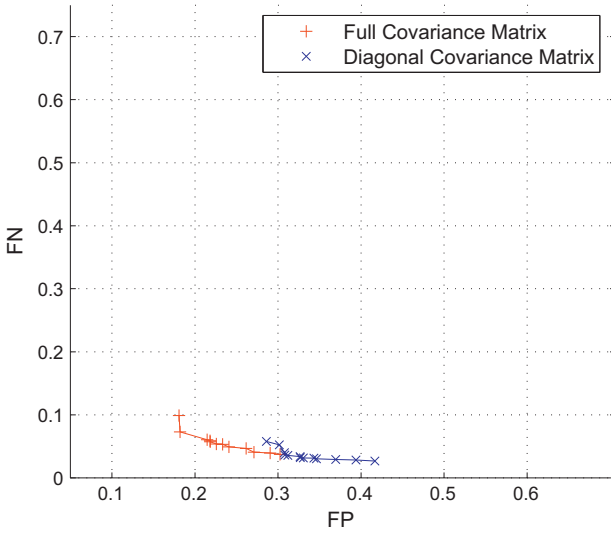



**Fig. 8.** False negatives versus false positives of each covariance matrix type for Pareto front configurations.

due to the difficulty with getting the ground truth: one shop corridor (OS). And we have completed the experiments with an outdoor sequence named level crossing (LC).[9]

The set of additional sequences has been chosen from two other well known datasets: ChangeDetection.net[10] and the BMC 2012 benchmark.[11] The first two sequences with adverse weather conditions named blizzard (BL) and snowFall (SF) are from ChangeDetection.net. The sequence named tunnel (TL) featuring heterogeneous lighting and the sequence named windy day (WD) showing a traffic camera with fast moving cars are available in BMC 2012 benchmark. As mentioned earlier, due to their special characteristics these videos are studied separately in Section 5.8.

### 5.3. Covariance matrix type study

As previously pointed out, our method uses the full covariance matrix. To assess its performance against a diagonal covariance matrix we have compared both alternatives. The results can be seen in Figs. 7–12 which show the average performance of the studied configurations (see Section 5.4) both the full covariance matrix and the diagonal covariance matrix. The use of a full covariance matrix achieved fewer FP without significantly increasing FN (Fig. 8). In terms of PR and RC, the full covariance matrix is able to improve PR without worsening RC (Fig. 10). Based on AC versus F-measure results, the best performance is achieved by using the full covariance matrix (Fig. 12). It is also worth noting that the worst configurations with the full matrix attain results similar to those of the best configurations that use a diagonal covariance matrix (Fig. 11). These results support the use of a full covariance matrix in our model (Section 3). However, it must be taken into account that any distribution can be approximated to the desired degree of accuracy with a large enough number of diagonal Gaussian mixture components.

### 5.4. Parameter selection

The set of test parameters are shown in Table 3. For the proposed algorithm we have tuned the step size $\varepsilon$ and the set of features $\mathcal{F}$, using 5 and 10 different values respectively, so we tested a total of 50 different configurations. For competitors we tested different values of all parameters that could be tuned according to the BGS implementation we have used. We tried the values recommended in the articles corresponding to these methods, we also included those configurations which are used by default in BGS implementations and we have made an additional selection of parameters that performed well but were not among those which have been mentioned before. We tested a large range of values for those parameters whose values affect most significantly the result. For example, we have considered more than 300 different configurations for the FASOM method, due to its large number of adjustable parameters.

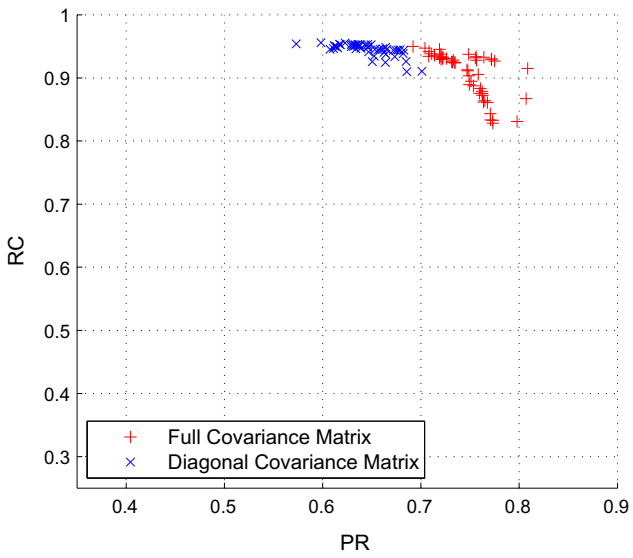



**Fig. 9.** Precision versus Recall of each covariance matrix type for all tested configurations.

The evaluation procedure has involved testing in each method all configurations discussed above for the six benchmark videos. The average performance of these configurations over the sequences is shown in Figs. 13–18, using the measures explained

[9] http://www.cs.cmu.edu/~yaser/.
[10] http://www.changedetection.net/.
[11] http://bmc.univ-bpclermont.fr/.

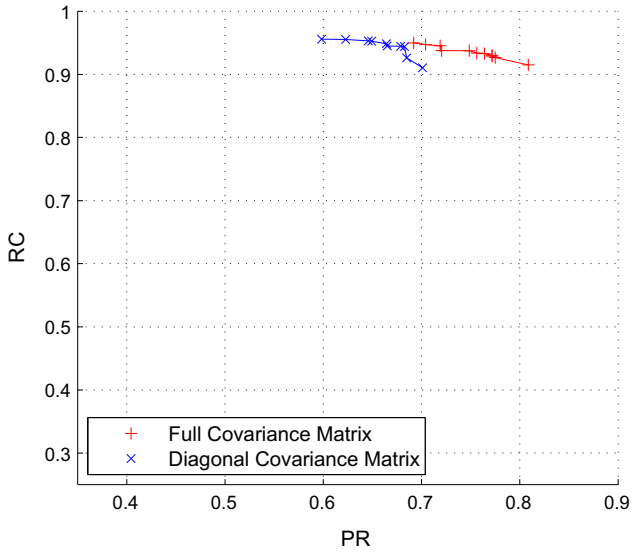



**Fig. 10.** Precision versus Recall of each covariance matrix type for Pareto front configurations.

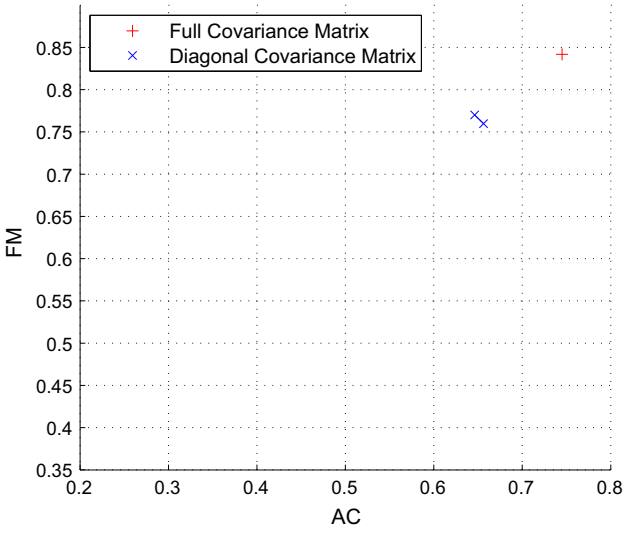



**Fig. 12.** Accuracy versus F-measure of each covariance matrix type for Pareto front configurations.

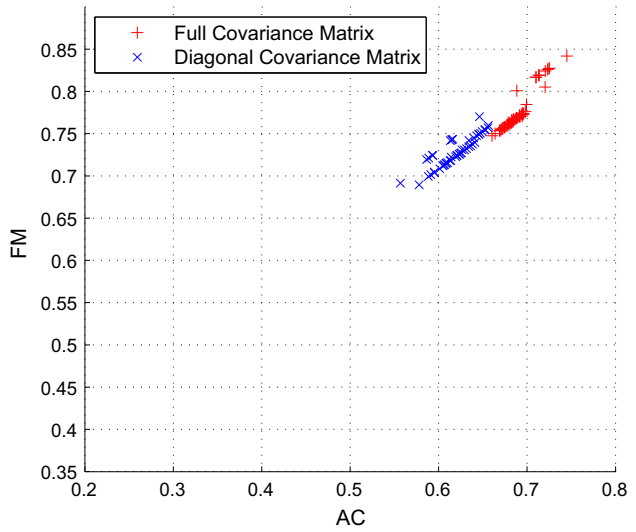



**Fig. 11.** Accuracy versus F-measure of each covariance matrix type for all tested configurations.

in Section 4.3. Each point in these plots indicates the average value obtained from a configuration for the six test videos.

### 5.5. Qualitative results

In Figs. 19–22 we show some qualitative results of the studied methods. Rows correspond to different frames of a benchmark video. The first and the second column are the original frame and the foreground mask (ground truth), respectively. Remaining columns are, in order, the results of applying methods MFBM, FASOM, GMMV2, GMMV1, GrimsonGMM and WrenGA to the frames. We use the setting that performed best according to F-measure for each of these sequences; see Table 6 for details.

There are some common problems that usually affect segmentation methods, for example *camouflage* and *cast shadow*. The camouflage problem seems to affect our method to a lesser degree than the other methods. This situation occurs when a method is not able to distinguish nearby pixels of the same color, some belonging to the background and others to the foreground, segmenting them wrongly. This can be seen in varying degrees in all the videos (Figs. 19–22), but particularly clearly in videos MR and LB (rows 2 and 3 of Figs. 20 and 21). However it must be said that there are particularly difficult situations where our method also fails to segment the foreground correctly due to the camouflage, just as the competitor methods do, e.g. video OS (row 4, Fig. 22).

Cast shadow problems occur when a method erroneously incorporates shadows into the foreground. Good examples of this problem can be found in the frames of MR (Fig. 20), some frames of LB (Fig. 21, second and third row) and in particular the OS sequence (rows 4 and 5, Fig. 22). In these cases we see that GMMV1 method virtually avoids these undesirable effects, but at the expense of a large number of false negatives, something that can be noticed in Fig. 14. WrenGA is also a robust method in this regard and it has many false negatives too, but it is not so remarkable as the case of GMMV1.

Another common problem is to confuse a moving background as part of the foreground. WS sequence (Fig. 19, rows 1 and 2) shows an exterior frame with this situation in which our method successfully finds the foreground, and at the same time it correctly segments the moving background. On the other hand, methods like GMMV2, WrenGA and less evidently FASOM, fail to segment the background. Furthermore, GrimsomGMM and GMMV1 do not have this problem, but they suffer from other troubles that make the result worse than that of our method.

As discussed before, the LB video (Fig. 21) is an interesting example of sudden illumination changes that occur in real applications. FASOM, WrenGA and MFBM show similar behavior with respect to a sudden change in lighting (Fig. 21, row 1). However our method is better in the frames after lighting changes (rows 2 and 3 of Fig. 21), showing little effect of camouflage and no occurrence of cast shadow. The same sequence is also interesting for other competing methods, as they react correctly to abrupt changes of illumination but, in the following frames, GMMV1 is unable to detect objects in the foreground, while GMMV2 and GrimsonGMM have important cast shadow problems.

### 5.6. Quantitative results

The previous section has allowed us to assess the performance of the different testing methods in a visual way. In order to show

**Table 3**
This table shows the test parameters, the combinations of them form the set of all tested configurations.

| Method | Parameters |
|---|---|
| WrenGA | Threshold, $\mathcal{N} = \{10, 12, 14, 16\}$<br>Alpha, $\alpha = \{0.0001, 0.002, 0.004, 0.006, 0.008, 0.01\}$<br>Learning frames, $\ell = \{20, 30, 40\}$ |
| GrimsonGMM | Threshold, $\mathcal{N} = \{8, 9, 10, 11, 12\}$<br>Learning rate,<br>$\alpha = \{0.0025, 0.005, 0.0075, 0.01, 0.0125, 0.015, 0.0175, 0.02\}$<br>Number of Gaussians in the mixture model, $\mathcal{K} = \{3, 4, 5\}$ |
| GMMV1 | Learning rate,<br>$\alpha = \{0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01\}$<br>History, $\mathcal{H} = \{100, 200\}$<br>Number of Gaussians components, $\mathcal{K} = \{4, 5, 6\}$<br>Threshold, $\mathcal{N} = \{0.6, 0.7, 0.8\}$ |
| GMMV2 | Learning rate, $\alpha = \{0.0001, 0.0005, 0.001, 0.005, 0.01\}$<br>History, $\mathcal{H} = \{400, 500\}$<br>Threshold, $\mathcal{N} = \{8, 10, 12, 14, 16, 18, 20\}$ |
| FASOM | Sensitivity, $s_1 = \{80, 90, 100\}$<br>Training sensitivity, $s_0 = \{200, 220, 240, 260\}$<br>Learning rate, $\alpha_1 = \{36, 38, 40\}$<br>Training learning rate, $\alpha_0 = \{240, 255, 270\}$<br>Training step, $\varepsilon = \{79, 81, 83\}$ |
| MFBM | Step size, $\varepsilon = \{0.0001, 0.0005, 0.001, 0.005, 0.01\}$<br>Sort of features shows in Table 2, $\mathcal{F}$ |

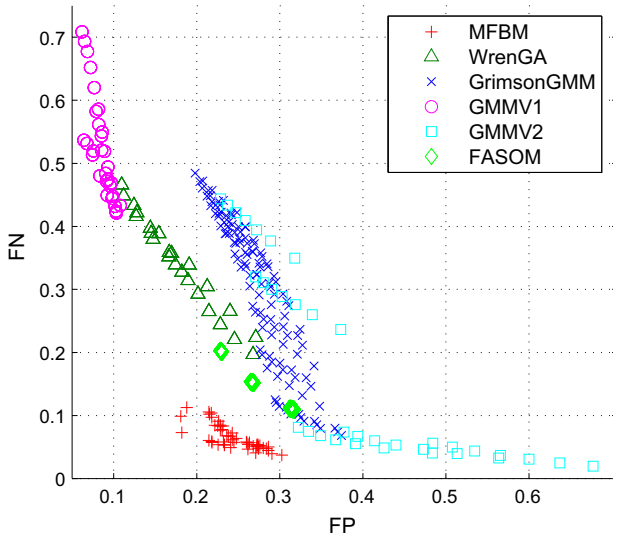



**Fig. 13.** False negatives versus false positives of each method for all tested configurations.

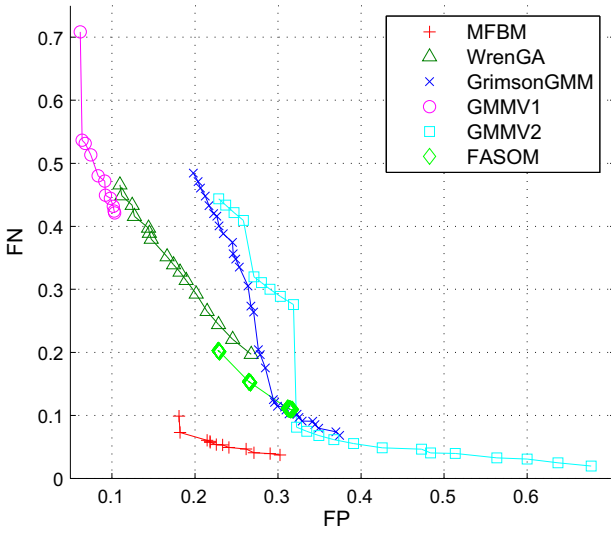



**Fig. 14.** False negatives versus false positives of each method for Pareto front configurations.

clearly the advantages of our proposal, we present Tables 4 and 5, where we list the mean and standard deviation calculated over the results obtained for the different configurations described in Section 5.4. As we see FASOM and GMMV2 are, among the competitors, the best performing methods behind our proposal, and they even surpass it in MR and LB sequences. However, in the case of MR the mean values of both methods are very close together. Also, it is worth noting the good average results obtained by WrenGA in video OS. To get a clearer picture, we can use Tables 6 and 7. They show the best configurations according to F-measure and to accuracy respectively; these values are calculated by averaging over all frames. So we can affirm that, considering the overall set of tested configurations, there is at least one MFBM configuration that is the best in the six benchmark sequences studied according to accuracy, and the same applies to F-measure except in the case of LB sequence, where it is slightly outperformed by GMMV2. These results are confirmed by the global data listed in Tables 4 and 5.

It must be pointed out that the evaluated frames have been those containing foreground objects. The reason for doing so is that the accuracy measure is reliable only when there are foreground objects in the frame. Please note that if no foreground pixels occur, then the most likely situation is $FP = 1, FN = 0, AC = 0$, and this outcome does not correctly assess the performance of a method. In other words, the measurement accuracy is based primarily on how well a method detects the foreground and not the background.

In Table 8 we can see which is FPS performance of each method for every sequence. Real time requirements are fulfilled if a method is over 15 fps. It can be observed that all the methods verify this condition for all the tested sequences. Our method is the slowest one due to the processing of more features, but there is room to

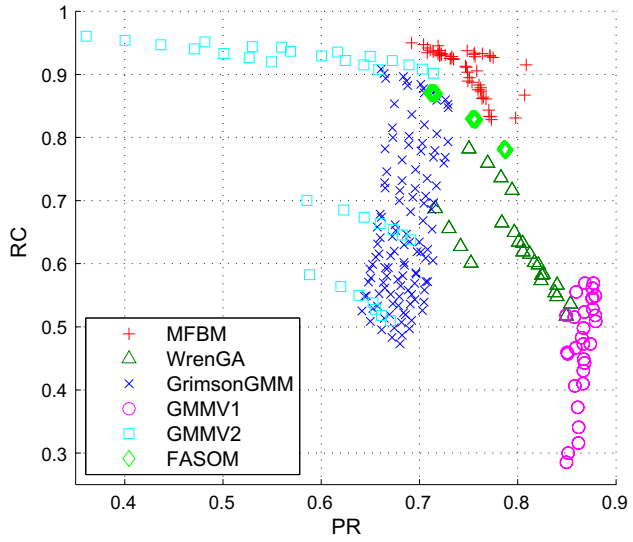



**Fig. 15.** Precision versus Recall of each method for all tested configurations.

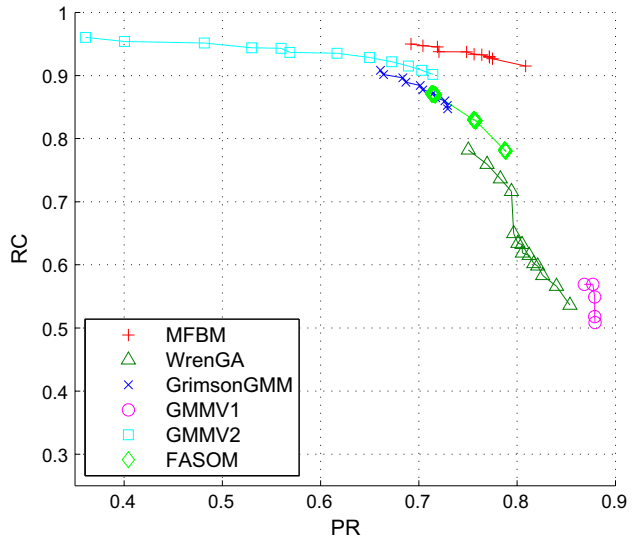



**Fig. 16.** Precision versus Recall of each method for Pareto front configurations.

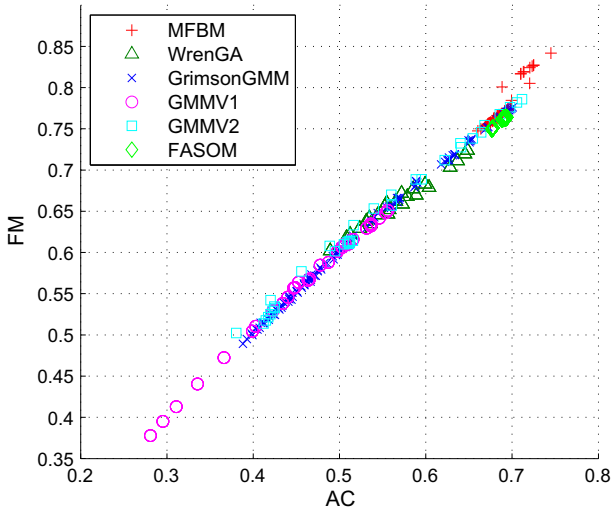



**Fig. 17.** Accuracy versus F-measure of each method for all tested configurations.

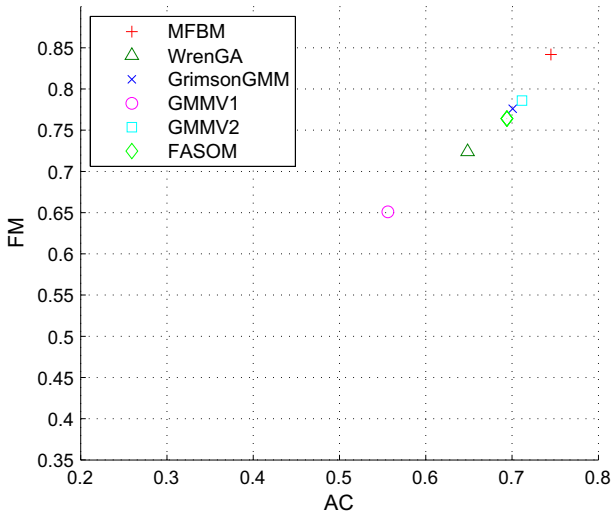



**Fig. 18.** Accuracy versus F-measure of each method for Pareto front configurations.

accelerate it, since our implementation does not use GPU hardware. Among the other methods, the FASOM is the worst of them in this respect because it uses more complex structures such as self-organizing maps.

### 5.7. ROC curves

In order to compare the different features it is useful to use Receiver Operating Characteristic (ROC) curves. The performance for varying values of the probability threshold $\rho$ (see Eq. (10)) is shown in terms of TPR (True Positive Rate) vs FPR (False Positive Rate). TPR is equivalent to Recall and FPR is defined as follows:

$$FPR = \frac{\mathrm{card}\left(\overline{A} \cap B\right)}{\mathrm{card}\left(\overline{A}\right)} \tag{70}$$

Fig. 23 shows the ROC curves of five of the best features, the closer to the point $(0,1)$, the better performance. As wee see feature set $\{3,20,21,22\}$is the best choice; it is even better than using five features. The other feature sets get a similar performance; as the number of used features increases, the performance gets slightly better.

### 5.8. Operation under difficult conditions

In this section a selection of videos recorded under difficult conditions are included in order to have a broader view of the performance of our proposal. For this purpose we use a set of additional videos which exemplify some of these situations, such as snow, fog, wind, poor lighting, fast moving objects and jitter camera.

These tests have been conducted using a single configuration for each method, namely those with better average performance in the six standard sequences studied previously (see Fig. 18). The only exception is FASOM in BL and SF where we use directly the results provided by the authors of the method. Table 9 details which parameters are used in each method. For BL and SF a prespecified value of $\sigma^2 = 10^{-4}$ had to be used instead of the quantization noise estimation procedure of Section 3.3. This was because the video compression works in a different way for these sequences with very little contrast.

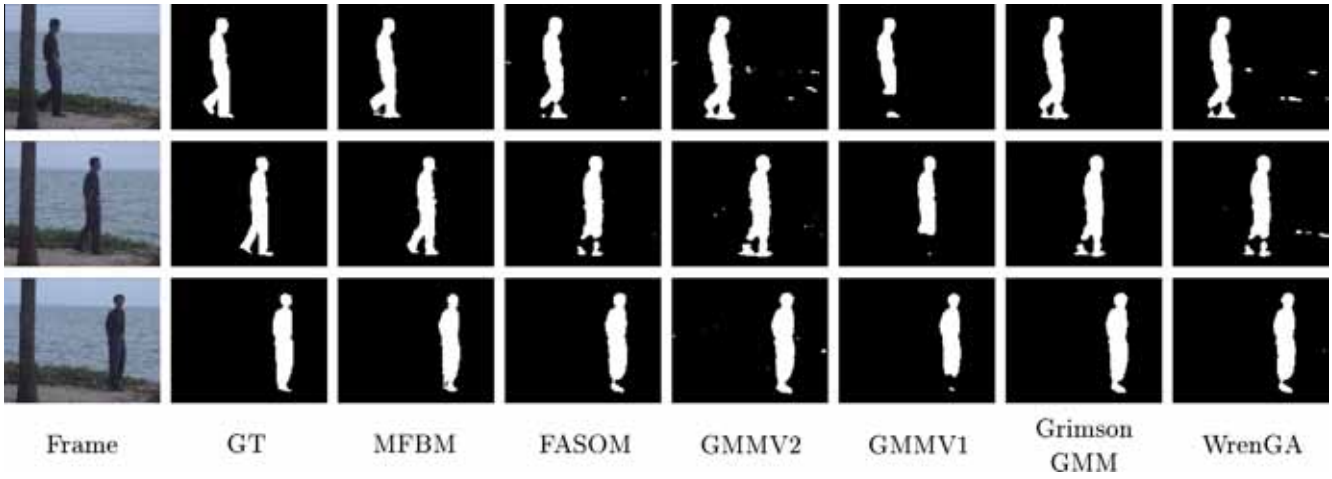



**Fig. 19.** Experimental results on scene with stationary foreground objects. These frames correspond to a sequence (WS) where the background of the picture moves continuously and a person appears walking from left to right. These three rows are the frames 1499, 1523 and 1605 respectively.

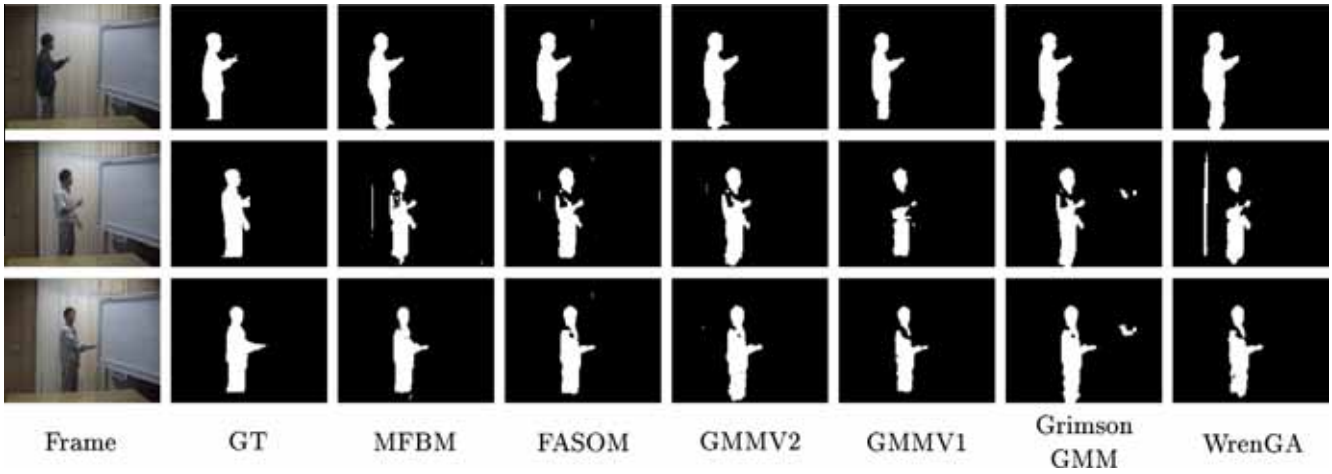



**Fig. 20.** Experimental results on sequence with a high variability in the background. In this video (MR) a person appears three times in the sequence with different clothes, and the curtains wave repeatedly due to the air flow. These three rows correspond to the frames 2774, 3226 and 3857 respectively.

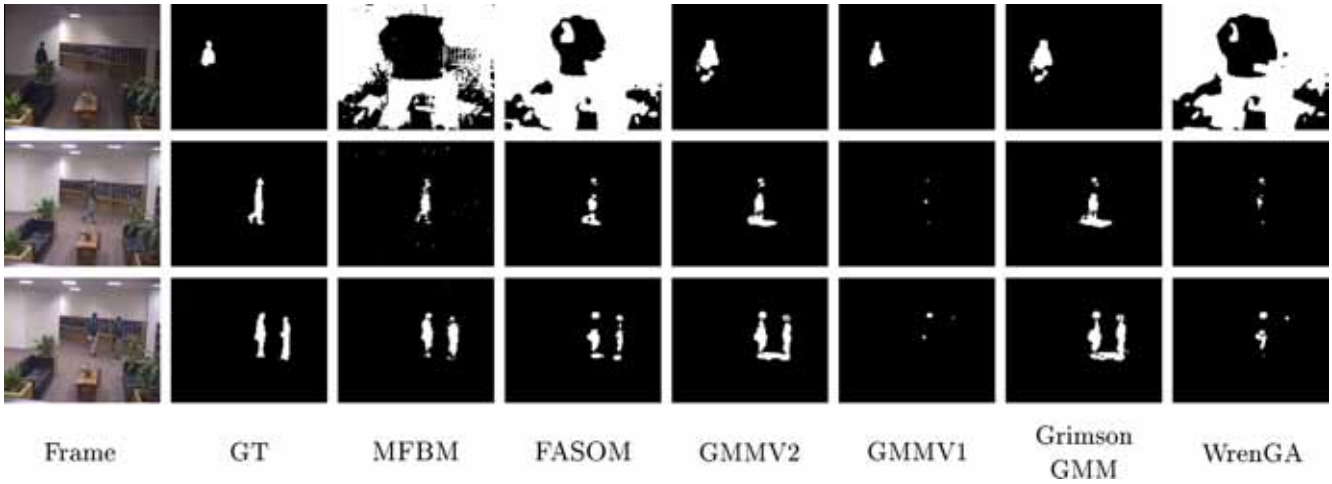



**Fig. 21.** Experimental results on lobby video (LB), which features abrupt lighting changes. These three rows correspond to the frames 1634, 2265 and 2457 respectively.

Fig. 24 illustrates the output of each method in the videos chosen from ChangeDetection.net. As we see on the first row this sequence is prone to camouflage problems, but our proposal detects all the objects in the scene with less FN than the

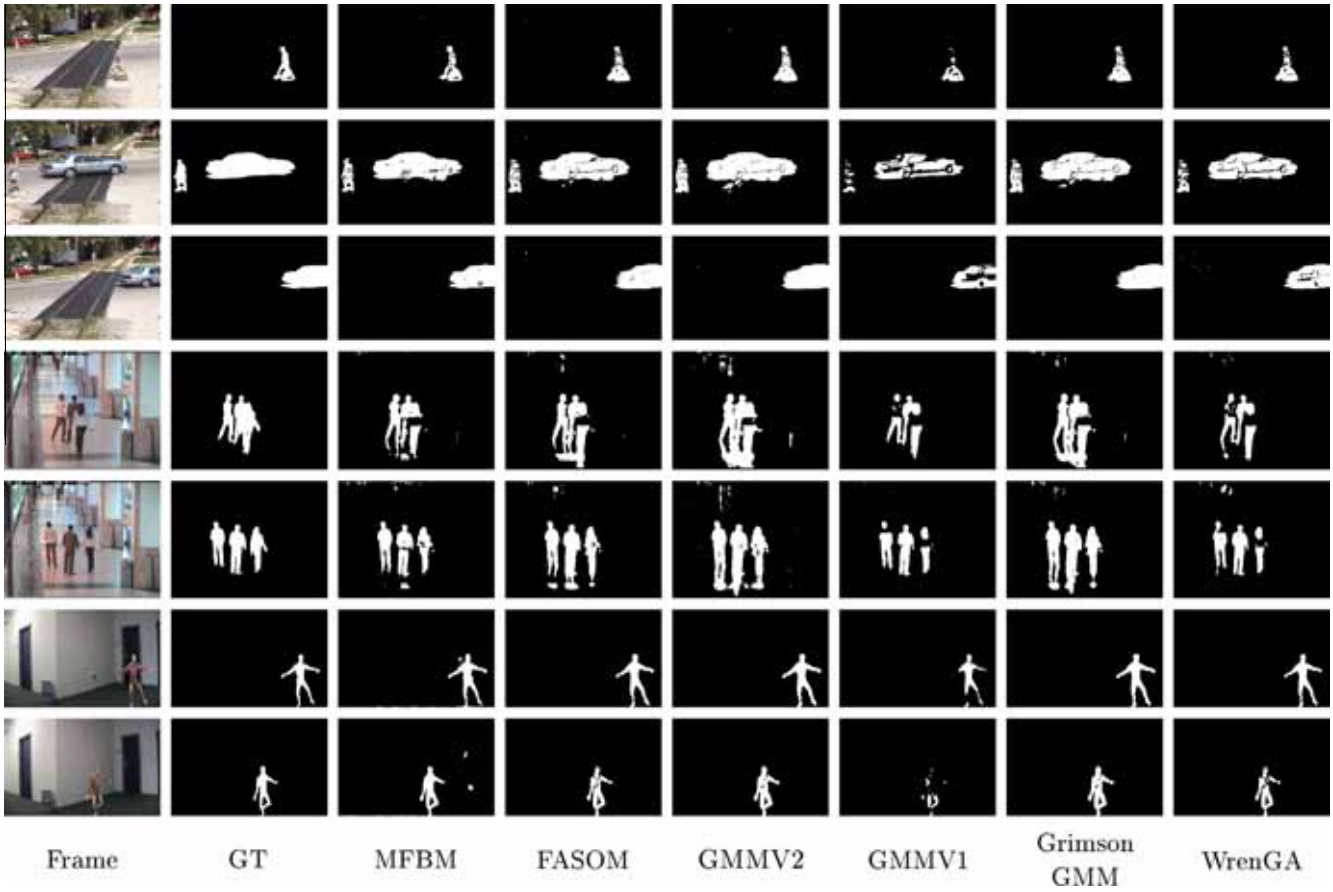



**Fig. 22.** Experimental results on several outdoor and indoor videos. The first three rows correspond to level crossing (LC), frames 324, 430 and 465 respectively. The fourth and fifth rows to one shop corridor (OS), frames 350 and 390. And the last sequence is Video2 (V2), frame 332 and 469.

**Table 4**
Quantitative results using the F-measure. Each column corresponds to a method and rows indicate the videos. Each cell shows the mean and standard deviation of F-measure over all tested configurations, best results are highlighted in **bold**.

| Sequence | *WrenGA* | *GrimsonGMM* | *GMMV1* | *GMMV2* | *FASOM* | *MFBM* |
|---|---|---|---|---|---|---|
| WS | 0.87 ± 0.03 | 0.67 ± 0.18 | 0.80 ± 0.12 | 0.67 ± 0.15 | 0.91 ± 0.00 | **0.95 ± 0.00** |
| MR | 0.68 ± 0.09 | 0.57 ± 0.09 | 0.58 ± 0.16 | 0.68 ± 0.15 | **0.87 ± 0.02** | 0.87 ± 0.02 |
| LC | 0.79 ± 0.04 | 0.86 ± 0.02 | 0.68 ± 0.06 | 0.80 ± 0.07 | 0.88 ± 0.01 | **0.92 ± 0.01** |
| V2 | 0.53 ± 0.04 | 0.55 ± 0.04 | 0.45 ± 0.05 | 0.57 ± 0.03 | 0.60 ± 0.01 | **0.84 ± 0.03** |
| LB | 0.29 ± 0.08 | 0.54 ± 0.08 | 0.18 ± 0.03 | **0.58 ± 0.11** | 0.54 ± 0.05 | 0.28 ± 0.20 |
| OS | **0.80 ± 0.03** | 0.54 ± 0.12 | 0.67 ± 0.16 | 0.53 ± 0.12 | 0.76 ± 0.02 | 0.79 ± 0.03 |

**Table 5**
Quantitative results using the accuracy. Each column corresponds to a method and rows indicate the videos. Each cell shows the mean and standard deviation of accuracy over all tested configurations, best results are highlighted in **bold**.

| Sequence | *WrenGA* | *GrimsonGMM* | *GMMV1* | *GMMV2* | *FASOM* | *MFBM* |
|---|---|---|---|---|---|---|
| WS | 0.77 ± 0.04 | 0.55 ± 0.20 | 0.68 ± 0.13 | 0.53 ± 0.16 | 0.83 ± 0.01 | **0.91 ± 0.01** |
| MR | 0.54 ± 0.09 | 0.44 ± 0.09 | 0.46 ± 0.15 | 0.54 ± 0.16 | **0.78 ± 0.03** | 0.77 ± 0.03 |
| LC | 0.66 ± 0.06 | 0.76 ± 0.03 | 0.53 ± 0.06 | 0.68 ± 0.08 | 0.78 ± 0.02 | **0.85 ± 0.01** |
| V2 | 0.55 ± 0.09 | 0.62 ± 0.08 | 0.43 ± 0.07 | 0.65 ± 0.08 | 0.72 ± 0.02 | **0.73 ± 0.04** |
| LB | 0.21 ± 0.06 | 0.40 ± 0.06 | 0.12 ± 0.03 | **0.44 ± 0.10** | 0.41 ± 0.05 | 0.22 ± 0.17 |
| OS | **0.67 ± 0.04** | 0.38 ± 0.11 | 0.53 ± 0.16 | 0.37 ± 0.11 | 0.61 ± 0.03 | 0.66 ± 0.05 |

competitors. The second row presents a situation where WrenGA and GMMV2 get good results but MFBM is better than WrenGA in terms of FN and better than GMMV2 in terms of FP. Rows 3 and 4 (SF sequence) show situations where it is snowing or it has snowed recently. This causes false positive alarms in GMMV2, GrimsonGMM, WrenGA and lesser extent in MFBM. However MFBM gets less false negatives than FASOM and GMMV1 which appear to be robust against the snow, so again our proposal is the best choice.

On the other hand Fig. 25 shows the results from the BMC benchmark. TL lighting causes several shadow cast problems. As we see MFBM is the best performing method in this respect. Only

**Table 6**
Best configurations according F-measure. First column denotes the sequence name, whereas the second column shows the method. Best F-Measures of each method are shown on the third column; this value is calculated by performing the average of all F-measures of the sequence. On the last column we have written the configuration corresponding to that result. Best results of each sequence are highlighted in **bold**.

| Sequence | Method | F-measure | Parameters |
|---|---|---|---|
| WS | WrenGA | 0.91 | $\mathcal{N} = 14, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.93 | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.87 | $\alpha = 0.003, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.91 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.91 | $s_1 = 90, s_0 = 260\ \alpha_1 = 40, \alpha_0 = 270, \varepsilon = 83$ |
| | MFBM | **0.96** | $\varepsilon = 0.005, \mathcal{F} = \{5, 6, 19, 20, 22\}$ |
| MR | WrenGA | 0.81 | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.78 | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.78 | $\alpha = 0.001, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.87 | $\alpha = 0.0005, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.89 | $s_1 = 100, s_0 = 240\ \alpha_1 = 40, \alpha_0 = 240, \varepsilon = 79$ |
| | MFBM | **0.90** | $\varepsilon = 0.005, \mathcal{F} = \{19, 20, 22\}$ |
| LC | WrenGA | 0.86 | $\mathcal{N} = 12, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.89 | $\mathcal{N} = 10, \alpha = 0.0025, \mathcal{K} = 4$ |
| | GMMV1 | 0.73 | $\alpha = 0.005, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.88 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.89 | $s_1 = 80, s_0 = 260\ \alpha_1 = 40, \alpha_0 = 240, \varepsilon = 79$ |
| | MFBM | **0.93** | $\varepsilon = 0.001, \mathcal{F} = \{3, 20, 21, 22\}$ |
| V2 | WrenGA | 0.60 | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.61 | $\mathcal{N} = 8, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.51 | $\alpha = 0.005, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.61 | $\alpha = 0.0005, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.61 | $s_1 = 80, s_0 = 200\ \alpha_1 = 36, \alpha_0 = 255, \varepsilon = 81$ |
| | MFBM | **0.88** | $\varepsilon = 0.005, \mathcal{F} = \{5, 19, 21, 22\}$ |
| LB | WrenGA | 0.42 | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.71 | $\mathcal{N} = 9, \alpha = 0.0025, \mathcal{K} = 5$ |
| | GMMV1 | 0.26 | $\alpha = 0.01, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | **0.74** | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.60 | $s_1 = 80, s_0 = 200\ \alpha_1 = 36, \alpha_0 = 240, \varepsilon = 83$ |
| | MFBM | 0.72 | $\varepsilon = 0.0001, \mathcal{F} = \{3, 20, 21, 22\}$ |
| OS | WrenGA | 0.83 | $\mathcal{N} = 16, \alpha = 0.002, \ell = 20$ |
| | GrimsonGMM | 0.76 | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.84 | $\alpha = 0.004, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.71 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.79 | $s_1 = 100, s_0 = 220\ \alpha_1 = 40, \alpha_0 = 240, \varepsilon = 79$ |
| | MFBM | **0.87** | $\varepsilon = 0.005, \mathcal{F} = \{3, 20, 21, 22\}$ |

GMMV1 is able to attain a similar performance in the first row but it fails in the second one. The WD sequence (third and fourth rows) is difficult due to several factors: the background is very dynamic due to wind and the movement of the clouds, the clouds cast shadows on the road and the camera shakes (jitter effects). These issues cause that in general terms all methods get a very large number of false positives. The only exception is GMMV1 but its problem is just the opposite, it makes too many false negatives. Thus our method offers an interesting alternative because it produces a small number of false negatives and less false positives than most competitors.

In order to get a more accurate picture of the performance of each method, Table 10 shows the performance of each method over the difficult condition sequences. The results of all the compared methods are poor with respect to those reported in Section 5. As mentioned before, this is due to the extreme conditions recorded in the studied sequences, which produce noise and compression artifacts. As seen our proposal is the best alternative in BL and SF sequences. MFBM is also the best alternative in TL but GMMV1 gets similar results. In the WD video GMMV1 is the best method and our proposal gets the third best results. Therefore our proposal is a good performing method in these sequences. GMMV1 performs well in the last two sequences but it provides very poor results in BL and SF, while our method is only behind in the last sequence but is the best in the other three.

Additionally we have included Tables 11 and 12. Table 11 shows the F-measure' achieved by our proposal over the whole BadWeather category using the configuration $\varepsilon = 0.005$, $\mathcal{F} = \{4, 19, 20, 21\}$ which optimizes the performance in terms of F-measure' (see Section 4.3 for the definition of F-measure'). This table also shows the results of three of the competitors that are published in the ChangeDetection.net website. Please note that we use F-measure' rather than the usual F-measure because this website uses F-measure'. As we see MFBM achieves the best average result for the entire category. Specifically our proposal is the best alternative in two of the four videos and it also obtains competitive results in the rest. On the other hand Table 12 compares our proposal using the configuration $\varepsilon = 0.005$, $\mathcal{F} = \{4, 19, 20, 21\}$ with other proposals published in the Special Section on Background Models Comparison of CVIU Journal (May 2014). As we see our proposal is the best method in TL and a competitive alternative in WD. The methods considered for this purpose are the following:

- SAM [7]: They propose a self-adaptative Gaussian mixture model which performs pixel level and frame level processes that dynamically adapt to illumination changes.
- VC [10]: This paper presents the so called Visual Cortex model which is a biologically-inspired proposal.
- 3dSOBS+ [27]: It is a biologically inspired method that uses a neural background model automatically generated by a self-organizing method.
- Spampinato [39]: They propose a method to model background and foreground based on color and texture features.

**Table 7**
Best configurations according accuracy. First column denotes the sequence name, whereas the second column shows the method. Best accuracy of each method are shown on the third column; this value is calculated by performing the average of all accuracy measures of the sequence. On the last column we have written the configuration corresponding to that result. Best results of each sequence are highlighted in **bold**.

| Sequence | Method | Accuracy | Parameters |
|---|---|---|---|
| WS | WrenGA | 0.84 | $\mathcal{N} = 14, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.87 | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.78 | $\alpha = 0.003, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.83 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.84 | $s_1 = 90, s_0 = 260\ \alpha_1 = 40, \alpha_0 = 270, \varepsilon = 83$ |
| | MFBM | **0.92** | $\varepsilon = 0.005, \mathcal{F} = \{5, 6, 19, 20, 22\}$ |
| MR | WrenGA | 0.69 | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.66 | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.65 | $\alpha = 0.001, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.77 | $\alpha = 0.0005, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.80 | $s_1 = 100, s_0 = 240\ \alpha_1 = 40, \alpha_0 = 240, \varepsilon = 79$ |
| | MFBM | **0.82** | $\varepsilon = 0.005, \mathcal{F} = \{19, 20, 22\}$ |
| LC | WrenGA | 0.76 | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.80 | $\mathcal{N} = 9, \alpha = 0.0025, \mathcal{K} = 5$ |
| | GMMV1 | 0.58 | $\alpha = 0.005, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.79 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.80 | $s_1 = 80, s_0 = 260\ \alpha_1 = 40, \alpha_0 = 240, \varepsilon = 79$ |
| | MFBM | **0.87** | $\varepsilon = 0.001, \mathcal{F} = \{3, 20, 21, 22\}$ |
| V2 | WrenGA | 0.73 | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| | GrimsonGMM | 0.74 | $\mathcal{N} = 8, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.52 | $\alpha = 0.005, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.74 | $\alpha = 0.0005, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.74 | $s_1 = 80, s_0 = 200\ \alpha_1 = 36, \alpha_0 = 255, \varepsilon = 81$ |
| | MFBM | **0.79** | $\varepsilon = 0.005, \mathcal{F} = \{5, 19, 21, 22\}$ |
| LB | WrenGA | 0.30 | $\mathcal{N} = 10, \alpha = 0.01, \ell = 20$ |
| | GrimsonGMM | 0.55 | $\mathcal{N} = 9, \alpha = 0.0025, \mathcal{K} = 5$ |
| | GMMV1 | 0.20 | $\alpha = 0.01, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.60 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.46 | $s_1 = 80, s_0 = 200\ \alpha_1 = 36, \alpha_0 = 240, \varepsilon = 83$ |
| | MFBM | **0.61** | $\varepsilon = 0.0001, \mathcal{F} = \{3, 20, 21, 22\}$ |
| OS | WrenGA | 0.71 | $\mathcal{N} = 16, \alpha = 0.002, \ell = 20$ |
| | GrimsonGMM | 0.62 | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| | GMMV1 | 0.72 | $\alpha = 0.004, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{B} = 0.6$ |
| | GMMV2 | 0.55 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| | FASOM | 0.65 | $s_1 = 100, s_0 = 220\ \alpha_1 = 40, \alpha_0 = 240, \varepsilon = 79$ |
| | MFBM | **0.77** | $\varepsilon = 0.005, \mathcal{F} = \{3, 20, 21, 22\}$ |

**Table 8**
Frames per second (FPS, higher is better) for each analyzed sequence. The second column displays the size of the frame. The bigger the frame size, the fewer FPS ratio. Each shown value corresponds to the best possible configuration according to the accuracy measurement. The best frame rate for each sequence is shown in *italic*.

| Sequence | Size | *WrenGA* | *GrimsonGMM* | *GMMV1* | *GMMV2* | *FASOM* | *MFBM* |
|---|---|---|---|---|---|---|---|
| WS | $160 \times 128$ | 1041.63 | 431.91 | 1637.14 | *3821.61* | 255.85 | 65.20 |
| MR | $160 \times 128$ | 1017.13 | 374.66 | 1655.48 | *3720.99* | 254.31 | 94.97 |
| LC | $360 \times 240$ | 257.65 | 96.84 | 416.57 | *1232.40* | 57.86 | 25.03 |
| V2 | $384 \times 240$ | 241.53 | 99.71 | 406.19 | *1154.81* | 54.18 | 20.48 |
| LB | $160 \times 128$ | 1039.14 | 300.62 | 1764.97 | *3301.33* | 260.87 | 77.90 |
| OS | $384 \times 288$ | 200.95 | 59.25 | 335.64 | *866.02* | 48.34 | 19.59 |

- SLDP and StSIC [47]: This paper defines the SLDP and StSIC models, which are statistical and local feature-based approaches.

## 6. Discussion

From the study of our proposal that we have carried out in the preceding sections, we can discuss some important points:

- Some features which have been reported to be adequate for kernel density estimation based foreground detection, such as Haar-like filters [14], are not well suited for our probabilistic mixture based approach. Haar-like filters are weak features, so one can use a large amount of them to build a strong classifier; but in our case we need few features with the best possible discriminative power.
- The detection performance of probabilistic mixture models increases as we add more relevant features, but the improvement diminishes until we reach a point where inserting more features is useless. From our experiments we have found that our proposal reaches that point at $D = 5$ features.
- Normalized color channels and median filtered features yield particularly good results. This comes from their resilience against illumination changes (for normalized features) and their ability to filter the background noise (for median filtered features). Moreover, the positive effects are maximized in normalized median filtered features.
- State-of-the-art background modeling approaches are outperformed by our method. It is possible that these competing methods could be improved by adapting them to use any number and kind of features, as our proposal does.

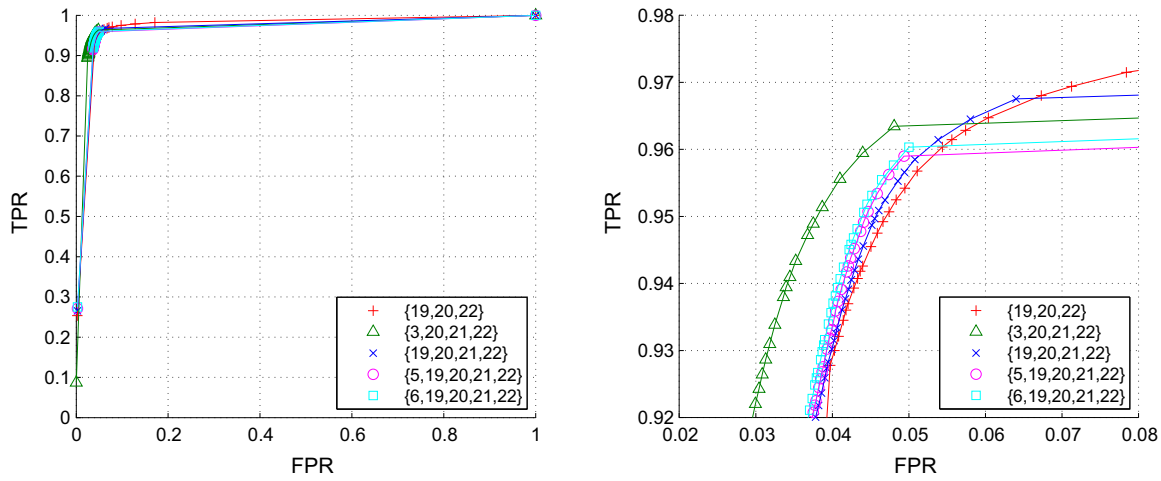



**Fig. 23.** ROC curves. The graph on the left shows the ROC curve of each feature set based on the $\rho$ value, the graph on the right shows these curves in detail. We set $\varepsilon = 0.0001$ to get these results.

**Table 9**
Parameters used in the difficult condition sequences. They correspond to the configurations shown in Fig. 6.

| Method | Parameters |
|---|---|
| WrenGA | $\mathcal{N} = 10, \alpha = 0.0001, \ell = 20$ |
| GrimsonGMM | $\mathcal{N} = 12, \alpha = 0.0025, \mathcal{K} = 3$ |
| GMMV1 | $\alpha = 0.004, \mathcal{H} = 100, \mathcal{K} = 4, \mathcal{N} = 0.6$ |
| GMMV2 | $\alpha = 0.001, \mathcal{H} = 400, \mathcal{N} = 20$ |
| FASOM | $s_1 = 90, s_0 = 240, \alpha_1 = 38, \alpha_0 = 240, \varepsilon = 81$ |
| MFBM | $\varepsilon = 0.0001, \mathcal{F} = \{3, 20, 21, 22\}$ |

- The search for relevant features for probabilistic mixture background models is an open field of research. It should be aimed to highly informative features which do not require much computation time.

Next the main novelties with respect to our previous proposal [24] are listed:

- The new model supports any number and kind of pixel features (Section 3.1).
- The uniform mixture component which models the foreground objects has been modified in order to suit features with any range of possible values, see (7), (8).
- A probability threshold $\rho$ has been introduced in this version to allow the user to tune the specificity and sensitivity of the foreground detection.
- The implementation of the method has been reworked completely because the implementation of [24] does not scale to more than 3 features. In particular, the Cholesky decomposition of the covariance matrix is used to compute the determinant of the covariance matrix and the Mahalanobis distance of the observed pixel feature vector (Section 3.6). This procedure is both fast and numerically stable because the inverse of the covariance matrix is never computed explicitly.

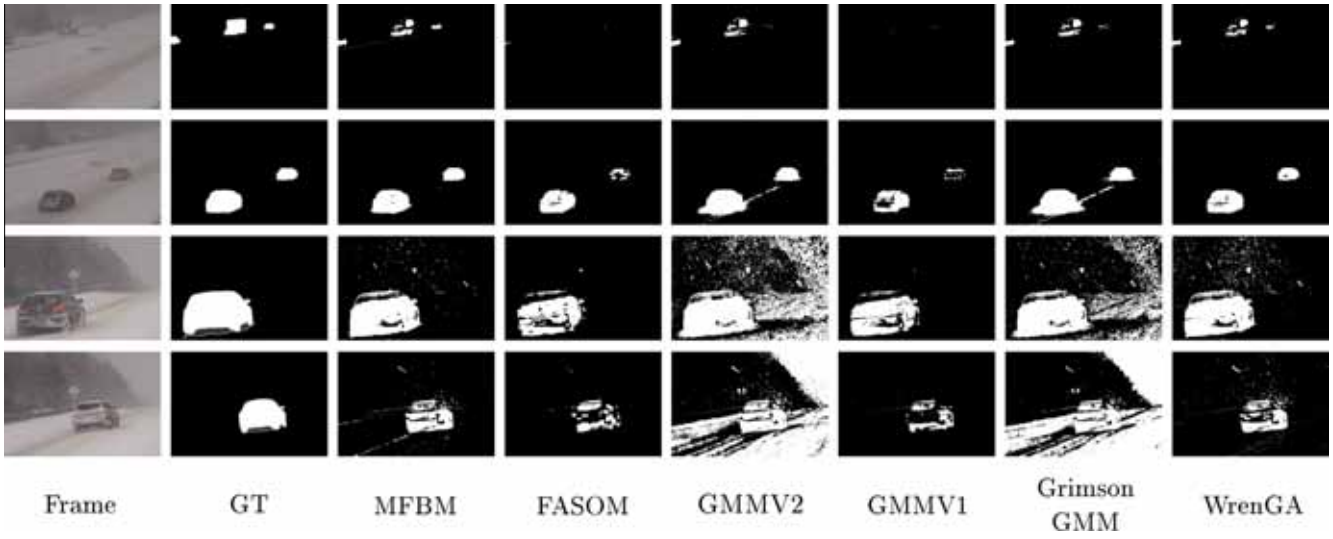



**Fig. 24.** Experimental results on ChangeDetection.net dataset. The first two rows correspond to blizzard (BL), frames 1500 and 3150 respectively. The third and fourth rows to snowFall (SF), frames 810 and 2785.

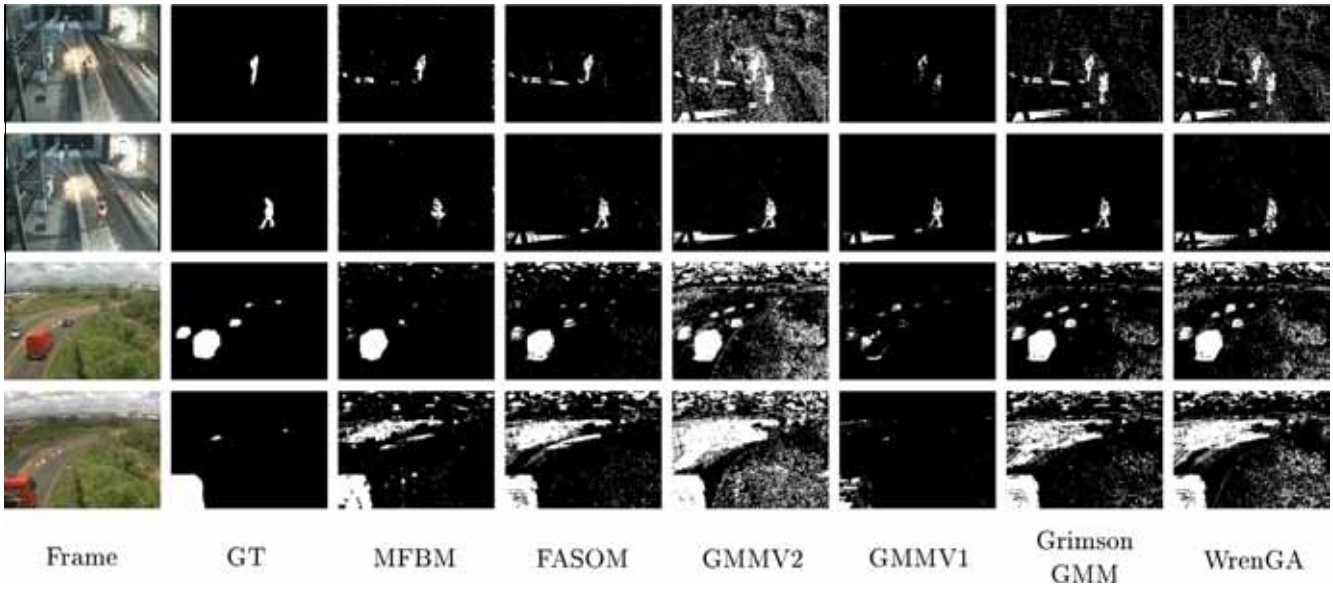



**Fig. 25.** Experimental results on BMC 2012 benchmark. The first two rows correspond to tunnel sequence (TL), frames 120 and 1363 respectively. The third and fourth rows to windy day (WD), frames 127 and 208.

**Table 10**
Best configurations according accuracy and F-measure for the difficult condition sequences. First column denotes the sequence name, whereas the second column shows the method. The accuracy and F-measure of each method is shown on the third and fourth columns, respectively. Best results for each sequence are highlighted in **bold**.

| Sequence | Method | Accuracy | F-measure |
|---|---|---|---|
| BL | WrenGA | 0.35 | 0.42 |
| | GrimsonGMM | 0.38 | 0.46 |
| | GMMV1 | 0.14 | 0.19 |
| | GMMV2 | 0.39 | 0.46 |
| | FASOM | 0.19 | 0.24 |
| | MFBM | **0.40** | **0.46** |
| SF | WrenGA | 0.15 | 0.19 |
| | GrimsonGMM | 0.14 | 0.18 |
| | GMMV1 | 0.08 | 0.11 |
| | GMMV2 | 0.13 | 0.16 |
| | FASOM | 0.11 | 0.16 |
| | MFBM | **0.16** | **0.20** |
| TL | WrenGA | 0.18 | 0.27 |
| | GrimsonGMM | 0.28 | 0.40 |
| | GMMV1 | 0.31 | 0.44 |
| | GMMV2 | 0.22 | 0.34 |
| | FASOM | 0.28 | 0.41 |
| | MFBM | **0.31** | **0.44** |
| WD | WrenGA | 0.09 | 0.15 |
| | GrimsonGMM | 0.16 | 0.26 |
| | GMMV1 | **0.22** | **0.35** |
| | GMMV2 | 0.09 | 0.16 |
| | FASOM | 0.13 | 0.21 |
| | MFBM | 0.15 | 0.23 |

**Table 11**
Results in terms of F-measure' for 'BadWeather' category. First column denotes the method name. From the second to the fifth column the F-measure' of each method is shown, and sixth column denotes the average performance for the whole 'BadWeather' category. Please note that WrenGA and GMMV1 results are not available in Changedetection.net website, hence the N/As. Best results for each sequence are highlighted in **bold**.

| Method | Blizzard | SnowFall | Skating | WetSnow | Average |
|---|---|---|---|---|---|
| WrenGA | N/A | N/A | N/A | N/A | N/A |
| GrimsonGMM | 0.74 | 0.73 | 0.88 | **0.61** | 0.74 |
| GMMV1 | N/A | N/A | N/A | N/A | N/A |
| GMMV2 | 0.76 | **0.76** | 0.86 | 0.58 | 0.74 |
| FASOM | 0.62 | 0.67 | 0.76 | 0.50 | 0.64 |
| MFBM | **0.83** | 0.75 | **0.89** | 0.57 | **0.76** |

**Table 12**
Results in terms of F-measure for the two BMC sequences of other additional proposals published in the Special Section on Background Models Comparison of CVIU Journal (May 2014). First column denotes the method name. Second and third columns show the performance obtained by each method in TL and WD sequences respectively. Best results for each sequence are highlighted in **bold**.

| Method | TL | WD |
|---|---|---|
| SAM | 0.754 | 0.608 |
| VC | 0.8 | 0.5 |
| 3dSOBS+ | 0.808 | 0.752 |
| Spampinato | 0.688 | 0.573 |
| SLDP | 0.785 | 0.827 |
| StSIC | 0.741 | **0.836** |
| MFBM | **0.810** | 0.611 |

## 7. Conclusions

A new model for the detection of foreground objects has been proposed. It is able to manage any number and kind of pixel features. We have also proposed a set of relevant features according to several properties. It has been found that normalized and median filtered pixel values outperform the standard RGB color channels. On the other hand, some features which have been reported to work well with kernel density estimation approaches are not suited for our method. The performance of our approach has been compared with state-of-the-art alternatives over a set of benchmark videos, with favorable results. We foresee that these alternative methods could be enhanced by modifying them to accept any number and kind of pixel features.

## Acknowledgments

This work is partially supported by the Ministry of Economy and Competitiveness of Spain under Grant TIN2011-24141, project name Detection of anomalous activities in video sequences by self-organizing neural systems. It is also partially supported by the Autonomous Government of Andalusia (Spain) under Projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Malaga.

## References

[1] C. Audet, J. Dennis Jr., S. Le Digabel, Trade-off studies in blackbox optimization, Optim. Methods Softw. 27 (4–5) (2012) 613–624.
[2] C. Benedek, T. Szirányi, Study on color space selection for detecting cast shadows in video surveillance, Int. J. Imaging Syst. Technol. 17 (3) (2007) 190–201.
[3] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, C. Rosenberger, Review and evaluation of commonly-implemented background subtraction algorithms, in: 19th International Conference on Pattern Recognition, December 2008, pp. 1–4.
[4] T. Bouwmans, F. El Baf, B. Vachon, Background modeling using mixture of gaussians for foreground detection – a survey, Recent Patents Comput. Sci. 1 (3) (2008) 219–237.
[5] F. Campbell-West, P. Miller, H. Wang, Independent moving object detection using a colour background model, in: IEEE International Conference on Video and Signal Based Surveillance, 2006 (AVSS '06), November 2006, pp. 31–31.
[6] P. Chen, X. Chen, B. Jin, X. Zhu, Online EM algorithm for background subtraction, Proc. Eng. 29 (2012) 164–169.
[7] Z. Chen, T. Ellis, A self-adaptive gaussian mixture model, Comput. Vision Image Understand. 122 (2014) 35–46.
[8] G. Cucchiara, P. Piccardi, Detecting moving objects, ghosts and shadows in video streams, IEEE Trans. Pattern Anal. Machine Intell. 25 (10) (2003) 1337–1342.
[9] A. Elgammal, R. Duraiswami, D. Harwood, L. Davis, Background and foreground modeling using nonparametric kernel density estimation for visual surveillance, Proc. IEEE 90 (7) (2002) 1151–1163.
[10] A.-L. Ellis, J. Ferryman, Biologically-inspired robust motion segmentation using mutual information, Comput. Vision Image Understand. 122 (2014) 47–64.
[11] X. Gao, T. Boult, F. Coetzee, V. Ramesh, Error analysis of background adaption, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2000, pp. 503–510.
[12] M. Greiffenhagen, V. Ramesh, H. Niemann, The systematic design and analysis cycle of a vision system: a case study in video surveillance, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001 (CVPR 2001), vol. 2, 2001, pp. II-704–II-711.
[13] T.S. Haines, T. Xiang, Background subtraction with Dirichlet processes, in: A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid (Eds.), Computer Vision ECCV 2012, Lecture Notes in Computer Science, vol. 7575, Springer, Berlin, Heidelberg, 2012, pp. 99–113.
[14] B. Han, L.S. Davis, Density-based multifeature background subtraction with support vector machine, IEEE Trans. Pattern Anal. Machine Intell. 34 (5) (2012) 1017–1023.
[15] M. Hedayati, W.M.D.W. Zaki, A. Hussain, A qualitative and quantitative comparison of real-time background subtraction algorithms for video surveillance applications, J. Comput. Inform. Syst. 8 (2) (2012) 493–505.
[16] M. Heikkila, M. Pietikainen, A texture-based method for modeling the background and detecting moving objects, IEEE Trans. Pattern Anal. Machine Intell. 28 (4) (2006) 657–662.
[17] P. KaewTrakulPong, R. Bowden, A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes, Image Vision Comput. 21 (9) (2003) 913–929.
[18] P. KaewTraKulPong, R. Rowden, An improved adaptive background mixture model for real-time tracking with shadow detection, in: Proceedings of the Second European Workshop on Advanced Video Based Surveillance Systems, 2001, 149–158.
[19] K. Kim, T.H. Chalidabhongse, D. Harwood, L. Davis, Real-time foreground-background segmentation using codebook model, Real-Time Imaging 11 (3) (2005) 172–185.
[20] H.J. Kushner, G.G. Yin, Stochastic Approximation and Recursive Algorithms and Applications, Springer-Verlag, New York, NY, USA, 2003.
[21] L. Li, W. Huang, I.Y.-H. Gu, Q. Tian, Statistical modeling of complex backgrounds for foreground object detection, IEEE Trans. Image Process. 13 (11) (2004) 1459–1472.
[22] X. Li, G. Du, BSTBGA: a hybrid genetic algorithm for constrained multi-objective optimization problems, Comput. Oper. Res. 40 (1) (2013) 282–302.
[23] M. López-Ibáñez, T. Stützle, An experimental analysis of design choices of multi-objective ant colony optimization algorithms, Swarm Intell. 6 (3) (2012) 207–232.
[24] E. López-Rubio, R.M. Luque-Baena, Stochastic approximation for background modelling, Comput. Vision Image Understand. 115 (6) (2011) 735–749.
[25] E. López-Rubio, R.M. Luque-Baena, E. Domínguez, Foreground detection in video sequences with probabilistic self-organizing maps, Int. J. Neural Syst. 21 (3) (2011) 225–246.
[26] L. Maddalena, A. Petrosino, A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection, Neural Comput. Appl. 19 (2) (2010) 179–186.
[27] L. Maddalena, A. Petrosino, The 3dsobs+ algorithm for moving object detection, Comput. Vision Image Understand. 122 (2014) 65–73.
[28] J.A. Marchant, C.M. Onyango, Shadow-invariant classification for scenes illuminated by daylight, J. Opt. Soc. Am. 17 (11) (2000) 1952–1961.
[29] N. McFarlane, C. Schonfield, Segmentation and tracking of piglets in images, Machine Vision Appl. 8 (3) (1995) 187–193.
[30] S. Mukherjee, K. Das, An adaptive GMM approach to background subtraction for application in real time surveillance, Int. J. Res. Eng. Technol. 2 (1) (2013) 25–29.
[31] N. Oliver, B. Rosario, A. Pentland, A Bayesian computer vision system for modeling human interactions, IEEE Trans. Pattern Anal. Machine Intell. 22 (8) (2000) 831–843.
[32] T. Parag, A. Elgammal, A. Mittal, A framework for feature selection for background subtraction, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2006, pp. 1916–1923.
[33] D. Parks, S. Fels, Evaluation of background subtraction algorithms with post-processing, in: IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance, AVSS, 2008, pp. 192–199.
[34] M. Piccardi, Background subtraction techniques: a review, in: IEEE Intl. Conf. on Systems, Man and Cybernetics, 2004, pp. 3099–3104.
[35] J. Pilet, C. Strecha, P. Fua, Making background subtraction robust to sudden illumination changes, in: Proceedings of the European Conference on Computer Vision, 2008.
[36] V. Saligrama, J. Konrad, P.-M. Jodoin, Video anomaly identification, IEEE Signal Process. Mag. 27 (5) (2010) 18–33.
[37] O. Schütze, X. Esquivel, A. Lara, C. Coello, Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 16 (4) (2012) 504–522.
[38] J. Shen, W. Yang, Z. Lu, Q. Liao, Information integration for accurate foreground segmentation in complex scenes, IET Image Process. 6 (5) (2012) 596–605.
[39] C. Spampinato, S. Palazzo, I. Kavasidis, A texton-based kernel density estimation approach for background modeling under extreme conditions, Comput. Vision Image Understand. 122 (2014) 74–83.
[40] C. Stauffer, W. Grimson, Adaptive background mixture models for real-time tracking, in: Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition, 1999, pp. 246–252.
[41] C. Stauffer, W. Grimson, Learning patterns of activity using real-time tracking, IEEE Trans. Pattern Anal. Machine Intell. 22 (8) (2000) 747–757.
[42] Y. Tian, Y. Wang, Z. Hu, T. Huang, Selective eigenbackground for background modeling and subtraction in crowded scenes, IEEE Trans. Circ. Syst. Video Technol. 23 (11) (2013) 1849–1864.
[43] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2001, pp. 511–518.
[44] L. Vosters, C. Shan, T. Gritti, Real-time robust background subtraction under rapidly changing illumination conditions, Image Vision Comput. 30 (12) (2012) 1004–1015.
[45] C. Wren, A. Azarbayejani, T. Darrell, A. Pentl, Pfinder: real-time tracking of the human body, IEEE Trans. Pattern Anal. Machine Intell. 19 (7) (1997) 780–785.
[46] Z. Xu, I.-H. Gu, P. Shi, Recursive error-compensated dynamic eigenbackground learning and adaptive background subtraction in video, Opt. Eng. 47 (5) (2008).

[47] S. Yoshinaga, A. Shimada, H. Nagahara, R. ichiro Taniguchi, Object detection based on spatiotemporal background models, Comput. Vision Image Understand. 122 (2014) 84–91.
[48] J. Zhong, S. Sclaroff, Segmenting foreground objects from a dynamic textured background via a robust Kalman filter, in: Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 1, October 2003, pp. 44–50.
[49] Z. Zivkovic, Improved adaptive gaussian mixture model for background subtraction, in: Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04), vol. 2, IEEE Computer Society, Washington, DC, USA, 2004, pp. 28–31.
[50] Z. Zivkovic, F. Van der Heijden, Recursive unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Machine Intell. 26 (5) (2004) 651–656.
[51] Z. Zivkovic, F. van der Heijden, Efficient adaptive density estimation per image pixel for the task of background subtraction, Pattern Recogn. Lett. 27 (7) (2006) 773–780.

Image and Vision Computing 37 (2015) 31–47

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Editor's Choice Article

# Local color transformation analysis for sudden illumination change detection ☆

Francisco Javier López-Rubio, Ezequiel López-Rubio *

*Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur, 35, 29071 Málaga, Spain*

ARTICLE INFO

*Article history:*
Received 1 October 2013
Received in revised form 26 May 2014
Accepted 30 March 2015
Available online 3 April 2015

*Keywords:*
Background modeling
Foreground detection
Illumination invariance
Color transformation

ABSTRACT

Sudden illumination changes are a fundamental problem in background modeling applications. Most strategies to solve it are based on determining the particular form of the color transformation which the pixels undergo when an illumination change occurs. Here we present an approach which does not assume any specific form of the color transformation. It is based on a quantitative assessment of the smoothness of the local color transformation from one frame to the background model. In addition to this, an assessment of the obtained illumination states of the pixels is carried out with the help of fuzzy logic. Experimental results are presented, which demonstrate the performance of our approach in a range of situations.



## 1. Introduction

The segmentation of the foreground objects in a scene is a fundamental task which lays at the foundation of many computer vision systems. Most approaches to this task create a model of the background that is updated progressively as time passes. Consequently sudden illumination changes are challenging problems, since the appearance of the background no longer matches the past observations. This issue has been studied for many years [6,9,10,20] due to its relevance to applications. Practical computer vision systems are placed in environments where the illumination conditions vary through time. For example, indoor scenes frequently exhibit light switching, while passing clouds affect outdoor cameras.

There are different ways to address this problem. On one side there are pixel-level algorithms that work by analyzing the scene on a pixel by pixel basis, so that an independent decision is made for each pixel. On the other hand, there are other approaches which work at a higher level, namely block-level (where the decision for one pixel depends on the information from several nearby pixels) or frame-level (where all the pixels of the frame might be taken into account to decide whether a pixel belongs to the foreground). It is often the case that methods use multiple levels simultaneously to analyze the video. A representative case of this approach is the Wallflower system 28, which uses the three levels already mentioned. One of the downsides attributed to this method is that it uses a non flexible criterion in real situations because it selects a set of background scene models representing different situations and each frame has to be assigned to the model which produces the fewest foreground pixels. This implies that each possible situation must be predicted in advance, and that a representative background model of each situation must be found. Hence, the approach is less adequate for scenes where unpredictable events occur which affect the background, such as a left object or a parked car which starts moving.

There are other methods using region-level analysis as in the case of [22]. This approach analyzes the image at block and pixel levels. The main idea is that each pixel belongs to several overlapping blocks, so that it is determined whether it belongs to the background or not depending on how it has been classified in each of these blocks.

A more recent algorithm is [14], which uses pixel-level and image-level elements. The proposal consists in using a two-layer architecture based on a Gaussian Mixture Model to represent the background. Then the result is optimized using a Markov random field decision framework.

On the other hand, there are many algorithms that work at the pixel level. For example [27] is based on applying a homomorphic filter, while [12] performed an analysis with stereo vision and employs a disparity model created offline to mitigate the consumption of extra CPU time required for this type of processing. Also, [8] uses discriminative texture features to capture background statistics, by means of texture operators named local binary patterns (LBP). Finally, [6] presents an adaptive algorithm that uses multiple feature subspaces and Principal Component Analysis to capture and learn different lighting conditions.

Our aim here is to develop an illumination change detection system which works along an existing foreground detection algorithm. A

☆ Editor's Choice Articles are invited and handled by a select rotating 12 member Editorial Board committee. This paper has been recommended for acceptance by Dr. Bodo Rosenhahn.

* Corresponding author.
*E-mail addresses:* xavierprof@hotmail.com (F.J. López-Rubio), ezeqlr@lcc.uma.es (E. López-Rubio).
*URL:* http://www.lcc.uma.es/%7Eezeqlr/index-en.html (E. López-Rubio).

http://dx.doi.org/10.1016/j.imavis.2015.03.001
0262-8856/

previous example of an add-on illumination change detection algorithm can be found in [32]. Unlike our proposal, they assume that the order of pixel values is preserved in local neighborhoods when illumination changes occur, and they root their proposal on the analysis of physical properties, namely the radiance. In contrast to this, we are not interested in the particular form of the color transformation, but in its smoothness properties. This way, we consider that any color transformation which is not smooth can not correspond to a lighting change, i.e. it is due to a foreground object. Hence, we are able to detect variations in illumination irrespective of the particular features of the color transformation at hand. Moreover, the procedure is largely independent from the baseline background model, so it can be used to improve a number of well known methods which are not specifically designed to work under illumination changes.

This paper is organized as follows. Section 2 presents the illumination change detection method. Section 3 presents some experimental results to demonstrate the ability of our approach to manage complex scenes. The main features and properties of our proposal are discussed in Section 4. Finally, Section 5 is devoted to conclusions.

## 2. The method

The illumination change management procedure that we present here has two parts. The first one classifies the pixels of the current frame according to its current state with respect to illumination changes (Subsection 2.1). The second part uses this information to decide which pixels must undergo a reset because an illumination change has rendered their background models outdated (Subsection 2.2).

### 2.1. Illumination state estimation

Here we must estimate the illumination state of each pixel of the current video frame. The illumination state of pixel $i$ is formed by three fuzzy variables *Rough*, *Difference*, and *Baseline*; their values (membership degrees) will be noted $\alpha_i, \beta_i, \gamma_i \in [0, 1]$, respectively. The interpretation of the variables is as follows:

- *Rough* indicates whether the transformation of the colors in the previous frame to the colors in the current frame is not smooth in the vicinity of pixel $i$. If $\alpha_i$ is high, then it is unlikely that an illumination change is happening, since illumination changes produce smooth changes in the colors of the background and the foreground objects.
- *Difference* indicates whether the color of pixel $i$ in the current frame is very different from that stored in the background model for pixel $i$. If $\beta_i$ is high, then either an illumination change is happening or a foreground object is present.
- *Baseline* indicates whether the baseline background model has detected a foreground object. Please note that $\gamma_i \in \{0, 1\}$ for background models that do not output a degree of confidence for the foreground detection.

Next we describe how to compute the fuzzy membership values $\alpha_i$ and $\beta_i$; please note that $\gamma_i$ is simply the output of the baseline background model.

Let us center our attention in a small neighborhood $W_i$ of pixel $i$. In our experiments we have considered square windows of size $5 \times 5$ pixels, which offer a good tradeoff between efficiency and accuracy. Now we define a vector field which represents the transformation of the colors of the neighborhood $W_i$ in the background model to the colors of $W_i$ in the current frame:

$$f_i : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \tag{1}$$

where tristimulus color values are assumed, but any color space could be used. Now, our task is to detect those vector fields $f_i$ which are not smooth. A simple and fast way of measuring the smoothness of $f_i$ is based on the computation of the following quotients:

$$q_i(j,k) = \frac{a_i(j,k)}{b_i(j,k)} \tag{2}$$

$$a_i(j,k) = \left\| f_i\left(\mathbf{x}_j\right) - f_i(\mathbf{x}_k) \right\|^2 \tag{3}$$

$$b_i(j,k) = \left\| \mathbf{x}_j - \mathbf{x}_k \right\|^2 \tag{4}$$

where $\mathbf{x}_j$ and $\mathbf{x}_k$ are the colors in the background model of two pixels $j, k \in W_i$, and $f_i(\mathbf{x}_j)$ and $f_i(\mathbf{x}_k)$ are the colors of those pixels in the current frame. The vector field $f_i$ is non smooth whenever $q_i$ attains high values for some pairs $j, k \in W_i$. As seen in Fig. 1, a smooth transformation is one that maps similar colors in the background model to similar colors in the current frame, even if the colors change considerably from the background model to the current frame. That is, the distances $||\mathbf{x}_j - f_i(\mathbf{x}_j)||$ are irrelevant to the smoothness of $f_i$. This way we can manage the switching of lights of any color, for example yellow or blue lights. It must be pointed out that low values of $q_i$ are not interesting because they are usually associated with homogeneous foreground objects passing in front of textured backgrounds, which are adequately managed by standard foreground detection algorithms.

In practice pixel noise can lead to large errors in the estimation of $q_i$, in particular if the denominator $b_i$ contains noise. We alleviate this by considering a filtered quantity $\phi_i$:

$$\phi_i(j,k) = \begin{cases} \dfrac{a_i(j,k)}{B_{low}} & \text{if } b_i(j,k) < B_{low} \\ 0 & \text{if } b_i(j,k) > B_{high} \\ q_i(j,k) & \text{otherwise} \end{cases} \tag{5}$$

where $B_{low}$ and $B_{high}$ are suitable lower and upper thresholds for the denominator value $b_i$, with $B_{low} < B_{high}$. Please note that $B_{low}$ manages low lighting conditions, where pixel values have little precision. On the other hand, $B_{high}$ ensures that highly textured backgrounds (which are associated to high values of $a_i$ and $b_i$) are not mistaken as non

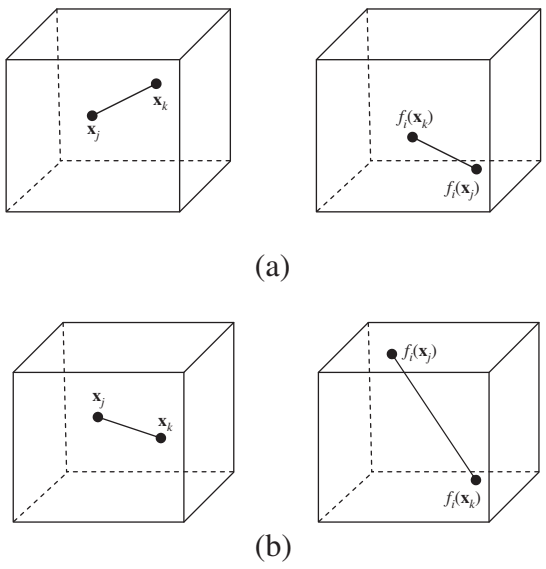



**Fig. 1.** Color transformations from the background model to the current frame: (a) smooth, (b) rough. The cubes represent the tristimulus color space.

smooth color transformations. Finally we use the highest $\phi_i(j,k)$ as a measure of the roughness of $f_i$:

$$\alpha_i = min\left(1, \frac{1}{K_\alpha} \max_{j,k \in W_i} \phi_i(j,k)\right) \quad (6)$$

where the min function ensures that $\alpha_i \in [0, 1]$, and $K_\alpha$ are a suitable scaling parameter.

On the other hand, the second membership value $\beta_i$ is obtained from the squared Euclidean distance between the color of pixel $i$ in the current frame and the color stored in the background model for pixel $i$:

$$\beta_i = \frac{1}{K_\beta} \|\mathbf{x}_0 - f_i(\mathbf{x}_0)\|^2 \quad (7)$$

where $\mathbf{x}_1$ and $f(\mathbf{x}_1)$ are the colors of the pixel at hand in the background model and in the current frame, respectively; and $K_\beta$ is another scaling parameter.

The possible values of the three fuzzy membership variables are mapped to eight possible illumination states for pixel $i$ according to Table 1. The standard fuzzy set operations are used to compute the fuzzy memberships of the eight states from $\alpha_i$, $\beta_i$, $\gamma_i$. Then a defuzzification is carried out by declaring pixel $i$ to be in the state with the maximum membership.

Pixels in state 'dubious' must be corrected to either 'Foreground object' or 'Background with illumination change'. That is, given a pixel with a smooth color transformation and a large difference with respect to the background model color, which is classified as foreground by the baseline algorithm, we cannot tell whether it is a real foreground object or a background pixel subject to an illumination change. At the pixel level this ambiguity cannot be solved. Hence we propose a frame level procedure for this task. From each dubious pixel $i$ we trace lines to the four directions up, down, left, and right, and to the four diagonals. If any of these lines hits a pixel in the 'Foreground object' or 'Background with illumination change' states, then we count a vote for that state. If a line exits the frame without having hit a pixel in any of these two states, then that line does not cast any vote. Finally, pixel $i$ is changed to the state with the most votes. If there is a tie, then no correction is made. The rationale behind this procedure is that dubious pixels belonging to foreground objects are usually located in the interior of the objects. Hence, for these pixels most of the eight lines hit some pixels in the 'Foreground object' state which lie in the borders of the object. It must be noted that dubious pixels are more frequent in objects with a homogeneous interior. On the other hand, dubious pixels belonging to the background are usually surrounded by some pixels in the 'Background with illumination change' state.

Fig. 2 exemplifies the illumination state estimation procedure described in this subsection. Please note that the states are represented in the pseudocolors specified in Table 1. As shown, the dubious pixels inside the person are corrected to the 'Foreground object' state, while those which lie in his shadow are corrected to the 'Background with illumination change' state. Additionally, Fig. 3 shows the flowchart of our proposal for the $k$-th frame.

**Table 1**
Possible illumination states of a pixel $i$.

| Rough | Difference | Baseline | Pseudocolor | State description |
|---|---|---|---|---|
| False | False | False | Black | Inactive background |
| False | False | True | Blue | Error in the background model (usually corresponds to background) |
| False | True | False | Green | Background with illumination change |
| False | True | True | Cyan | Dubious (must be corrected) |
| True | False | False | Red | Background near a foreground object |
| True | False | True | Magenta | Very unlikely occurrence (it can be assigned to the foreground) |
| True | True | False | Yellow | Error in the background model (usually corresponds to foreground) |
| True | True | True | White | Foreground object |

### 2.2. Pixel reset procedure

Once the illumination state of every pixel has been computed by the procedure explained in the previous subsection, a decision must be made about which pixels must be reset. A reset should be carried out whenever the system has failed to detect an illumination change in the background, so that the background model is not updated and pixels stay wrongly in the foreground state during many frames. We keep a counter $c_i$ for each pixel $i$ which counts how many consecutive frames the pixel has been declared as foreground by the baseline background model. If $c_i$ surpasses a prespecified limit $C_{limit}$, then we assume that the baseline algorithm has failed, so that the pixel must be reset. A pixel is reset by application of the initialization procedure of the baseline algorithm to it.

The illumination state of the pixels is used to perform an early reset of some pixels by increasing their counters $c_i$ when an illumination change has been detected on them. Two rules are applied on each video frame. The first one ('soft reset') says that pixels $i$ which are in the 'Background with illumination change' state in the current frame modify their counters as follows:

$$c_i = max\left(c_i, C_{soft}\right). \quad (8)$$

The second rule ('hard reset') says that pixels $i$ which have been in the 'Background with illumination change' state continuously for the last $K_{hard}$ frames modify their counters as follows:

$$c_i = max(c_i, C_{hard}) \quad (9)$$

where $C_{hard} > 0$ and

$$0 < C_{soft} \leq C_{hard} \leq C_{limit}. \quad (10)$$

The soft and hard reset rules model situations where the likelihood of an illumination change is low and high, respectively. The longer a pixel stays in the 'Background with illumination change' state, the most likely that the illumination change is real.

## 3. Experimental results

This section analyzes the performance of our proposal[1]. To this end we compare the original versions of several methods against these same methods modified according to our proposal. For this study various sequences and a wide set of parameters have been employed in order to obtain a clear idea of the merits of the considered alternatives. The methods to be tested are described in Subsection 3.1, and the video sequences are presented in Subsection 3.2. Our choice for the parameters of the competing models is explained in Subsection 3.4, and the quantitative performance measures are outlined in Subsection 3.3. The effect of the values of the parameters on the performance of the methods is studied in Subsection 3.4. Finally, the qualitative and quantitative results are shown in Subsections 3.5 and 3.6, respectively.

### 3.1. Methods

Our proposal analyzes the scene at a low level pixel by pixel, as is the case of the so-called pixel-level methods. This is why we use a set of five methods of this type for the experiments. They have been chosen on the basis of the availability of a public and reasonably well tested implementation, and their popularity in the computer vision community.

[1] The source code and some demo videos of our proposal are available at http://www.lcc.uma.es/%7Eezeqlr/lighting/lighting.html.

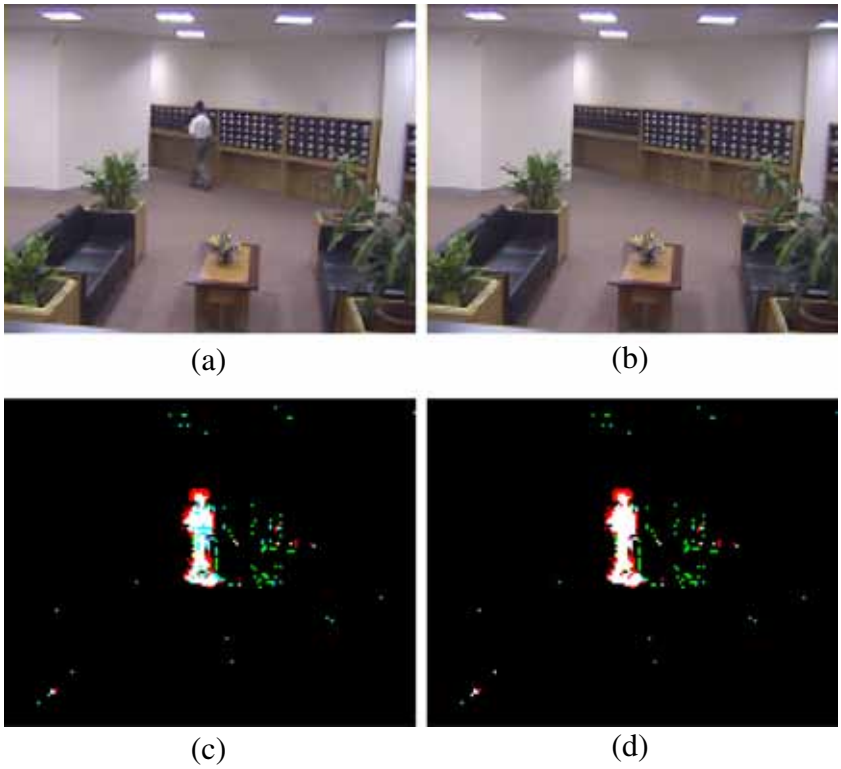



**Fig. 2.** Illumination state estimation: (a) current frame, (b) background model, (c) illumination states before dubious pixel correction, (d) illumination states after dubious pixel correction.

In order to fulfill these requirements, we have restricted our attention to version 1.3.0 of the BGS library. It is freely accessible from its website[2]. The first studied method is Pfinder [30], noted WrenGA, which uses only a single Gaussian per pixel. Also, three methods have been selected as representatives of the methods based on mixtures of Gaussians: GrimsonGMM [26], GMMV1 [13] and GMMV2 [34]. A fifth method has been chosen which is based on a fuzzy approach to self-organizing background subtraction (SOBS) [18]; it will be named FASOM. On the other hand, we have tested three background subtraction methods which are specifically designed for this kind of problems and therefore have not been modified with our proposal: Agrawal [1], Horprasert [9] and Reddy [21]. We have used the implementation available in the author's website[3] for Agrawal, a freely available implementation[4] to test Horprasert, and finally the authors of the Reddy method have provided us the source code.

Except Agrawal and Horprasert which use Matlab, all the code is written in C++ language. Version 2.4.3 of the OpenCV Computer Vision library[5] has also been employed. Please note that GMMV1 and GMMV2 are the current standard foreground detection methods of the OpenCV library.

In order to reduce the CPU time of our proposal, the Threading Building Blocks library version 4.1 has been used. It is also freely available in its website[6]. All the experiments have been carried out on a desktop PC with a quad core 3.10 GHz CPU, 6 GB RAM and standard hardware. No GPU acceleration has been done.

To make a comparison as clear and fair as possible, no additional post-processing has been done either in the original algorithms or their modified versions.

### 3.2. Sequences

With the aim of analyzing the behavior of the methods in real situations, we consider changes in spot lights and also to ambient light, testing a total of eight real sequences: five outdoor and three indoor. The objects to detect are persons or vehicles. The chosen sequences are available on the Internet. They represent different conditions like abrupt lighting changes, such as switching a light; or soft such as the passage of clouds in an outdoor scene. Moreover, although most of the selected sequences present midway objects, there are sequences with distant objects and other with close objects.

The first indoor sequence is Lobby (LB), available on this website[7], which has abrupt lighting changes and is also characterized by having background areas particularly vulnerable to *camouflage* effects. This situation occurs when the algorithm erroneously segments as background some regions that in fact belong to the foreground because they have very similar colors. The next indoor sequence is RecCenter (RC) which exhibits smooth lighting changes and a large number of people crossing the scene, available at GTILT Dataset[8]. The opposite situation happens in LightSwitch (LS), where illumination changes are abrupt and there is only one person who traverses the scene several times during the sequence.

On the other hand, outdoor scenes are represented by the following videos: Cars1 (C1), Cars3 (C3), Roadside (RS), PED1 (P1) and Bank (BK), which have been obtained also from GTILT Dataset. C1 and C2 are urban sequences characterized by having most objects relatively close to the camera. RS is a rural sequence that presents lighting changes affecting the image progressively as the clouds move over the scene. P1 is very interesting because it has significant changes in lighting and there are foreground objects near and far from the camera. Finally BK is a sequence with large illumination changes, in which both people and vehicles appear at close range.

[2] https://github.com/andrewssobral/bgslibrary.
[3] http://www.umiacs.umd.edu/aagrawal/cvpr06/EdgeSuppression.html.
[4] http://www.markyd13.com/code/background_subtraction/.
[5] http://sourceforge.net/projects/opencvlibrary.html.
[6] http://threadingbuildingblocks.org/.
[7] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html.
[8] http://www.ece.gatech.edu/research/pica/GTILT.html.

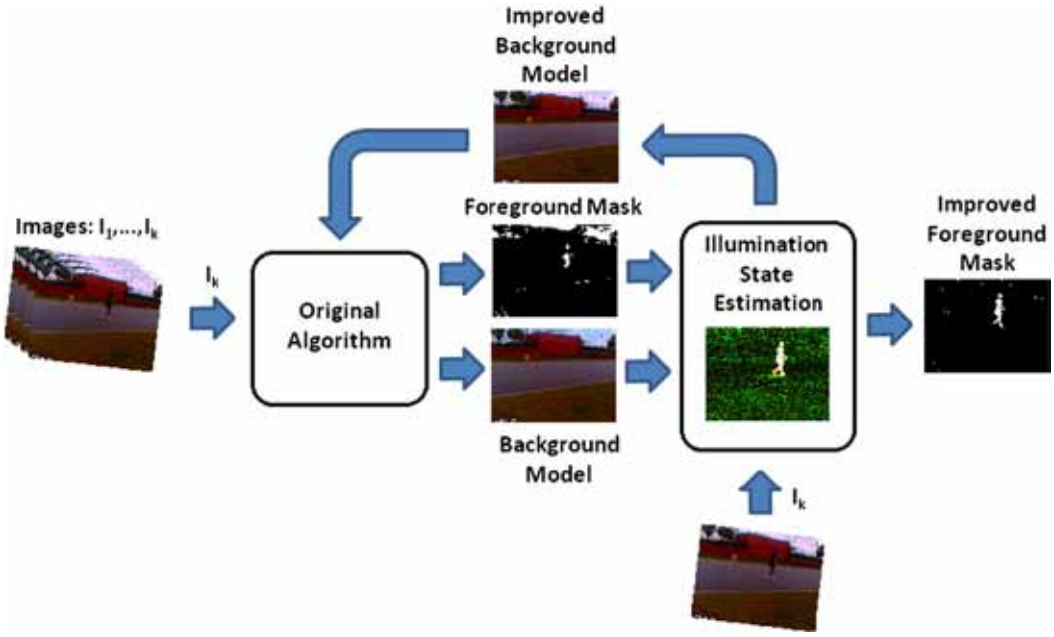



**Fig. 3.** Flowchart of our proposal.

To be as fair as possible, a constant sampling rate was used to determine which frames would be used as ground truth. Since the frequencies at which foreground objects appear in the scene are different in each video, the sampling frequency is also different in each case so that all benchmarks have a similar number of ground truth frames. For Lobby, half of the ground truth frames available in the repository only consider the last 200 frames in a sequence of 1546 frames, so we chose a wider sample representing 1446 frames (the first hundred frames are reserved for training), including frames with different amounts of lighting, and we performed a manual segmentation of these frames. Furthermore the GTILT Dataset provides one or no ground truth frames depending on the case, so that the ground truth frames of these sequences have also been segmented manually.

### 3.3. Performance measures

We have selected six quantitative performance measures to compare the segmentation methods we have considered. First of all, the accuracy of a method is obtained as follows (higher is better):

$$AC = \frac{\text{card}\,(A \cap B)}{\text{card}\,(A \cup B)} \tag{11}$$

where 'card' stands for the number of elements of a set, $A$ is the set of all pixels which belong to the foreground, and $B$ is the set of all pixels which are classified as foreground by the analyzed method:

$$A = \{\mathbf{t} \,|\, \chi(\mathbf{t}) = 1\} \tag{12}$$

$$B = \{\mathbf{t} \,|\, \widetilde{\chi}(\mathbf{t}) = 1\}. \tag{13}$$

On the other hand, the proportions of false negatives and false positives (lower is better) are given by:

$$FN = \frac{\text{card}\,(A \cap \overline{B})}{\text{card}\,(A \cup B)} \tag{14}$$

$$FP = \frac{\text{card}\left(\overline{A} \cap B\right)}{\text{card}\,(A \cup B)}. \tag{15}$$

From set theory we know that:

$$AC,\ FN,\ FP \in [01] \tag{16}$$

$$AC + FN + FP = 1. \tag{17}$$

The optimal performance would be achieved for $AC = 1$, $FN = 0$, $FP = 0$. On the other hand, the worst possible performance corresponds to $AC = 0$, $FN + FP = 1$.

**Table 2**
Parameter values for the experiments. The combinations of them form the set of all tested configurations for the original algorithm. For our proposal, we tested the parameters of the first five methods in combination with one value of $C_{limit} = \{50, 80, 110\}$.

| Method | Parameters |
|---|---|
| WrenGA | Threshold, $\mathcal{N} = \{6, 8, 10, 12, 14\}$<br>Alpha, $\alpha = \{0.0025, 0.005, 0.0075, 0.01\}$<br>Learning frames, $\ell = \{20, 30, 60\}$ |
| GrimsonGMM | Threshold, $\mathcal{N} = \{8, 9, 10, 11, 12\}$<br>Learning rate,<br>$\alpha = \{0.0025, 0.006, 0.0095, \ldots$<br>$0.013, 0.0165, 0.02\}$<br>Number of Gaussians, $\mathcal{K} = \{3, 4, 5\}$ |
| GMMV1 | History, $\mathcal{H} = \{50, 100, 150\}$<br>Number of Gaussians, $\mathcal{K} = \{2, 3, 5\}$<br>Threshold, $\mathcal{N} = \{0.4, 0.55, 0.7\}$<br>Noise sigma, $\sigma = \{5, 10, 15\}$ |
| GMMV2 | History, $\mathcal{H} = \{50, 100, 200, 300\}$<br>Threshold, $\mathcal{N} = \{14, 16, 18, 20, 22, 24\}$ |
| FASOM | Sensitivity, $s_1 = \{65, 90, 115\}$<br>Training sensitivity, $s_0 = \{100, 200, 300\}$<br>Learning rate, $\alpha_1 = \{20, 40\}$<br>Training learning rate, $\alpha_0 = \{180, 240, 300\}$<br>Training step, $\varepsilon = \{60, 100\}$ |
| Agrawal | Sigma, $\sigma = \{0.3, 0.4, 0.5\}$<br>Threshold small, $\mathcal{N} = \{10, 15, 25\}$<br>Low threshold, $\tau_1 = \{0.15, 0.3, 0.45\}$<br>High threshold, $\tau_2 = \{0.9, 0.8, 0.7\}$ |
| Horprasert | Normalized chromaticity distortion,<br>$\tau_{CD} = \{20000, 200000, 2000000, 20000000\}$<br>Normalized brightness distortion,<br>$\tau_{\alpha lo} = \{-10, -20, -30\}$<br>Low brightness threshold, $\tau_{\alpha 1} = \{6, 10\}$<br>High brightness threshold, $\tau_{\alpha 2} = \{-6, -10\}$ |
| Reddy | Block size, $B_S = \{2, 8, 14\}$<br>Block advancement, $B_A = \{1, 2, 4, 8\}$ |

The precision and recall measures can be computed as follows (higher is better):

$$PR = \frac{\text{card}\,(A \cap B)}{\text{card}\,(B)} \tag{18}$$

$$RC = \frac{\text{card}\,(A \cap B)}{\text{card}\,(A)} \tag{19}$$

$$PR,\ RC \in [01]. \tag{20}$$

Often there is a tradeoff between precision and recall, i.e. we can make one of them grow as desired at the expense of diminishing the other. Finally, the F-measure combines both of them into an overall performance measure:

$$FM = \frac{2 \cdot PR \cdot RC}{PR + RC}. \tag{21}$$

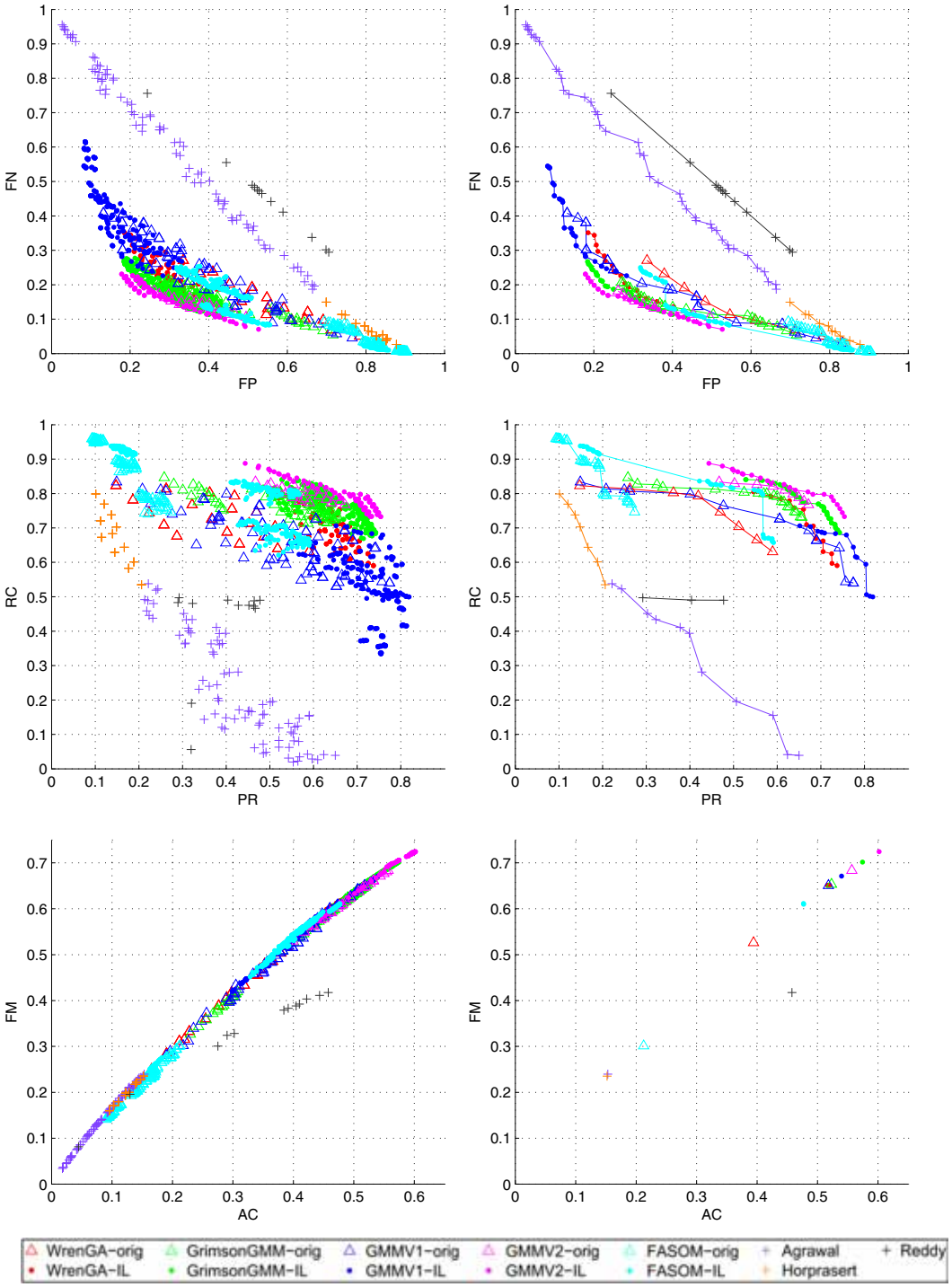



**Fig. 4.** Quantitative performance of the chosen configurations. First column shows in this order false negatives (FN) versus false positives (FP), recall (RC) versus precision (PR) and F-measure (FM) versus accuracy (AC). Second column shows the same measures for Pareto front configurations. Each point corresponds to a configuration of an algorithm and represents the mean of the average results achieved in each sequence. The original versions are marked with triangles, our proposal with dots and Agrawal, Horprasert and Reddy with crosses.

### 3.4. Parameter selection

Here the parameter selections for the competing methods are presented. Table 2 shows the parameters tested in the simulations. All the values that have been found to yield good results for at least some of the sequences have been included. Moreover, we have also included the values recommended by the BGS library and those from the original implementations. An exhaustive parameter search would be unfeasible

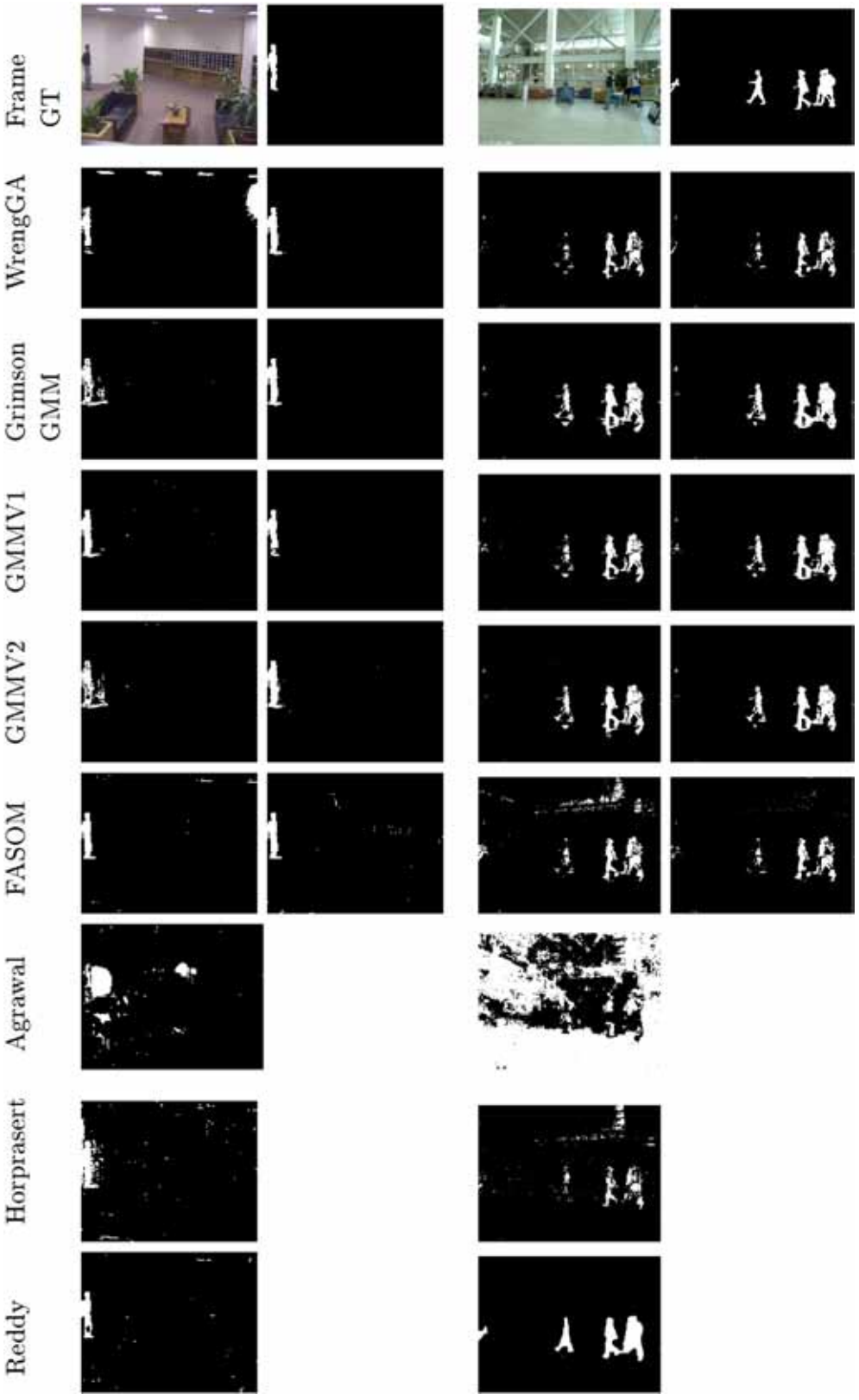



**Fig. 5.** Experimental results on indoor videos. The odd columns show the original frame and the output of the original algorithms. The even columns show ground truth and the results of the versions modified with our proposal for the original algorithms. From left to right: Lobby and RecCenter; frames 2240 and 490, respectively. As seen, our proposal attenuates the effects of the cast shadows problem.

due to the number of parameters to be tuned for each method and sequence. It was found that our proposal only requires tuning parameter $C_{limit}$, which determines when the pixels are reset, as seen in Sub-section 2.2 because the other parameters do not appreciably affect the performance. Consequently the following fixed values were used: $\beta_{low} = 10$, $\beta_{high} = 50$, $K_\alpha = 100$, $K_\beta = 20$, $C_{soft} = 25$ and $C_{hard} = 50$. The combination of all considered values for each parameter yields the set of tested configurations for each algorithm.

Fig. 4 shows the results of each algorithm in terms of various performance measures. In the case of FN versus FP, the closer the points are to the origin, the better the segmentation. For the other performance measures, the farther from the origin, the better the achieved results.

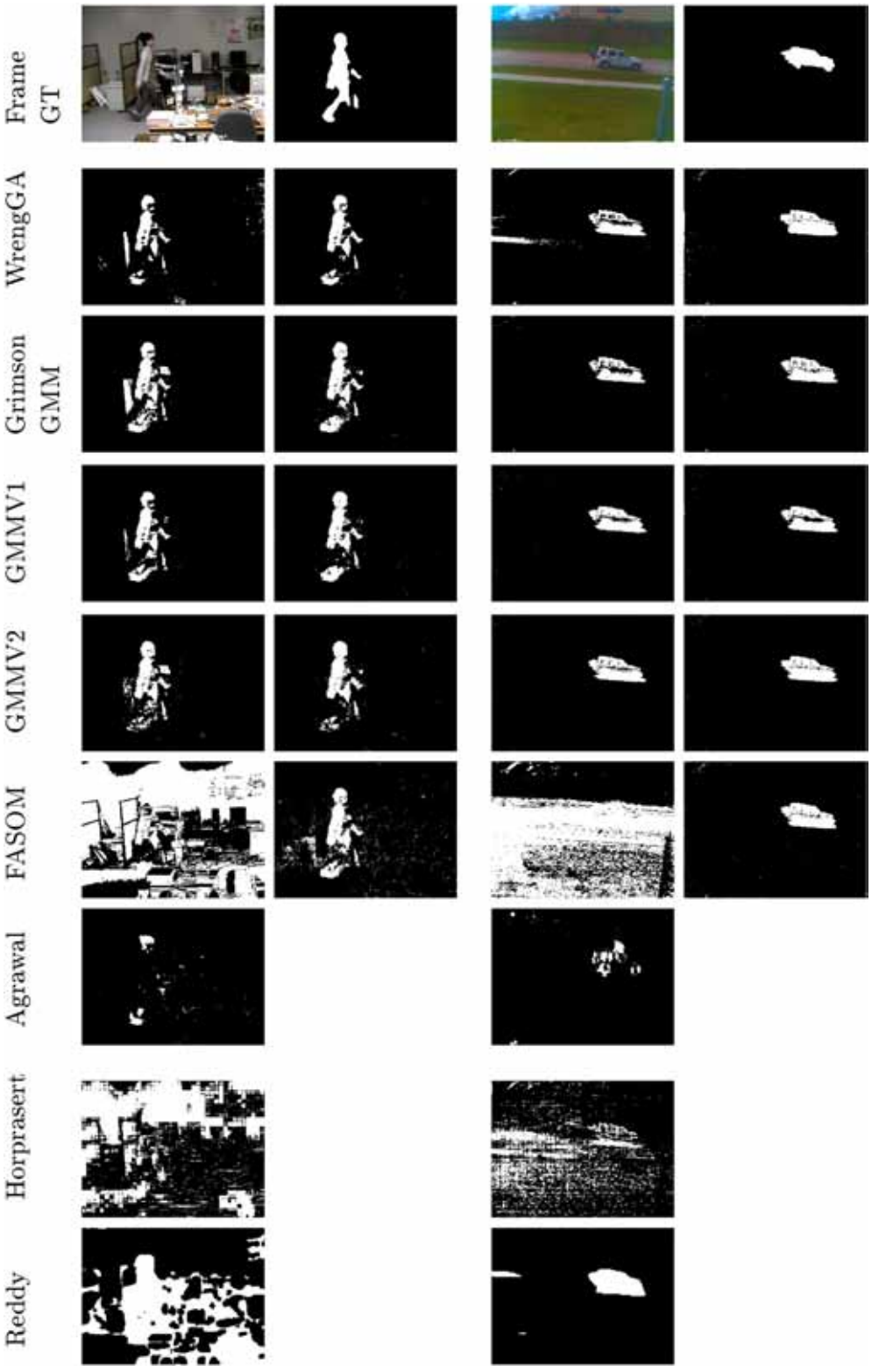



**Fig. 6.** Experimental results on indoor and outdoor videos. The odd columns show the original frame and the output of the original algorithms. The even columns show ground truth and the results of the versions modified with our proposal for the original algorithms. From left to right: LightSwitch and Cars3; frames 1220 and 310 respectively. The original methods segment some background objects as foreground due to changes in lighting, while our proposal corrects this behavior.

Therefore, this figure indicates that our proposal improves the original algorithm for all the considered methods, since for every configuration of the original algorithm there is a configuration of our proposal that outperforms it. It is also worth mentioning that Agrawal, Horprasert and Reddy are unable to obtain good results in these sequences.

### 3.5. Qualitative results

In this subsection the quantitative performance of the methods is assessed. Figs. 4 to 7 display the results obtained by the original and modified algorithms in some frames of the studied sequences. In

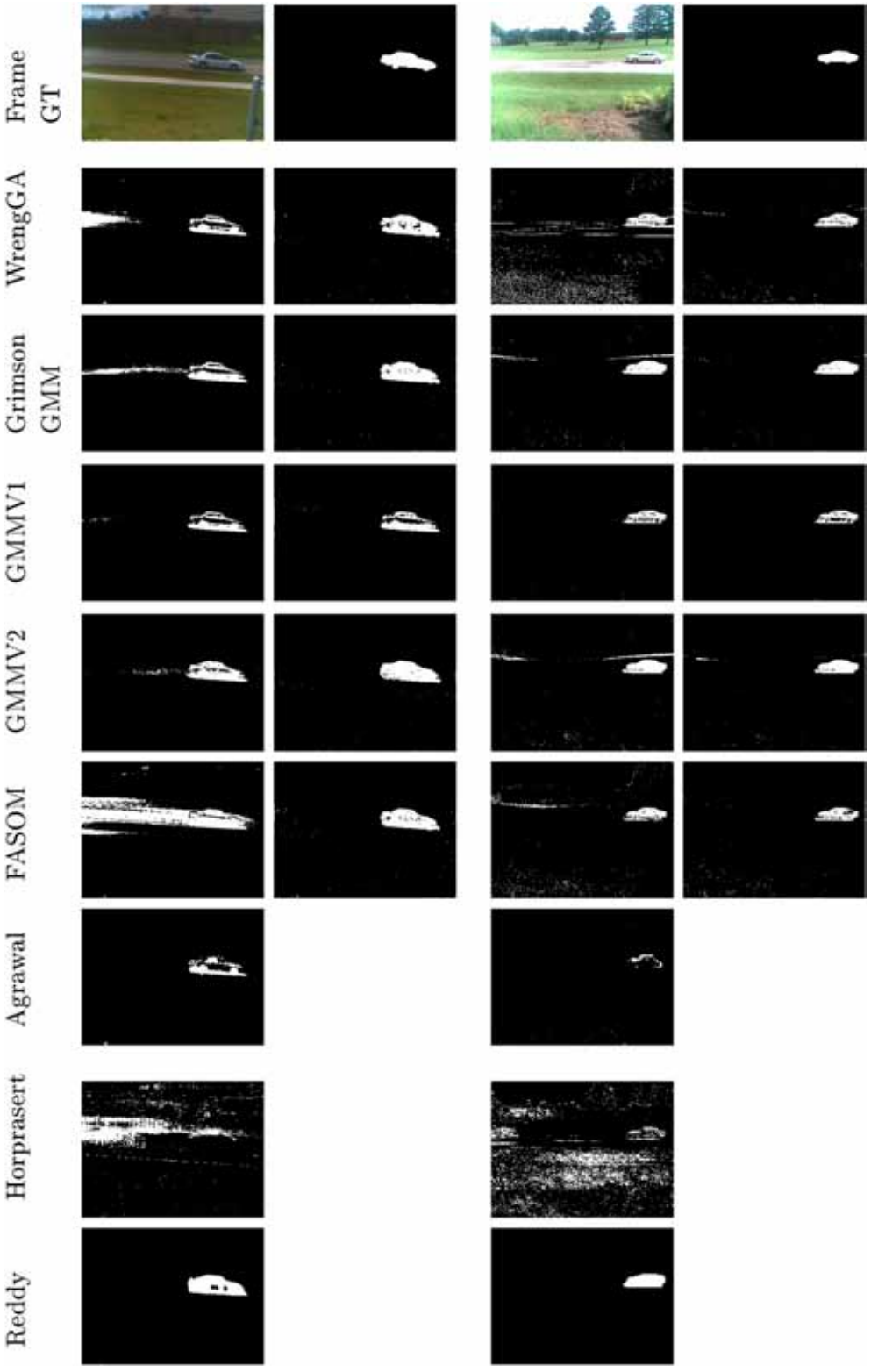



**Fig. 7.** Experimental results on outdoor videos. The odd columns show the original frame and the output of the original algorithms. The even columns show ground truth and the results of the versions modified with our proposal for the original algorithms. From left to right: Cars1 and Roadside; frames 178 and 135 respectively. False positive regions are substantially removed by our algorithm, and the gaps in the detected foreground objects are filled.

general it can be observed that our proposal is capable of removing a significant amount of false positives and, to a lesser extent eliminates some false negatives.

It is interesting to note that, in the sequences of Fig. 5, WrenGA and FASOM suffer from false positives due to changes in the light coming from the spotlights, but our proposal is able to correct this behavior. Moreover, virtually all original algorithms suffer from *cast shadows* in these sequences and again our proposal is able to attenuate the problem in several cases. This kind of artifact occurs when foreground objects cast shadows and they are segmented erroneously as part of the foreground.

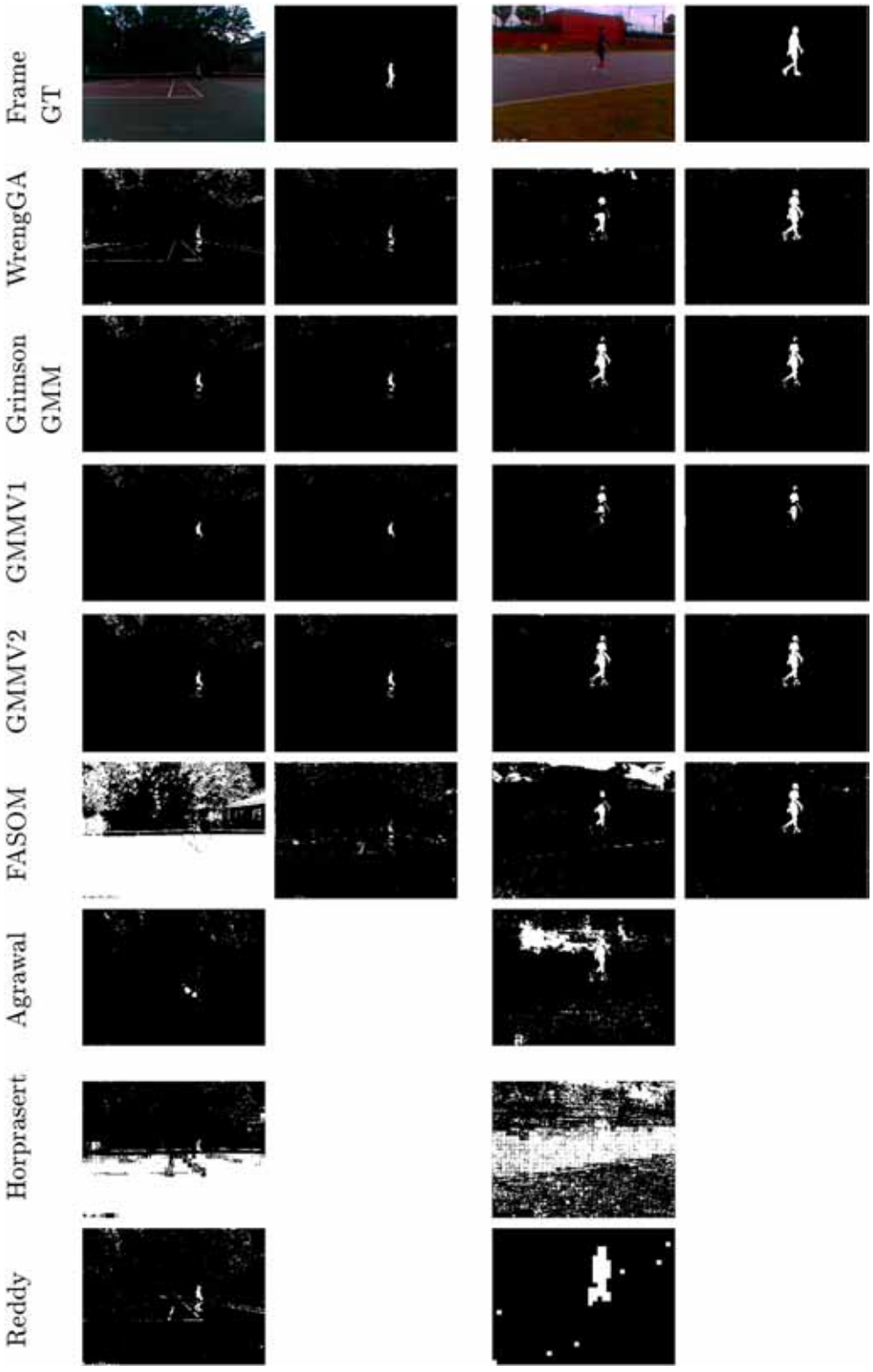



**Fig. 8.** Experimental results on outdoor videos. The odd columns show the original frame and the output of the original algorithms. The even columns show ground truth and the results of the versions modified with our proposal for the original algorithms. From left to right: Ped1 and Bank; frames 430 and 477 respectively. Isolated false positives are largely removed by our approach.

Camouflage is another common problem for the tested segmentation methods. For example, in the first column (Lobby), and particularly in the third (WrenGA) and fifth (GrimsonGMM) rows, the person is not completely segmented. In contrast to this, the second column shows how our proposal is able to alleviate this problem significantly.

For the last indoor sequence (LightSwitch) in Fig. 6, the chosen frame is a clear example in which background objects are segmented as foreground due to changes in lighting. On the left side of the scene there is an object which is segmented as part of the foreground. Our proposal detects the occurrence of a change of lighting in this position and therefore succeeds in segmenting it. In the same frame FASOM segments the scene poorly, and it produces a large number of false positives; this behavior also occurs in other sequences. As seen, our modification can correct this problem to the point of achieving results which are similar to those of the remaining algorithms.

On the other hand, the remaining sequences (Figs. 7 and 8) contain outdoor frames only. There are a significant number of false positives in the background due to lighting changes and our proposal is able to overcome this problem. These sequences also exhibit an important number of false negatives; e.g. the frames corresponding to C3, C1 and Bank show that the vehicle is not fully segmented. However, our proposal is able to fill in the gaps in a satisfactory way.

As we see in Fig. 5 our proposal produces some isolated foreground pixels for LB sequence with FASOM. In order to correct these effects we could use a two step post-processing, namely first erode the output image with a structuring element operator and then dilate the result using the same element. Two structuring elements have been considered; a square element $\mathbf{E}_{square}$ and a diamond shaped element $\mathbf{E}_{diamond}$:

$$\mathbf{E}_{square} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{22}$$

$$\mathbf{E}_{diamond} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{23}$$

Fig. 9 exemplifies the results of such post-processing. It shows the results of the original and the modified version of FASOM in a LB frame. If we do not use post-processing our proposal is able to get 0.79 F-measure. When we use $\mathbf{E}_{square}$ as the structuring element, it achieves 0.88 (fourth column). Finally, by employing $\mathbf{E}_{diamond}$ and the structuring element, we increase the score to 0.89 (fifth column). It must be highlighted that no post-processing has been used in any experiments other than those reported in Fig. 9.

### 3.6. Quantitative results

The above subsection has focused on the qualitative results. A clearer view of the differences among our proposal and the original algorithms can be obtained from Fig. 10. It shows the difference between the performance achieved by our proposal and that of the original version in terms of F-measure. The configuration that performs best in terms of F-measure is employed for this figure.

The improvement is different in each method. The FASOM method and, to a lesser extent WrenGA are cases where an improvement over the original algorithm can be seen more clearly, since our approach obtains better results in most situations.

Fig. 11 shows the performance in terms of F-measure of the Agrawal, Horprasert and Reddy methods compared with our modified versions of the rest of methods. All plots show poor results for these three methods and only in some frames they are able to outperform our proposal.

In order to get a more comprehensible view of the differences it is better to use Table 3, which displays the mean values corresponding to Figs. 10 and 11. If we consider F-measure our proposal overcomes the original versions in most cases, only original GMMV1 is able to get better results compared to its modified version in LB, C1 and BK. We have to mention that original WrenGA and GrimsonGMM also get better results in RC and BK, but the difference with our proposal is smaller.

Complementary to F-measure, the performance using the accuracy measure is also analyzed in Table 3. It can be observed that our proposal improves the original algorithm even more than with F-measure. There are only three videos where there is a method that our algorithm cannot improve the original version: LB, RC and BK. Again GMMV1 is the clearest example of this, obtaining better results in LB and BK. WrenGA also overcomes our proposal in RC, but the difference is small.

It is interesting to note that Reddy gets better results than our proposal in two sequences according to accuracy. However this is not the case for the performance in terms of F-measure. Furthermore, Reddy is the only method that is able to get a higher score on accuracy than on F-measure. This behavior occurs because accuracy benefits algorithms with few FN, being less sensitive to FP than F-measure, which penalizes both situations [25]. This is the case of Reddy as we see in Fig. 4, where most configurations get more FP than FN.

A statistical significance study has been carried out for all the quantitative performance measures. The nonparametric Friedman test with the corresponding post-hoc Dunn test have been used to determine whether the difference of the best performing approach with respect to the second best performing one is statistically significant. These tests are robust for comparisons over multiple datasets [4]. A 95% confidence level has been chosen in all cases. For each sequence, a configuration marked with * means that it is better than the second best configuration with a 95% confidence level. Our proposal achieves significantly better results in four of the eight sequences in terms of F-measure and in three sequences in the case of accuracy. None of the original algorithms is significantly better than the modified version, while Reddy is significantly better in two sequences and according to accuracy only.

Finally, Table 4 shows the FPS performance of each algorithm for all sequences. Since real-time requirements are satisfied when it exceeds the threshold of 15 fps, it is observed that all algorithms verify this condition except Agrawal and Reddy. As we see, GrimsonGMM and FASOM are the methods improved with our algorithm with the worst


**Fig. 9.** Example of post-processing using morphological operators. From left to right: Ground truth, original algorithm, proposed modification without post-processing, proposed modification with post-processing using a square structuring element and proposed modification with post-processing using a diamond shaped structuring element.

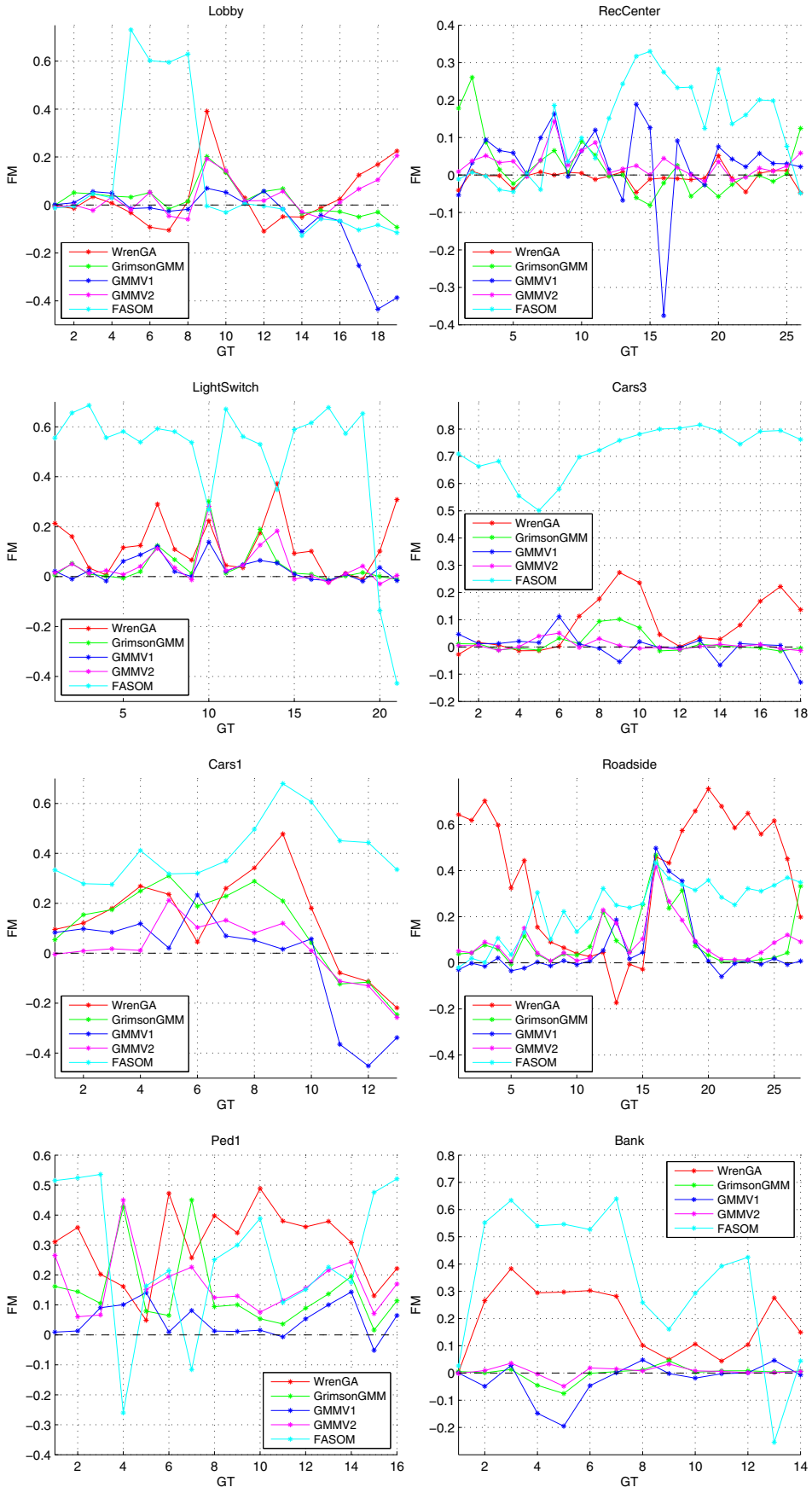



**Fig. 10.** Difference between the original and the proposed algorithm using F-measure. The horizontal axis shows the ground truth frame and the vertical axis the difference between our proposal and the original algorithm. A positive value means a better performance of our proposal.

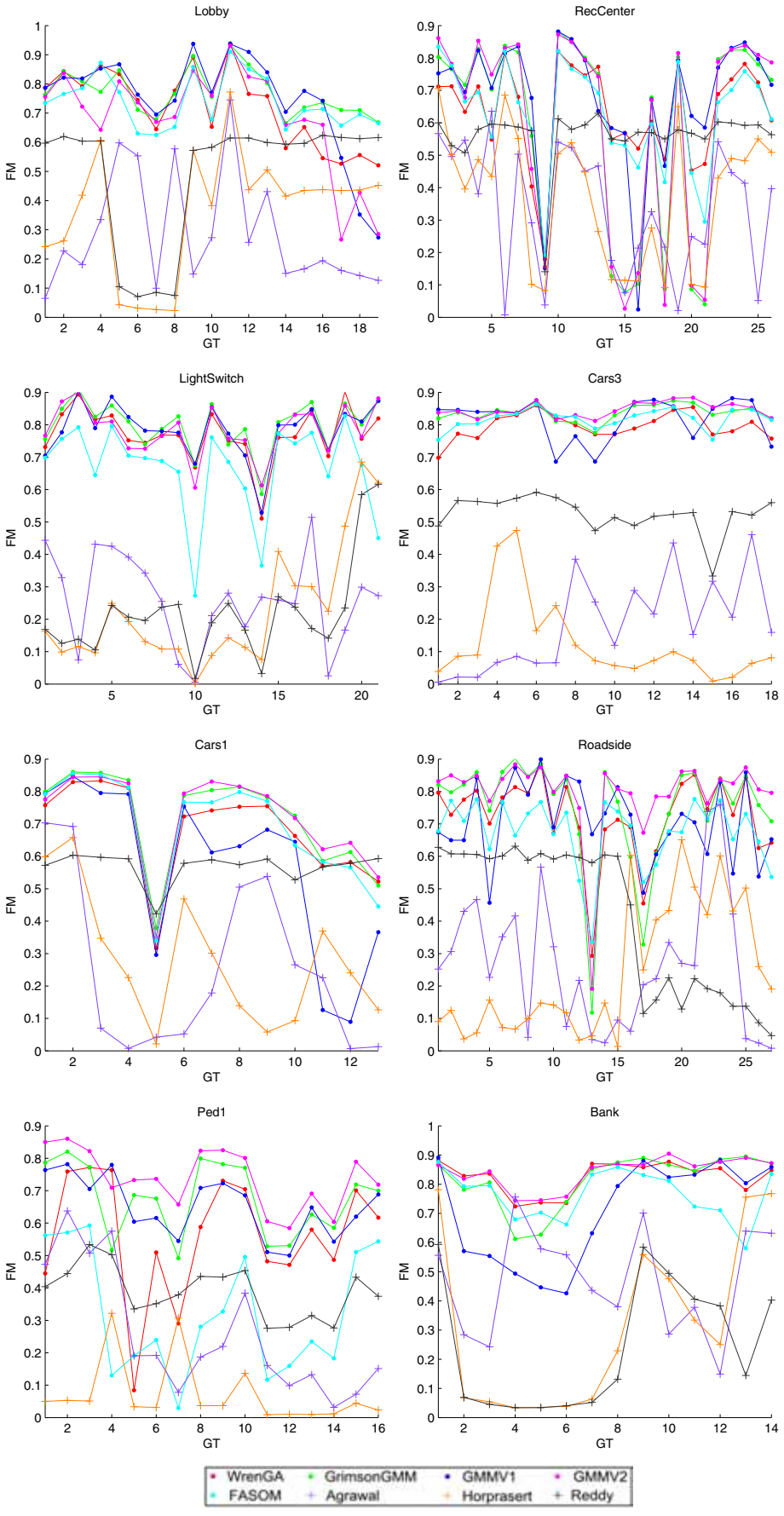



**Fig. 11.** Comparison among our modified methods and shadow/illumination detection methods (Agrawal, Horprasert and Reddy) using F-measure. The horizontal axis shows the ground truth frame and the vertical axis the F-measure results.

**Table 3**
Best results according to F-measure and accuracy. The first column denotes the sequence name, whereas the third and fourth columns show the best results according to F-measure of our proposal and the original one, respectively. Fifth and sixth columns correspond to accuracy results. The best version of a method for each sequence is shown in bold. An asterisk means that a result is significantly better than the second best result for a given sequence.

| | | F-Measure | | Accuracy | |
|---|---|---|---|---|---|
| Sequence | Method | Proposed | Original | Proposed | Original |
| LB | WrenGA | **0.72 ± 0.13** | 0.68 ± 0.19 | **0.58 ± 0.16** | 0.55 ± 0.20 |
| | GrimsonGMM | **0.77 ± 0.08** | 0.75 ± 0.06 | **0.63 ± 0.11** | 0.60 ± 0.09 |
| | GMMV1 | 0.74 ± 0.17 | 0.80 ± 0.06 | 0.62 ± 0.19 | **0.67 ± 0.09** |
| | GMMV2 | **0.68 ± 0.17** | 0.65 ± 0.21 | **0.55 ± 0.20** | 0.51 ± 0.20 |
| | FASOM | **0.74 ± 0.09** | 0.63 ± 0.31 | **0.59 ± 0.11** | 0.53 ± 0.27 |
| | Agrawal | n/a | 0.29 ± 0.19 | n/a | 0.18 ± 0.15 |
| | Horprasert | n/a | 0.36 ± 0.20 | n/a | 0.24 ± 0.15 |
| | Reddy | n/a | 0.49 ± 0.21 | n/a | 0.61 ± 0.30 |
| RC | WrenGA | 0.63 ± 0.15 | **0.64 ± 0.15** | 0.49 ± 0.19 | **0.49 ± 0.15** |
| | GrimsonGMM | **0.59 ± 0.31** | 0.57 ± 0.28 | **0.48 ± 0.28** | 0.45 ± 0.27 |
| | GMMV1 | **0.68 ± 0.20*** | 0.65 ± 0.17 | **0.54 ± 0.19** | 0.50 ± 0.17 |
| | GMMV2 | **0.60 ± 0.32** | 0.57 ± 0.31 | **0.49 ± 0.28** | 0.46 ± 0.27 |
| | FASOM | **0.63 ± 0.16** | 0.51 ± 0.23 | **0.49 ± 0.21** | 0.37 ± 0.20 |
| | Agrawal | n/a | 0.34 ± 0.19 | n/a | 0.22 ± 0.15 |
| | Horprasert | n/a | 0.37 ± 0.20 | n/a | 0.25 ± 0.16 |
| | Reddy | n/a | 0.56 ± 0.09 | n/a | 0.66 ± 0.14* |
| LS | WrenGA | **0.77 ± 0.08** | 0.65 ± 0.17 | **0.63 ± 0.10** | 0.50 ± 0.17 |
| | GrimsonGMM | **0.80 ± 0.07** | 0.76 ± 0.13 | **0.67 ± 0.11** | 0.62 ± 0.15 |
| | GMMV1 | **0.78 ± 0.08** | 0.75 ± 0.10 | **0.65 ± 0.10** | 0.61 ± 0.13 |
| | GMMV2 | **0.78 ± 0.08** | 0.74 ± 0.14 | **0.65 ± 0.10** | 0.60 ± 0.15 |
| | FASOM | **0.67 ± 0.14** | 0.18 ± 0.22 | **0.52 ± 0.14** | 0.12 ± 0.20 |
| | Agrawal | n/a | 0.26 ± 0.14 | n/a | 0.16 ± 0.09 |
| | Horprasert | n/a | 0.22 ± 0.18 | n/a | 0.14 ± 0.14 |
| | Reddy | n/a | 0.22 ± 0.14 | n/a | 0.17 ± 0.19 |
| C3 | WrenGA | **0.80 ± 0.04** | 0.71 ± 0.11 | **0.66 ± 0.05** | 0.56 ± 0.13 |
| | GrimsonGMM | **0.84 ± 0.03** | 0.82 ± 0.05 | **0.72 ± 0.04** | 0.70 ± 0.07 |
| | GMMV1 | **0.82 ± 0.06** | 0.81 ± 0.05 | **0.69 ± 0.09** | 0.69 ± 0.07 |
| | GMMV2 | **0.85 ± 0.02** | 0.84 ± 0.02 | **0.73 ± 0.03** | 0.73 ± 0.04 |
| | FASOM | **0.82 ± 0.03** | 0.10 ± 0.10 | **0.69 ± 0.04** | 0.06 ± 0.06 |
| | Agrawal | n/a | 0.18 ± 0.14 | n/a | 0.11 ± 0.09 |
| | Horprasert | n/a | 0.12 ± 0.13 | n/a | 0.07 ± 0.08 |
| | Reddy | n/a | 0.53 ± 0.05 | n/a | 0.58 ± 0.10 |
| C1 | WrenGA | **0.68 ± 0.14** | 0.54 ± 0.19 | **0.53 ± 0.15** | 0.39 ± 0.16 |
| | GrimsonGMM | **0.72 ± 0.14** | 0.61 ± 0.17 | **0.58 ± 0.16** | 0.46 ± 0.15 |
| | GMMV1 | 0.57 ± 0.25 | **0.60 ± 0.12** | **0.44 ± 0.23** | 0.44 ± 0.18 |
| | GMMV2 | **0.72 ± 0.14** | 0.71 ± 0.17 | **0.58 ± 0.16** | 0.57 ± 0.16 |
| | FASOM | **0.69 ± 0.16** | 0.28 ± 0.20 | **0.55 ± 0.17** | 0.18 ± 0.14 |
| | Agrawal | n/a | 0.25 ± 0.25 | n/a | 0.17 ± 0.19 |
| | Horprasert | n/a | 0.28 ± 0.19 | n/a | 0.18 ± 0.14 |
| | Reddy | n/a | 0.57 ± 0.05 | n/a | 0.67 ± 0.10* |
| RS | WrenGA | **0.72 ± 0.12** | 0.35 ± 0.26 | **0.58 ± 0.14** | 0.24 ± 0.22 |
| | GrimsonGMM | **0.75 ± 0.17** | 0.65 ± 0.25 | **0.62 ± 0.17** | 0.52 ± 0.23 |
| | GMMV1 | **0.71 ± 0.12** | 0.65 ± 0.20 | **0.56 ± 0.14** | 0.51 ± 0.19 |
| | GMMV2 | **0.80 ± 0.13*** | 0.70 ± 0.18 | **0.67 ± 0.13*** | 0.57 ± 0.17 |
| | FASOM | **0.68 ± 0.10** | 0.44 ± 0.18 | **0.52 ± 0.11** | 0.30 ± 0.15 |
| | Agrawal | n/a | 0.27 ± 0.21 | n/a | 0.17 ± 0.16 |
| | Horprasert | n/a | 0.24 ± 0.20 | n/a | 0.16 ± 0.14 |
| | Reddy | n/a | 0.44 ± 0.19 | n/a | 0.51 ± 0.31 |
| P1 | WrenGA | **0.56 ± 0.18** | 0.26 ± 0.19 | **0.41 ± 0.16** | 0.17 ± 0.14 |
| | GrimsonGMM | **0.67 ± 0.11** | 0.53 ± 0.20 | **0.52 ± 0.14** | 0.39 ± 0.16 |
| | GMMV1 | **0.65 ± 0.09** | 0.60 ± 0.11 | **0.49 ± 0.10** | 0.44 ± 0.11 |
| | GMMV2 | **0.74 ± 0.09*** | 0.57 ± 0.15 | **0.59 ± 0.11*** | 0.41 ± 0.15 |
| | FASOM | **0.32 ± 0.19** | 0.06 ± 0.09 | **0.21 ± 0.14** | 0.03 ± 0.06 |
| | Agrawal | n/a | 0.26 ± 0.19 | n/a | 0.16 ± 0.14 |
| | Horprasert | n/a | 0.07 ± 0.10 | n/a | 0.04 ± 0.06 |
| | Reddy | n/a | 0.39 ± 0.08 | n/a | 0.33 ± 0.11 |
| BK | WrenGA | **0.82 ± 0.05** | 0.63 ± 0.16 | **0.70 ± 0.08** | 0.48 ± 0.17 |
| | GrimsonGMM | 0.82 ± 0.09 | **0.82 ± 0.07** | **0.70 ± 0.12** | 0.69 ± 0.09 |
| | GMMV1 | 0.71 ± 0.17 | **0.73 ± 0.14** | 0.57 ± 0.20 | **0.59 ± 0.17** |
| | GMMV2 | **0.84 ± 0.05*** | 0.83 ± 0.05 | **0.73 ± 0.07*** | 0.72 ± 0.07 |
| | FASOM | **0.76 ± 0.08** | 0.42 ± 0.27 | **0.63 ± 0.11** | 0.31 ± 0.24 |
| | Agrawal | n/a | 0.47 ± 0.18 | n/a | 0.33 ± 0.16 |
| | Horprasert | n/a | 0.32 ± 0.29 | n/a | 0.23 ± 0.23 |
| | Reddy | n/a | 0.40 ± 0.13 | n/a | 0.37 ± 0.16 |

**Table 4**
Frames per second (FPS, higher is better) for each analyzed sequence. Each value corresponds to the best possible configuration according to the accuracy measurement. LB is 160 × 128 pixels, while the other sequences are 640 × 480 that have been reduced to 320 × 240 pixels.

| Sequence | Method | Proposed | Original |
|---|---|---|---|
| LB | WrenGA | 79 | 1040 |
| | GrimsonGMM | 59 | 288 |
| | GMMV1 | 124 | 606 |
| | GMMV2 | 149 | 765 |
| | FASOM | 102 | 260 |
| | Agrawal | n/a | 1 |
| | Horprasert | n/a | 239 |
| | Reddy | n/a | 76 |
| RC | WrenGA | 23 | 286 |
| | GrimsonGMM | 16 | 100 |
| | GMMV1 | 25 | 194 |
| | GMMV2 | 27 | 207 |
| | FASOM | 22 | 67 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 70 |
| | Reddy | n/a | 6 |
| LS | WrenGA | 22 | 283 |
| | GrimsonGMM | 16 | 41 |
| | GMMV1 | 21 | 154 |
| | GMMV2 | 25 | 191 |
| | FASOM | 22 | 73 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 66 |
| | Reddy | n/a | 6 |
| C3 | WrenGA | 23 | 286 |
| | GrimsonGMM | 16 | 96 |
| | GMMV1 | 23 | 170 |
| | GMMV2 | 25 | 222 |
| | FASOM | 23 | 71 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 67 |
| | Reddy | n/a | 11 |
| C1 | WrenGA | 22 | 291 |
| | GrimsonGMM | 17 | 110 |
| | GMMV1 | 25 | 177 |
| | GMMV2 | 25 | 214 |
| | FASOM | 23 | 69 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 67 |
| | Reddy | n/a | 5 |
| RS | WrenGA | 23 | 272 |
| | GrimsonGMM | 16 | 64 |
| | GMMV1 | 20 | 172 |
| | GMMV2 | 25 | 176 |
| | FASOM | 22 | 71 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 68 |
| | Reddy | n/a | 4 |
| P1 | WrenGA | 22 | 279 |
| | GrimsonGMM | 15 | 64 |
| | GMMV1 | 21 | 140 |
| | GMMV2 | 25 | 175 |
| | FASOM | 22 | 71 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 69 |
| | Reddy | n/a | 9 |
| BK | WrenGA | 24 | 287 |
| | GrimsonGMM | 15 | 48 |
| | GMMV1 | 25 | 169 |
| | GMMV2 | 26 | 201 |
| | FASOM | 22 | 74 |
| | Agrawal | n/a | 0.3 |
| | Horprasert | n/a | 68 |
| | Reddy | n/a | 156 |

results regarding to computational time. On the other hand the modified version of GMMV2 is the fastest modified method and, as we have previously shown, it is one of the best performing options. Our proposal has a higher computational load, but it must be taken into account that we do not use hardware acceleration, so it is possible to improve this feature.

## 4. Discussion

In this section several key features of our proposal are discussed:

- The main difference of our approach with respect to other algorithms for illumination change detection is that there is no specific form of the color change to be expected when an illumination change takes place. This means that our proposal does not depend on the physical properties of the lights that are present in the environment. Therefore, it can be applied to a wide range of situations where the lighting conditions vary in an unpredictable way. It must be highlighted that most current approaches assume some model of pixel color variation under illumination changes either explicitly or implicitly [7,11,31,29]. A particularly interesting case is the approach in [32], which only assumes that the sign of the difference between two pixel luminance measurements is maintained across illumination changes in a local neighborhood of the frame. However, this assumption does not hold in practice for textured regions, where some pixels can increase their luminance while others decrease. In contrast to this, our approach handles textured regions correctly because it checks whether pixels in a local neighborhood that had a similar color before the change continue to have a similar color after the change, so that pixels corresponding to different materials of the local texture are never compared unlike [32].
- As reported in the experiments, the approaches in [1,9,21] are outperformed by our proposal. This is because these methods consider an excessively restrictive model of illumination changes. In particular, a linear transformation in the RGB values is considered in [9], where the observed color is projected onto a line in the RGB color cube which is assumed to model all possible illumination states of the pixel. This is unrealistic, since the response of the materials with respect to light may not be linear, in particular if the color content of the incident light is different for the three RGB channels. The illumination change management in [21] considers that the angle subtended between the observed color and the estimated mean background color is small under varying illumination. Again this amounts to assuming that the response of the materials to light is similar for the three RGB channels, which is not realistic for the reasons outlined before. On the other hand, the approach in [1] focuses on detecting foreground objects as those with edges which are not present in the background model. This strategy will fail on homogeneous regions or noisy videos where the local edge information is not reliable. Also, the different response of the materials with respect to light means that local edges may have a different appearance when illumination changes.
- Given the previous considerations, a qualitative assessment of the challenges faced by the competing illumination change detection algorithms is carried out next. As we see in Fig. 12 Agrawal performs well in situations with clearly defined edges (first row), but fails to segment those frames with flat regions or camouflage effects (second row). Please note that these failures are due to unreliable edge information, as explained before. Horprasert is highly dependent on the training phase because the initial model is not updated. The first row in Fig. 13 shows a situation where this method correctly segments the image, but the second row shows that it fails to adapt to lighting changes that occur on the left side of the image, which exemplifies the limitations of its illumination change model. One of the parameters needed to configure Reddy is the block advancement. High values yield poorly defined edges on foreground objects and therefore high values of FP, see the last image of Fig. 8 which corresponds to a block advancement of 8. If we use a lower value (see first row of Fig. 14 or the last image of Fig. 5), the problem is partially corrected and the edges are smoother, but the computational performance drops down dramatically in terms of FPS. Another common problem is that Reddy does not perform well with variations in ambient light due to its poor illumination change model, see the second row of Fig. 14.
- The decision that a pixel has undergone an illumination change is not driven by binary variables (yes/no), but by fuzzy variables which are able to manage the inherent uncertainty in the determination of the illumination state of a pixel. This way the proposal is able to use as much information as possible from the evaluation of the lighting conditions. The validity of fuzzy approaches to foreground detection under varying lighting conditions has been demonstrated in literature [33,24,5].
- A procedure to combine the illumination state information from nearby pixels is established. This is of paramount importance, since at the pixel level it is not possible to distinguish a sudden illumination change from an incoming object with a homogeneous color. This situation, which is common to all pixel level approaches [23,17,15], is modeled in our proposal by means of the 'dubious' pixel state (Table 1).
- The experimental results show that our procedure is able to enhance the performance of several state of the art foreground detection methods, so that the difference is statistically significant with a high confidence level. It is also observed that the main part of the enhancement comes from the removal of false positives, which means that illumination changes are no longer mistaken as foreground objects as expected from this kind of algorithms [29,19,16,3,2].

## 5. Conclusions

We have presented a new approach to illumination change detection. Our proposal is designed to be attached to an existing foreground detection algorithm. It is based on the study of the local color transformation which the pixels undergo when a lighting change occurs. One of the advantages is that we only need to tune one parameter and, using a few different values, good results can be obtained. Therefore it is easily usable and does not require a vast expertise.

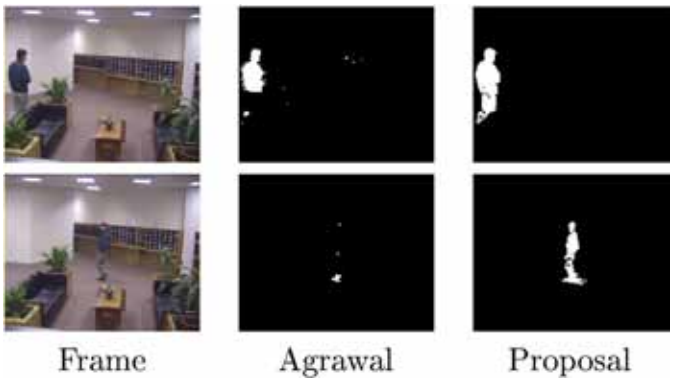



**Fig. 12.** Examples to illustrate typical Agrawal difficulties. First and second rows correspond to frame 2340 and 1365 of LB sequence. From left to right: Original frame, Agrawal results and GrimsonGMM modified with our proposal.

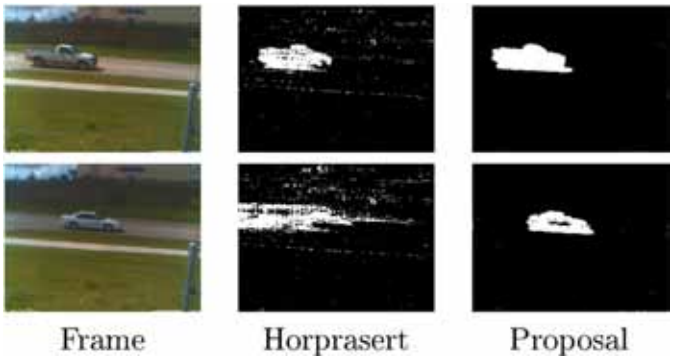



**Fig. 13.** Examples to illustrate typical Horprasert difficulties. First and second rows correspond to frame 47 and 174 of C1 sequence. From left to right: Original frame, Horprasert results and GrimsonGMM modified with our proposal.

In order to demonstrate its performance, we have considered a set of eight real sequences with different complex environments, representing indoors and outdoors situations. All of them can be found in public repositories. Five state-of-the-art methods have been modified with our proposal. This amounts to a total of 40 benchmarks, and our proposal overcomes the original algorithm in 35 of them, while it achieves competitive results in the other five. There are also three background subtraction and shadow detection methods that we have tested; they achieve poor results compared to our proposal.

From the above, it can inferred that our proposal increases the performance of existing background segmentation methods in a wide range of illumination change conditions, which demonstrates its potential to enhance computer vision systems which include a foreground detection subsystem.

## Acknowledgments

This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2011-24141, project name Detection of anomalous activities in video sequences by self-organizing neural systems. It is also partially supported by the Autonomous Government of Andalusia (Spain) under projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga.

## References

[1] A. Agrawal, R. Raskar, R. Chellappa, Edge suppression by gradient field transformation using cross-projection tensors, 2006 IEEE Computer Society Conference Computer Vision and Pattern Recognition2006. 2301–2308.
[2] K. Anderson, P. McOwan, Robust real-time face tracker for cluttered environments, Comput. Vis. Image Underst. 95 (2004) 184–200.
[3] M. Bales, D. Forsthoefel, B. Valentine, D. Wills, L. Wills, Bigbackground-based illumination compensation for surveillance video, Eurasip J. Image Video Process. 2011 (2011).
[4] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[5] Y. Ding, W.H. Li, J.T. Fan, H.M. Yang, Robust moving object detection under complex background, Comput. Sci. Inf. Syst. 7 (2010) 201–210.
[6] Y. Dong, G.N. DeSouza, Adaptive learning of multi-subspace for foreground detection under illumination changes, Comput. Vis. Image Underst. 115 (2011) 31–49.
[7] D. Farcas, C. Marghes, T. Bouwmans, Background subtraction via incremental maximum margin criterion: a discriminative subspace approach, Mach. Vis. Appl. 23 (2012) 1083–1101.
[8] M. Heikkila, M. Pietikainen, A texture-based method for modeling the background and detecting moving objects, Pattern Analysis and Machine Intelligence IEEE Transactions, 282006. 657–662.
[9] T. Horprasert, D. Harwood, L.S. Davis, A statistical approach for real-time robust background subtraction and shadow detection, Proceedings of the IEEE International Conference on Computer Vision 1999, pp. 1–19.
[10] J.S. Hu, T.M. Su, Robust background subtraction with shadow and highlight removal for indoor surveillance, EURASIP J. Appl. Signal Process. 2007 (2007) 108.
[11] I. Huerta, A. Amato, X. Roca, J. González, Exploiting multiple cues in motion segmentation based on background subtraction, Neurocomputing 100 (2013) 183–196.
[12] Y. Ivanov, A. Bobick, J. Liu, Fast lighting independent background subtraction, Int. J. Comput. Vis. 37 (2000) 199–207.
[13] P. KaewTraKulPong, R. Rowden, An improved adaptive background mixture model for real-time tracking with shadow detection, Proceedings of the Second European Workshop on Advanced Video Based Surveillance Systems 2001, pp. 149–158.
[14] J. Li, Z. Miao, Foreground segmentation for dynamic scenes with sudden illumination changes, Image Process.,IET 6 (2012) 606–615.
[15] L. Li, I.H. Gu, M. Leung, Q. Tian, Adaptive background subtraction based on feedback from fuzzy classification, Opt. Eng. 43 (2004) 2381–2394.
[16] L. Liu, N. Sang, R. Huang, Background subtraction using shape and colour information, Electron. Lett. 46 (2010) 41–43.

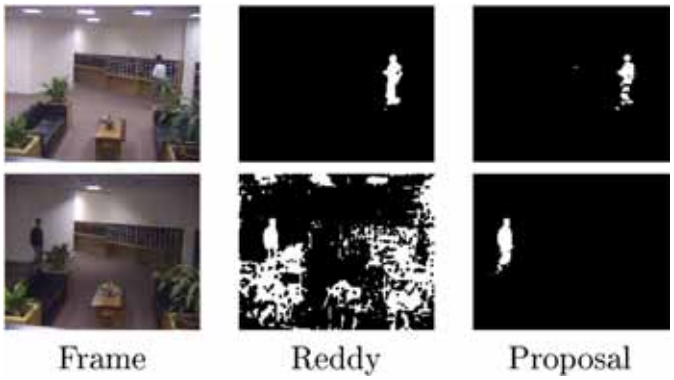



**Fig. 14.** Examples to illustrate typical Reddy difficulties. First and second rows correspond to frame 1163 and 1630 of LB sequence. From left to right: Original frame, Reddy results and GrimsonGMM modified with our proposal.

[17] X. Lu, T. Izumi, L. Teng, T. Horie, L. Wang, A novel background subtraction method for moving vehicle detection, IEEJ Trans. Fundam. Mater. 132 (2012) 857–863.
[18] L. Maddalena, A. Petrosino, A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection, Neural Comput. & Applic. 19 (2010) 179–186.
[19] Y.F. Mao, P.F. Shi, Gaussian kernel density estimation-based background modeling with noise and shadow suppression Xitong Fangzhen Xuebao, J. Syst. Simul. 17 (2005) 1182–1184.
[20] A. Prati, I. Mikic, M. Trivedi, R. Cucchiara, Detecting moving shadows: algorithms and evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2003) 918–923.
[21] V. Reddy, C. Sanderson, B. Lovell, Improved foreground detection via block-based classifier cascade with probabilistic decision integration, Circuits and Systems for Video Technology, IEEE Transactions on 232013. 83–93, http://dx.doi.org/10.1109/TCSVT.2012.2203199.
[22] V. Reddy, C. Sanderson, A. Sanin, B. Lovell, Adaptive patch-based background modelling for improved foreground object segmentation and tracking, Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference2010. 172–179.
[23] A. Shimada, R.I. Taniguchi, Hybrid background modeling for long-term and short-term illumination changes, IEEJ Transactions on Electronics, Information and Systems1302010. (1524–1529 + 4).
[24] M. Sivabalakrishnan, D. Manjula, Performance analysis of fuzzy logic-based background subtraction in dynamic environments, Imaging Sci. J. 60 (2012) 39–46.
[25] M. Sokolova, N. Japkowicz, S. Szpakowicz, Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation, in: A. Sattar, B.h. Kang (Eds.), AI 2006: Advances in Artificial Intelligence, volume 4304 of, Lecture Notes in Computer ScienceSpringer Berlin Heidelberg 2006, pp. 1015–1021.
[26] C. Stauffer, W. Grimson, Adaptive background mixture models for real-time tracking, Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition 1999, pp. 246–252.
[27] D. Toth, T. Aach, V. Metzler, X, B.E., Illumination-invariant change detection, in IEEE Southwest Symposium on Image Analysis and Interpretation 2000, pp. 3–7.
[29] J. Van Es, T. Vladusich, F. Cornelissen, Local and relational judgements of surface colour: constancy indices and discrimination performance, Spat. Vis. 20 (2007) 139–154.
[30] C. Wren, A. Azarbayejani, T. Darrell, A. Pentl, Pfinder: real-time tracking of the human body, IEEE Trans. Pattern Anal. Mach. Intell. 19 (1997) 780–785.
[31] Y. Xiangdong, Y. Jie, W. Na, Removal of disturbance of sudden illumination change based on color gradient fusion Gaussian model, Int. J. Adv. Comput. Technol. 5 (2013) 86–92.
[32] B. Xie, V. Ramesh, T. Boult, Sudden illumination change detection using order consistency, Image Vis. Comput. 22 (2004) 117–125.
[33] X.Q. Yu, X.N. Chen, M.Y. Jiang, Detection of moving object in moving background based on feature vector field fuzzy segmentation and OTSU method, Opto-Electron. Eng. 39 (2012) 94–102.
[34] Z. Zivkovic, Improved adaptive gaussian mixture model for background subtraction, Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2–Volume 02, IEEE Computer Society, Washington, DC, USA 2004, pp. 28–31.

Pattern Recognition Letters 68 (2015) 161–168

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# Foreground detection for moving cameras with stochastic approximation☆

Francisco Javier López-Rubio, Ezequiel López-Rubio*

*Department of Computer Languages and Computer Science, University of Málaga, Bulevar Louis Pasteur 35, Málaga 29071, Spain*

ARTICLE INFO

*Article history:*
Received 5 May 2015
Available online 11 November 2015

*Keywords:*
Background modeling
Moving camera
Covariance matrix interpolation
Stochastic approximation

ABSTRACT

Most foreground detection algorithms do not perform well with pan-tilt-zoom (PTZ) cameras for video surveillance and static cameras that experience vibration, since they rely on the assumption that the background does not move. Here a novel approach based on stochastic approximation learning of probabilistic mixtures is proposed. It assumes that the camera can zoom and move in both horizontal and vertical planes, and it is also adequate for egomotion sequences without abrupt changes. In other words it is a non panoramic model for moving cameras, where the camera movement allows to reuse enough background information from the previous frame. Two pixel models are used, one to follow the camera movement and the other to detect foreground objects. A procedure is developed to transform and interpolate the covariance matrices of the Gaussian mixture components as the camera moves and zooms. Moreover, a background extrapolation method is presented in order to generate new mixture models for previously unseen regions. The proposal is compared with some state-of-the-art alternatives, with competitive results both quantitatively and qualitatively.



## 1. Introduction

The vast majority of foreground detection algorithms build background models and identify foreground pixels or regions because their features are different from those of the background, but they are based on the assumption that the background does not change significantly with time or at least does not move. However in real situations this condition is rarely met even for static cameras due to camera shake, which can be produced by wind or other external factors. This kind of problem is called jitter and is very common, e.g. in street or highway surveillance cameras. Moreover, with the spread of video surveillance technology, it is increasingly common to find moving cameras of the PTZ kind (pan, tilt and zoom) covering a larger area than static cameras. Finally, foreground detection algorithms are also necessary for hand-held cameras that allow free movement. Again the usual background segmentation methods for static cameras are not capable of dealing with these situations.

There are several proposals to address background segmentation of a scene recorded by a moving camera. A joint representation of pixel color and spatial structures is used in [21] to build background and foreground models. This idea is extended in [9] by finding trajectories of key points in the scene, which requires the analysis of several consecutive frames.

Some proposals are based on building a panoramic model of the scene [2,5]. That is, they put each frame in the proper position of the model of the scene, and then foreground objects are detected using a traditional segmentation approach for static cameras. Panoramic methods can only be applied to cameras whose movement is restricted to a scene of fixed size, such as those used to broadcast sports events.

Non-panoramic methods like Kim et al. [13] are suitable for free moving cameras, since they only store a background model of the current frame. Their main challenge is how to recycle the information from the model of the current frame to build the model of the next frame. This is commonly done by estimating the camera motion, which usually involves the search for corresponding keypoints in the current and next frames [23].

Here a non panoramic method for foreground detection with a single camera is proposed. It overcomes the limitations of previous algorithms such as in [13] by employing a more realistic background model for the pixels, based on our previous works [15,17]. This requires new ways to pass the background information from the current frame to the next frame, as we will see. Moreover, our method does not require any knowledge of the internal camera parameters. Hence it is applicable to a wide range of situations. Nevertheless, it is assumed that there is a substantial fraction of the background model

☆ This paper has been recommended for acceptance by R. Davies.

* Corresponding author. Tel.: +34 95 213 71 55.
*E-mail addresses:* xavierprof@hotmail.com (F.J. López-Rubio), ezeqlr@lcc.uma.es, elr@uma.es (E. López-Rubio).
*URL:* http://www.lcc.uma.es/%7Eezeqlr/index-en.html (E. López-Rubio)

http://dx.doi.org/10.1016/j.patrec.2015.09.007
0167-8655/

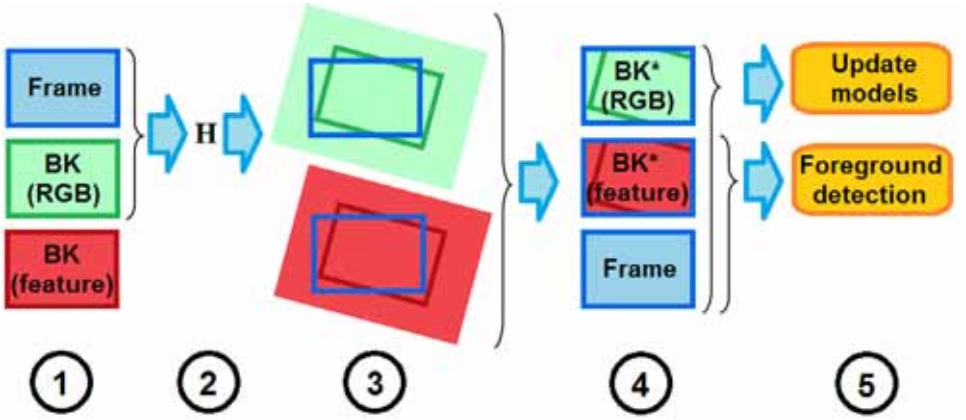



**Fig. 1.** Flowchart of our proposal. 'BK(RGB)' stands for the first background model and 'BK(features)' stands for the second background model. In the first step we have the input frame to be processed and the current state of the algorithm represented by 'BK(RGB)' and 'BK(features)'. Step two estimates the transformation matrix **H** using the input frame and 'BK (RGB)'. In the third step the background models are projected using **H**, which leaves a previously unseen region $\mathcal{U}$. The fourth step determines the area corresponding to the current frame in the projected models. In the fifth and last step 'BK*(feature)' is used to perform the foreground detection and both models are updated.

previous frame which can be reused for the current frame, so very fast camera movements are not managed. Consequently it is best suited for PTZ cameras, jitter compensation, and egomotion without abrupt background changes.

The main novelties of this work with respect to [17] and previous literature are twofold:

1. On one hand, a procedure to interpolate the full covariance matrices of the pixel models is proposed (Section 3.3). To the best of our knowledge, full covariance matrix interpolation has not been researched in background modeling literature.
2. On the other hand, following the camera movement and detecting foreground objects are distinct problems with their own complexities. Here we propose to have two pixel models, one for each task so that each of them can be tuned to their specific goals (Section 2).

This paper is organized as follows. Section 2 outlines the proposed foreground detection method, which is presented in detail in Section 3. Section 4 presents some experimental results to demonstrate the ability of our approach to manage complex scenes. The main features and properties of our proposal are discussed in Section 5. Finally, Section 6 is devoted to conclusions.

## 2. Overview

Our system considers two probabilistic background models for each pixel at each frame:

- The first model is aimed to match the key points of the scene between two successive frames. Its background features are the plain RGB values. The probability density function of the RGB values at the pixel with frame coordinates $\mathbf{x}$ is noted $p_{\mathbf{x},RGB}$.
- The second model is used to classify the pixel into background or foreground. A set of background features $\mathcal{F}$ must be chosen, as seen in Section 3.1. The probability density function of the selected background features at the pixel with frame coordinates $\mathbf{x}$ is noted $p_{\mathbf{x},\mathcal{F}}$.

Both background models have the same mathematical structure, so the model definition and the learning procedure described in Section 3 applies to both of them equally.

Each time that a new frame arrives, it is necessary to estimate the transformation matrix **H** which transforms from the old pixel coordinate system to the new pixel coordinate system. Then new background models for its pixels are built using the information from the models of the previous frame (Section 3.3). That is, no panorama is maintained. The interpolation procedure to obtain the parameters for the new background models includes the use of correlation matrices in order to produce accurate approximations of the covariance matrices of the Gaussian mixture components.

Some of the background models of the previously unseen regions of the new frame can be initialized with the help of the models of the pixels near the borders of the old frame. A Bayesian decision is carried out to determine whether a previously unseen pixel is suitable to be initialized in this way (Section 3.4). Otherwise, a neutral initialization is used.

## 3. Methodology

In this section a model is proposed to address the foreground detection problem for mobile cameras. It is based on our earlier models [15,17], which were oriented to fixed cameras. The probabilistic mixture model is defined in Section 3.1, and the features are specified in Section 3.2. The camera motion compensation mechanism is described in Section 3.3, and finally the background extrapolation procedure is given in Section 3.4. Fig. 1 shows the flowchart of our proposal for a frame.

### 3.1. Mixture model

The model is based on a stochastic approximation algorithm which is used to train probabilistic mixtures which model distributions of pixel feature values $\mathbf{t}(\mathbf{x}) \in \mathbb{R}^D$ at frame coordinates $\mathbf{x} = (j, k)$, where $D$ is the dimension of the pixel feature vector. The set of features of interest will be noted $\mathcal{F}$. There is one Gaussian component $p(\mathbf{t}|Back)$ for the background and one uniform component $p(\mathbf{t}|Fore)$ for the foreground. It has been found that a single Gaussian with a full covariance matrix is flexible enough for most background distributions, while it speeds up the operation. On the other hand, the uniform distribution for the foreground models any incoming object equally well, no matter how unexpected their characteristics are. In contexts where some degree of redundancy in the foreground objects is expected, other foreground models could be used [6]. These considerations lead to the following probabilistic model $p(\mathbf{t})$ for the distribution of feature values at any given position $\mathbf{x}$ for the feature set $\mathcal{F}$, where we drop $\mathbf{x}$ and $\mathcal{F}$ for the sake of simplicity:

$$\begin{aligned} p(\mathbf{t}) &= \pi_{Back} p(\mathbf{t}|Back) + \pi_{Fore} p(\mathbf{t}|Fore) \\ &= \pi_{Back} K(\mathbf{t}|\boldsymbol{\mu}_{Back}, \mathbf{C}_{Back} + \boldsymbol{\Psi}) + \pi_{Fore} U(\mathbf{t}) \end{aligned} \quad (1)$$

$$K(\mathbf{t}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-D/2} \det(\boldsymbol{\Sigma})^{-1/2} \exp\left((\mathbf{t}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{t}-\boldsymbol{\mu})\right) \quad (2)$$

$$\mathbf{\Sigma} = \mathbf{C}_{Back} + \mathbf{\Psi} \tag{3}$$

$$\forall i \in \{Back, Fore\},\ \boldsymbol{\mu}_i = E[\mathbf{t}|i] \tag{4}$$

$$\mathbf{C}_{Back} = E[(\mathbf{t}-\boldsymbol{\mu}_{Back})(\mathbf{t}-\boldsymbol{\mu}_{Back})^T|Back] \tag{5}$$

$$U(\mathbf{t}) = \begin{cases} 1/Vol(\mathcal{S}) & \text{iff } \mathbf{t} \in \mathcal{S} \\ 0 & \text{iff } \mathbf{t} \notin \mathcal{S} \end{cases} \tag{6}$$

where $\mathcal{S}$ stands for the support of the input distribution, $Vol(\mathcal{S})$ is the $D$-dimensional volume of $\mathcal{S}$, and $\mathbf{\Psi}$ is a diagonal regularization matrix with strictly positive diagonal elements which ensures that $\mathbf{\Sigma}$ is not singular. The Gaussian mixture component $K(\mathbf{t}|\boldsymbol{\mu}, \mathbf{\Sigma})$ is able to adapt to many kinds of dynamic backgrounds because $\mathbf{C}_{Back}$ (and hence $\mathbf{C}_{Back} + \mathbf{\Psi}$) are not restricted to be diagonal matrices. Moreover, all types of foreground objects are modeled equally well by the uniform mixture component $U(\mathbf{t})$.

The Robbins–Monro stochastic approximation algorithm with constant step size $\epsilon$ is applied to the previous probabilistic mixture model to produce the update equations at time step $n$ for an observed pixel value $\mathbf{t}_n$; see [17] for details.

In the case of moving cameras, it has been found to be advantageous to have two mixture models (1) for each pixel $\mathbf{x}$. The first one, that we note $p_{\mathbf{x},RGB}$, is in the RGB space and it is used to estimate the camera motion. The second one, that we note $p_{\mathbf{x},\mathcal{F}}$, employs specifically tailored features (see Section 4.3) and it is used to estimate the probability that a pixel belongs to the foreground. This is because we have found by experimentation that RGB offers good results in terms of feature matching between successive frames, while other features perform better in terms of foreground pixel detection. Therefore, the class probabilities of the observed value $\mathbf{t}_n$ of a pixel $\mathbf{x}$ at time step $n$ are given by Bayes' theorem:

$$\forall i \in \{Back, Fore\},\ R_{\mathbf{x},n,i} = P_{\mathbf{x},\mathcal{F}}(i|\mathbf{t}_n) = \frac{\pi_{i,\mathbf{x},\mathcal{F}} p_{\mathbf{x},\mathcal{F}}(\mathbf{t}_n|i)}{p_{\mathbf{x},\mathcal{F}}(\mathbf{t}_n)}$$
$$= \frac{\pi_{i,\mathbf{x},\mathcal{F}} p_{\mathbf{x},\mathcal{F}}(\mathbf{t}_n|i)}{\pi_{Back,\mathbf{x},\mathcal{F}} p_{\mathbf{x},\mathcal{F}}(\mathbf{t}_n|Back) + \pi_{Fore,\mathbf{x},\mathcal{F}} p_{\mathbf{x},\mathcal{F}}(\mathbf{t}_n|Fore)} \tag{7}$$

That is, $R_{\mathbf{x},n,Back}$ and $R_{\mathbf{x},n,Fore}$ are the posterior probabilities that at time step $n$ the pixel $\mathbf{x}$ belongs to the background or the foreground, respectively.

### 3.2. Features

Since this work extends our previous model for fixed cameras in [17], we have considered the same 24 features defined therein. Given a set of selected features $\mathcal{F} \subset \{1, \ldots, 24\}$, the input pattern for the pixel at frame coordinates $(j, k)$ is given by:

$$\mathbf{t}_{j,k} = (F_{i,j,k})_{i \in \mathcal{F}} \tag{8}$$

Next, we present the set of background features we have considered. Features 15–22 were first introduced in [17], while the others are known in previous literature. We have adjusted all of them so that their range of values is the interval [0, 1]:

- The first three features are the red, green and blue channels of the input frame (features 1–3). Here we assume that the pixel values are real numbers in the interval [0, 1].
- The normalized RGB channels are also used (features 4–6) due to their robustness with respect to illumination changes [22].
- Next we consider the six Haar-like features of size 9 × 9 pixels considered in [12], which are weak features that convey texture information (features 7–12).
- Gradient features are also useful because they are less affected by illumination changes, while they provide local texture information [14]. We estimate the gradient components with Sobel operator (features 13–14).
- Next we introduce two features which consider both color information and a pixel adjacent to the pixel at hand, as a local indication of color texture. The first one (feature 15) is the red channel of the current pixel, normalized with the current pixel and the pixel immediately to the left of the current one. The second one (feature 16) is the green channel of the pixel immediately to the lower right of the current one, normalized with the current pixel.
- In order to include some robust features, we add versions of features 1–6 processed with the bidimensional median filter of window size 5 × 5 pixels (features 17–22). The filter size has been chosen as a balance between the robustness (which needs larger sizes) and the precision of the object detection (which calls for a smaller size).
- Finally, we use two small filters of size 3 × 3 pixels to extract the texture information from the close vicinity of the pixel at hand, namely $\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$, which are features 23 and 24 respectively.

### 3.3. Camera motion compensation

In order to estimate the camera motion from one frame to another a 8-parameter projective motion model is considered, as commonly done in literature [1,13]:

$$\tilde{\mathbf{x}}(n) = \mathbf{H}(n-1)\tilde{\mathbf{x}}(n-1) \tag{9}$$

where $\tilde{\mathbf{x}}(n) = (j_n, k_n, 1)$ and $\tilde{\mathbf{x}}(n-1) = (j_{n-1}, k_{n-1}, 1)$ are the pixel affine coordinates of a certain object or feature at time steps $n$ and $n-1$, respectively, and $\mathbf{H}$ is a 3 × 3 real matrix which transforms from the pixel coordinate system at time $n-1$ to the pixel coordinate system at time $n$. The matrix $\mathbf{H}$ is estimated by extracting and matching features by the SURF transform and then obtaining a consensus value by mean of the RANSAC algorithm. These procedures are standard in camera motion compensation [19,24].

Now the problem is how to transfer the information from the pixel models at time step $n$ to those at time step $n+1$, both for $p_{\mathbf{x},RGB}$ and $p_{\mathbf{x},\mathcal{F}}$. The covariance matrix $\mathbf{C} = E[(\mathbf{t}-\boldsymbol{\mu})(\mathbf{t}-\boldsymbol{\mu})^T]$ cannot be linearly interpolated in a straightforward way because their elements are relative to the mean vector $\boldsymbol{\mu}$, and the mean vector varies from one pixel to another, which makes the estimation seriously unreliable [16]. The solution that we propose is to linearly interpolate the correlation matrices $\mathbf{R} = E[\mathbf{t}\mathbf{t}^T]$, also called uncentered covariance matrices. Then the interpolated covariance matrix $\hat{\mathbf{C}}$ can be obtained instead of $\hat{\mathbf{R}}$ as follows:

$$\hat{\mathbf{C}} = \hat{\mathbf{R}} - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^T \tag{10}$$

which expresses the interpolated covariance matrix in terms of the interpolated correlation matrix $\hat{\mathbf{C}}$ and the interpolated mean vector $\hat{\boldsymbol{\mu}}$.

To prevent erroneous transformations that do not correspond to reality, two types of errors are defined: minor errors and severe errors. Minor errors occur when the size of the model before the projection is either too large or too small compared to the projected model. Let $L$ and $W$ be the height and width of the projected model frame, respectively. Also, let $l$ and $w$ be the height and width of the original model. A minor error occurs if the following condition is satisfied:

$$(\lambda_1 l < L < \lambda_2 l) \vee (\lambda_1 w < W < \lambda_2 w) \tag{11}$$

where $\lambda_2 < \lambda_1$. Experiments have proven that $\lambda_1 = 2$ and $\lambda_2 = 0.6$ are suitable values for the tested videos.

When a minor error occurs, a new matrix $\mathbf{H}$ is generated using different features. If there are 10 consecutive minor errors it is considered a severe error. When a severe error occurs, the current frame is skipped and no models are changed. If there are 3 consecutive severe errors both models are reset to the value of the current frame.

### 3.4. Background extrapolation

As the camera moves, previously unseen parts of the scene fall inside the field of view, while other parts cease to be visible. This means that a procedure is needed to generate new mixture models for the pixels corresponding to the previously unseen region $\mathcal{U}$. Our approach does the initialization depending on the likelihood that a pixel $\mathbf{x} \in \mathcal{U}$ has similar characteristics to those in the borders of the already known region $\mathcal{K}$. To this end, the pixel $\mathbf{y} \in \mathcal{K}$ which is closest to $\mathbf{x}$ is checked to see whether the probability to belong to the background of the observed feature vector $\mathbf{t}(\mathbf{x})$ at pixel $\mathbf{x}$ under the mixture model of pixel $\mathbf{y}$ is higher than certain classification threshold $\tau \in [0, 1]$:

$$P_{\mathbf{y}}(Back|\mathbf{t}(\mathbf{x})) > \tau \tag{12}$$

The use of tunable classification thresholds is a common technique in machine learning [3]. If condition (12) holds, then the already trained mixture model of pixel $\mathbf{y}$ is copied to pixel $\mathbf{x}$, since it is likely that pixel $\mathbf{x}$ is a portion of the scene background whose characteristics are similar to those of pixel $\mathbf{y}$. Otherwise, it is assumed that pixel $\mathbf{x}$ is either foreground or a part of the background which does not match to the nearest border pixel of $\mathcal{K}$, and it is initialized to a neutral state. If the camera does not move too fast or the background is homogeneous, it can be expected that many pixels satisfy condition (12) so that a significant amount of information is carried over to the next frame. Otherwise the new background is completely new, so there is no way to reuse the previous background models.

## 4. Experimental results

In this section an analysis of our approach in different situations is carried out[1]. The same tests have been repeated for some state-of-the-art competing methods. The structure of this section is as follows. First the competing methods are presented (Section 4.1). Then the set of sequences which has been used to perform the tests is described (Section 4.2). Next the quantitative performance measures that we have chosen to compare the methods and the selected parameters for the competing methods and our approach are considered (Section 4.3). Finally, Sections 4.4 and 4.5 are devoted to report the results in a qualitative and quantitative way, respectively.

### 4.1. Methods

We compare our proposal against four methods: ViBe [4], a non-panoramic method we call NP-Kim [13], and two optical-flow based methods we call FVS [20] and MoSeg [18]. The authors of ViBe and NP-Kim have collaborated with us. In the first case they provided an executable version of ViBe. In the NP-Kim case, the authors sent us the results of their method for the tested sequences. Executable versions and demos of the optical-flow methods are available on the Web: FVS[2] and MoSeg[3].

The implementation of our approach has been performed using Matlab in combination with MEX files, and no additional post-processing has been done. We used the OpenCV 2.4.8 and Threading Building Blocks 4.2 libraries. Both of them are freely available on the Internet.

### 4.2. Sequences

There is a specific problem with cameras that suffer from jitter effects, namely relatively small but abrupt horizontal and vertical camera movements. This behavior is often encountered in outdoor surveillance cameras, e.g. for the surveillance of streets or highways. In order to study this type of situation we use the following sequences: *traffic*, *sidewalk*, *badminton* and *boulevard*, all of them well known in the background segmentation field and available on the Web[4] [11].

Nevertheless, our proposal is aimed to address a wider range of problems such as pan-tilt and zoom using PTZ and hand-held cameras. Therefore we chose the following set of additional sequences: *Campus1* and *campus2*, which are named user19_2 and bren_user07-02 in the UC Santa Barbara tracking repository [8] and they are available in the Web[5]; *campus3* which is named seqD in the BoBoT – Bonn Benchmark on Tracking[6] [10]; *tennis* from Freiburg–Berkeley Motion Segmentation Dataset[7] [7]; and *woman* is a panning video example downloaded from YouTube and it is available in the Web[8].

The ground truth provided in the original repository was used in the case of the jitter set. On the other hand, for the second set of sequences we have carried out the segmentation manually because there is no publicly available ground truth. To be as fair as possible, we used a constant sampling rate in order to determine which frames would be used as ground truth. Manually segmented ground truth are available in the Web[9].

### 4.3. Performance measures and parameter selection

In order to carry out a quantitative assessment of the methods we have used six widely known measures: false negatives rate (FNR), false positives rate (FPR), precision (PR), recall (RC), accuracy (AC) and F-measure (FM). Lower is better for FNR and FPR, while higher is better for PR, RC, AC and FM. To calculate these measures it is necessary to count false negatives (FN), false positives (FP), true negatives (TN) and true positives (TP). In addition we compute the shadow error (SE):

$$\mathrm{FN} = \mathrm{card}(A \cap \overline{B}) \quad \mathrm{FP} = \mathrm{card}((\overline{A} \cup S) \cap B)$$
$$\mathrm{TN} = \mathrm{card}((\overline{A} \cup S) \cap \overline{B}) \quad \mathrm{TP} = \mathrm{card}(A \cap B)$$
$$\mathrm{SE} = \mathrm{card}(S \cap B)$$

where 'card' stands for the number of elements of a set. $A$, $B$ and $S$ are respectively the set of all pixels which belong to the foreground, the set of all pixels which are classified as foreground by the analyzed method and the set of all pixels that are considered shadow. So that the definitions of the main performance measures are the following:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \quad \mathrm{AC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \quad \mathrm{RC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$
$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \quad \mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad \mathrm{FM} = \frac{2 \cdot \mathrm{PR} \cdot \mathrm{RC}}{\mathrm{PR} + \mathrm{RC}}$$

We must note that these measures are used only to compare the performance of the methods, so that they are not involved into the background modeling algorithm. Quantitative results are reported for AC and FM, while the other measures are intermediate quantities in the computation of AC and FM.

When choosing the values of the parameters, we have attempted that the number of tested values is similar for each parameter of the studied methods except for NP-Kim, in which case we have not tried different configurations, since the source code is not available. We have used the recommended values found both in the documentation

[1] The source code and some demos of our proposal are available at: http://www.lcc.uma.es/%7Eezeqlr/nonpan/nonpan.html

[2] http://groups.inf.ed.ac.uk/calvin/FastVideoSegmentation/

[3] http://lmb.informatik.uni-freiburg.de/resources/binaries/

[4] http://www.changedetection.net/

[5] http://www.lcc.uma.es/%7Eezeqlr/nonpan/campus1frames.zip and http://www.lcc.uma.es/%7Eezeqlr/nonpan/campus2frames.zip

[6] http://www.iai.uni-bonn.de/~kleind/tracking/

[7] http://lmb.informatik.uni-freiburg.de/resources/datasets/sequences.en.html

[8] http://www.lcc.uma.es/%7Eezeqlr/nonpan/panning_example.mp4

[9] http://www.lcc.uma.es/%7Eezeqlr/nonpan/nonpan.html

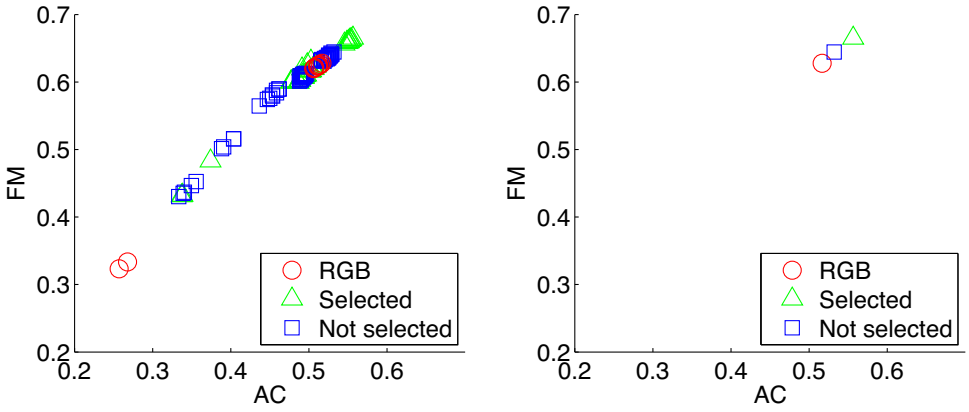



**Fig. 2.** Mean performance across all the sequences in terms of F-measure and accuracy. The plot to the left shows the mean performance of each configuration using all the combinations of $\varepsilon$ and $\tau$ shown in Section 4.3. The plot to the right shows the same measures for Pareto front configurations. Red circles represent the RGB features (i.e. {1, 2, 3}). Green triangles correspond to the chosen features (see Table 1). Blue squares correspond to feature sets considered in [17] that are not used in this paper. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

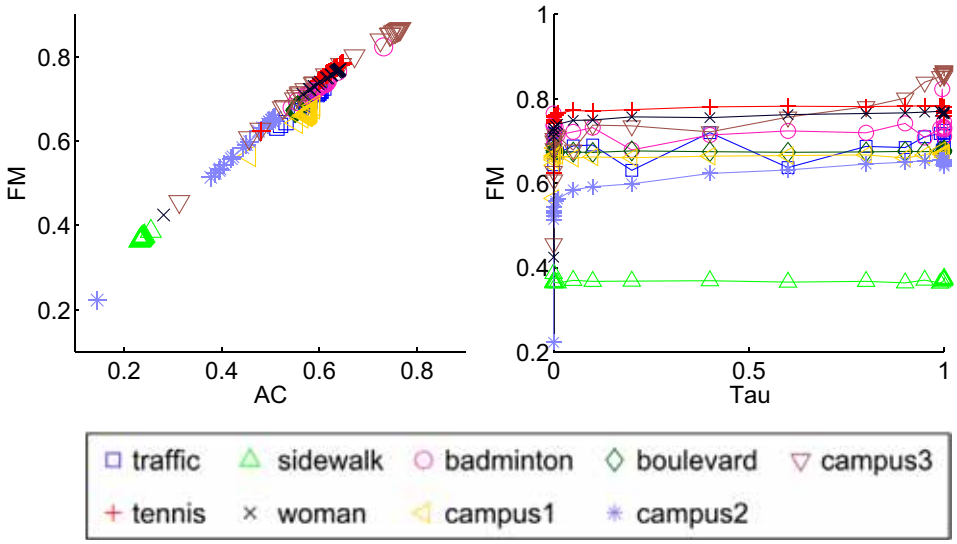



**Fig. 3.** Performance of our proposal when $\varepsilon$ and $\mathcal{F}$ are held constant at their optimal values for each sequence, and $\tau$ is varied. The first column shows the performance in terms of F-measure vs accuracy, while the second column shows the performance in terms of F-measure vs $\tau$.

**Table 1**
Chosen values for each of the configurable parameters. The combinations of these values form the set of all tested configurations.

| Method | Parameters |
|---|---|
| ViBe | Nb samples, $n = \{20, 40, 60\}$<br>Matching threshold, $t = \{10, 20, 30\}$<br>Matching number, $s = \{1, 2, 4\}$<br>Subsampling factor, $f = \{8, 16, 32\}$ |
| FVS | Fadeout, $F = \{0.0005, 0.0001, 0.00005\}$<br>Spatial weight, $w_S = \{1000, 5000, 10000, 15000, 20000\}$<br>Temporal weight, $w_T = \{500, 2000, 4000, 7000, 10000\}$ |
| MoSeg | Sampling, $S = \{4, 8, 16\}$<br>Trajectories weight, $T = \{40, 60, 80\}$ |
| Proposed | Step size, $\varepsilon = \{0.002, 0.01, 0.02, 0.03\}$<br>Threshold, $\tau = \{0.999, 0.9995, 0.9999\}$<br>Features, $\mathcal{F} = \{\begin{matrix}\{19, 20, 22\}, \{3, 20, 21, 22\}, \\ \{5, 19, 20, 21, 22\}\end{matrix}\}$ |

and in the papers. We have also chosen other values that we have found to perform well in experiments. The selected parameter values are listed in Table 1. For the proposed RGB model we only used one step size set to 0.2 because it is large enough to avoid blurring the scene.

The maximum number of iterations for the refinement step of the FVS method is 4, as the documentation recommends. However in some sequences the method ends before iteration 4 is reached due to Cholesky factorization failure caused by non positive definite matrices.

In order to test our method the three best performing sets of features found in [17] are employed. As seen in Fig. 2, RGB is outperformed by other feature sets; and the chosen selected feature sets outperform those that have been left out. The relevance of $\tau$ is shown in Fig. 3. The value of $\tau$ does not significantly modify the method performance in the majority of jitter sequences because the portion of new background is smaller compared to the other sequences. However in scenes with hand-held cameras and PTZ the value of $\tau$ is important so we use three values close to 1 that achieve good results.

### 4.4. Qualitative results

In this section a qualitative analysis of the results is carried out using Fig. 4. In addition the videos corresponding to the best results according to F-measure are available on the Web[10] where the performance of the five methods can be compared for all sequence frames.

[10] http://www.lcc.uma.es/%7Eezeqlr/nonpan/nonpan.html

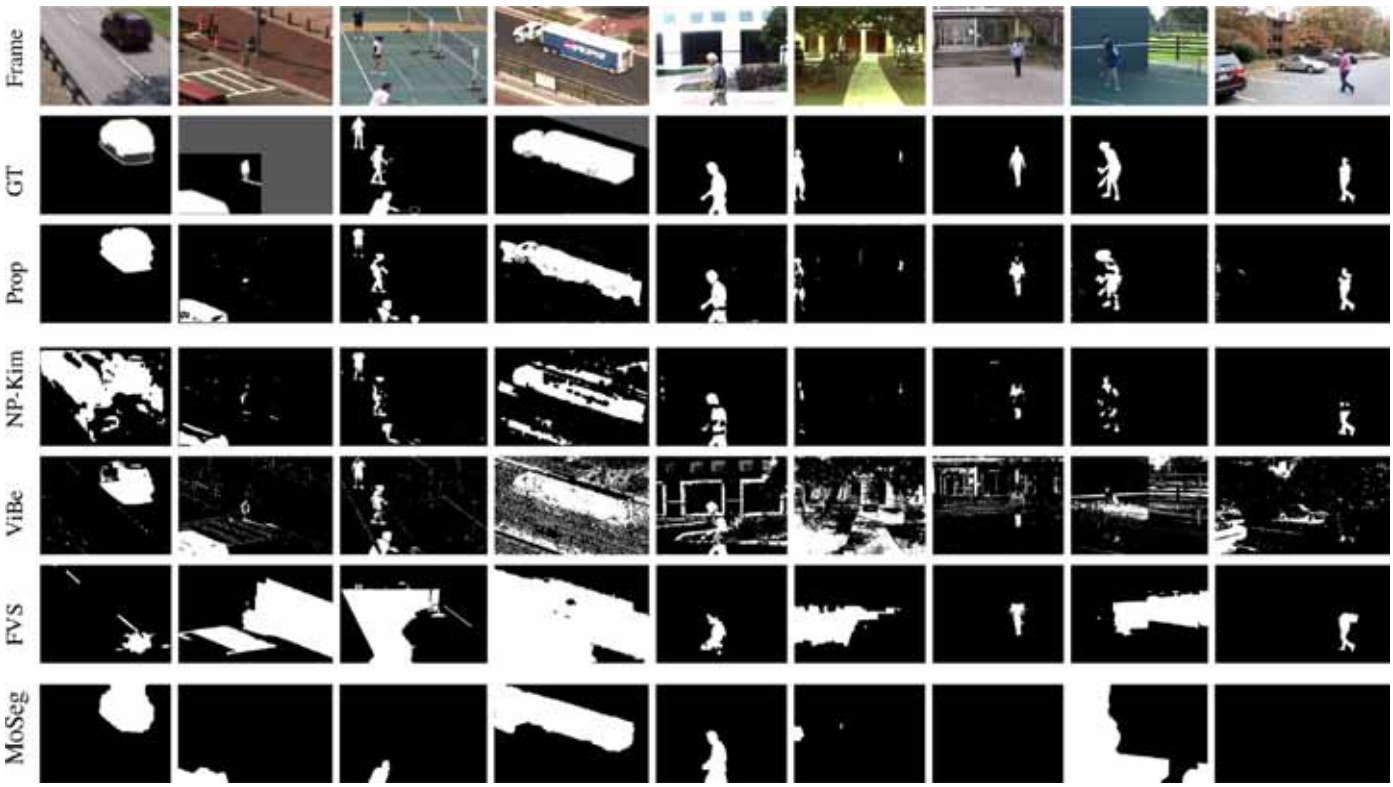



**Fig. 4.** Results for the sequences using the best configuration according to F-measure. From left to right: traffic, sidewalk, badminton, boulevard, campus1, campus2, campus3, tennis and woman. Rows from up to down: original frame, ground truth frame, proposed method, NP-Kim, ViBe, FVS and MoSeg.

Columns 1–4 of Fig. 4 depict the performance of each method in jitter sequences. NP-Kim produces large FP areas in some difficult situations (see first column of Fig. 4). However its main problem is FN, sometimes due to camouflage effects. ViBe is prone to high FP in textured edges (see columns 2 to 3 in Fig. 4). ViBe also gets high FN because it does not fill the foreground objects. FVS produces large FP areas in the majority of jitter sequences. MoSeg performs fair in traffic and boulevard. Its main difficulty in those sequences is FP, but it gets many FN in badminton and sidewalk. On the other hand our proposal seems to performs well in jitter sequences. It produces very little FP and only some FN due to camouflage effects but to a lesser extent than NP-Kim.

Columns 5–9 of Fig. 4 are devoted to PTZ and hand-held cameras. The situation is very different and the differences are larger than in the jitter case. Our proposal seems to perform much better than the other competitors in this kind of sequences. ViBe produces a very large quantity of FP and FN, so that it can be discarded as a valid choice for this kind of sequences. NP-Kim suffers an important amount of FN as in jitter situations (see columns 7–9 in Fig. 4). FVS performs fair in sequences like campus1, campus3 and woman, but it gets many FP or FN in the rest. MoSeg has an irregular performance. It correctly detects the foreground object in the campus1 sequence, but it wrongly segments the frame in the campus3 and woman sequences.

In summary our method produces some FP but compensates it by a small FN. Therefore, our proposal seems to overcome the competitor methods in the tested scenes. This is confirmed in the next section.

### 4.5. Quantitative results

In this section the best methods from the quantitative point of view for each of the sequences are reported. Table 2 lists the mean values of the best performing configurations.

As said in Section 4.2, two types of videos have been chosen. For the first type, i.e. those with jitter, it is difficult to say which method is the best one, because both NP-Kim and our proposal get the best results in two of the four benchmarks (Table 2). Nevertheless we must note that our proposal is the only alternative that gets best results in half of the videos and in the other two cases it remains in second and third place, which makes it a really competitive alternative.

Table 3 makes a comparison of the number of errors caused by shadows (SE) in the two videos whose ground truth provide information about shadows. The FVS method is the best method in terms of SE, even if it performs poorly so it does not detect some moving objects and shadows (see demos in the Web[11]). Hence the FM and AC measures provide a better view of the overall performance of the methods. As seen in Section 4.3 both measures consider errors due to shadows.

The quantitative analysis of the second type of sequences, i.e. PTZ and hand-held cameras, produces a clearer picture than in the jitter case. As shown in Table 2, the performance of our proposal exceeds by six or even more FM points those of its competitors. Notably, ViBe is the method with the worst results and it is unable to achieve an adequate performance in any of these sequences. FVS and MoSeg have an unstable performance, they get poor results in half of the sequences. The overall conclusion is that our proposal is the best choice for all tested sequences with PTZ and hand-held cameras.

## 5. Discussion

In this section the most important features of the proposal are discussed:

- The probabilistic background model is more realistic than previous non panoramic approaches, since it contains a full covariance

[11] http://www.lcc.uma.es/%7Eezeqlr/nonpan/nonpan.html

**Table 2**
Results of the best configuration according to F-measure. The first and fourth columns denote the sequence name, the second and fifth columns indicate the method name and the third and sixth columns show the mean F-measure. The best method for each video is shown in **bold**.

| Sequence | Method | F-measure | Sequence | Method | F-measure |
|---|---|---|---|---|---|
| Traffic | Proposed | **0.71 ± 0.25** | Campus2 | Proposed | **0.66 ± 0.17** |
| | NP-Kim | 0.57 ± 0.26 | | NP-Kim | 0.53 ± 0.21 |
| | ViBe | 0.68 ± 0.18 | | ViBe | 0.08 ± 0.05 |
| | FVS | 0.32 ± 0.34 | | FVS | 0.24 ± 0.25 |
| | MoSeg | 0.47 ± 0.41 | | MoSeg | 0.19 ± 0.33 |
| Sidewalk | Proposed | 0.37 ± 0.14 | Campus3 | Proposed | **0.87 ± 0.04** |
| | NP-Kim | **0.45 ± 0.20** | | NP-Kim | 0.78 ± 0.08 |
| | ViBe | 0.34 ± 0.21 | | ViBe | 0.27 ± 0.09 |
| | FVS | 0.17 ± 0.12 | | FVS | 0.81 ± 0.06 |
| | MoSeg | 0.06 ± 0.22 | | MoSeg | 0.43 ± 0.37 |
| Badminton | Proposed | **0.81 ± 0.17** | Tennis | Proposed | **0.78 ± 0.05** |
| | NP-Kim | 0.68 ± 0.21 | | NP-Kim | 0.67 ± 0.12 |
| | ViBe | 0.80 ± 0.16 | | ViBe | 0.21 ± 0.05 |
| | FVS | 0.14 ± 0.08 | | FVS | 0.31 ± 0.16 |
| | MoSeg | 0.22 ± 0.26 | | MoSeg | 0.23 ± 0.26 |
| Boulevard | Proposed | 0.68 ± 0.24 | Woman | Proposed | **0.77 ± 0.14** |
| | NP-Kim | **0.78 ± 0.18** | | NP-Kim | 0.70 ± 0.15 |
| | ViBe | 0.65 ± 0.19 | | ViBe | 0.10 ± 0.12 |
| | FVS | 0.14 ± 0.11 | | FVS | 0.66 ± 0.24 |
| | MoSeg | 0.76 ± 0.32 | | MoSeg | 0.15 ± 0.23 |
| Campus1 | Proposed | **0.68 ± 0.33** | | | |
| | NP-Kim | 0.56 ± 0.32 | | | |
| | ViBe | 0.23 ± 0.15 | | | |
| | FVS | 0.46 ± 0.35 | | | |
| | MoSeg | 0.59 ± 0.40 | | | |

**Table 3**
Shadow error (SE) of the best configuration according to F-measure. The first column denotes the sequence name. Columns 2–6 correspond to the SE achieved by each method.

| Sequence | Proposed | NP-Kim | ViBe | FVS | MoSeg |
|---|---|---|---|---|---|
| Traffic | 603408 | 603277 | 585806 | **301097** | 387451 |
| Sidewalk | 4094 | 2803 | 8928 | **19** | 1482 |

Gaussian for the background and a uniform distribution for the foreground (see Section 3.1). For example, the approach in [13] considers a spherical covariance Gaussian, i.e. all the input components share the same variance, and no correlation among the input components is modeled. This is far from optimal, since the input components are almost always correlated.

- Following the camera movement and detecting foreground objects are distinct problems with their own complexities. In this work we propose to have two pixel models, one for each task so that each of them can be tuned to their specific goals (Section 2). Two background models for moving cameras have already been proposed in [25], but in that proposal the aim is to use one of the models as the current foreground detector and the other one as a backup to replace the first one when it gets corrupted. So there is no division of labor, which is what we advocate here.
- The procedure to transform and interpolate the covariance matrices of the background model under the camera motion (Section 3.3) enables the use of a full covariance Gaussian. This is a key novelty of the method, since we have experimentally checked that a straightforward interpolation of the elements of the covariance matrices leads to extremely bad performance. The ideas presented here can be easily extended to other background models with full covariance matrices.
- The previously unseen regions which are similar to the already seen part of the scene are well modeled by background extrapolation (Section 3.4). This procedure is particularly important for a non panoramic model, since these models are aimed to adapt to unseen regions most of the time.
- The most outstanding results of our proposal are those achieved in videos with PTZ and hand-held cameras, where its competitors perform poorly. It is also the most robust method in situations with jitter.

## 6. Conclusions

A non-panoramic method to detect foreground objects for moving cameras has been proposed. It is based on the stochastic approximation framework and includes two background models for each pixel, one to follow the camera movement and the other to detect foreground objects. These background models consider full covariance Gaussians, being more realistic than previous approaches. A way to transform and interpolate the covariance matrices under camera motion has been developed, and a procedure to extrapolate the information in the current model to previously unseen regions has been considered. Both of them could be extended to other background models. A total of nine benchmarks have been carried out to compare our proposal with four state-of-the-art methods. The chosen video sequences represent real situations, including situations with jitter, PTZ cameras and hand-held cameras without abrupt background changes. In seven cases our proposal is the best performing, while in the other two cases obtains the second place and third place respectively. Consequently, our proposal has demonstrated its potential as a useful alternative to previous methods.

## Acknowledgments

This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2014-53465-R, project name Video surveillance by active search of anomalous events. It is also partially supported by the Autonomous Government of Andalusia (Spain) under projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators

for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga. They are also grateful to the authors of the ViBe method for providing an executable version, and to the authors of the NP-Kim method for having supplied the output of their method for the tested sequences.

## References

[1] S. Amri, W. Barhoumi, E. Zagrouba, A robust framework for joint background/foreground segmentation of complex video scenes filmed with freely moving camera, Multimed. Tools Appl. 46 (2–3) (2010) 175–205.
[2] P. Azzari, A. Bevilacqua, Joint spatial and tonal mosaic alignment for motion detection with PTZ camera, in: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition, Lecture Notes in Computer Science, vol. 4142, Springer Berlin Heidelberg, 2006, pp. 764–775.
[3] S. Bano, A. Cavallaro, Discovery and organization of multi-camera user-generated videos of the same event, Inf. Sci. 302 (2015) 108–121.
[4] O. Barnich, M. Van Droogenbroeck, Vibe: a powerful random technique to estimate the background in video sequences, in: Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2009, pp. 945–948.
[5] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
[6] C. Benedek, T. Sziranyi, Bayesian foreground and shadow detection in uncertain frame rate surveillance videos, IEEE Trans. Image Process. 17 (4) (2008) 608–621.
[7] T. Brox, J. Malik, Large displacement optical flow: descriptor matching in variational motion estimation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 500–513.
[8] C. Coffin, J. Ventura, T. Höllerer, A repository for the evaluation of image-based orientation tracking solutions, in: Proceedings of the 2nd International Workshop on AR/MR Registration, Tracking and Benchmarking (TrakMark 2011), in conjunction with ISMAR, 2011.
[9] A. Elqursh, A. Elgammal, Online moving camera background subtraction, in: A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid (Eds.), Computer Vision ECCV 2012, Lecture Notes in Computer Science, vol. 7577, Springer Berlin Heidelberg, 2012, pp. 228–241.
[10] G.M. García, D.A. Klein, J. Stückler, S. Frintrop, A.B. Cremers, Adaptive multi-cue 3D tracking of arbitrary objects, in: Pattern Recognition, Springer, Berlin Heidelberg, 2012.
[11] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, P. Ishwar, Changedetection.net: a new change detection benchmark dataset, in: Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2012, pp. 1–11.
[12] B. Han, L.S. Davis, Density-based multifeature background subtraction with support vector machine, IEEE Trans. Pattern Anal. Mach. Intell. 34 (5) (2012) 1017–1023.
[13] S.W. Kim, K. Yun, K.M. Yi, S.J. Kim, J.Y. Choi, Detection of moving objects with a moving camera using non-panoramic background model, Mach. Vis. Appl. 24 (3) (2013) 1015–1028.
[14] L. Li, W. Huang, I.Y.-H. Gu, Q. Tian, Statistical modeling of complex backgrounds for foreground object detection, IEEE Trans. Image Process. 13 (11) (2004) 1459–1472.
[15] E. López-Rubio, R.M. Luque-Baena, Stochastic approximation for background modelling, Comput. Vis. Image Underst. 115 (6) (2011) 735–749.
[16] E. López-Rubio, J.M. Ortiz-de Lazcano-Lobato, Soft clustering for nonparametric probability density function estimation, Pattern Recognit. Lett. 29 (2008) 2085–2091.
[17] F.J. López-Rubio, E. López-Rubio, Features for stochastic approximation based foreground detection, Comput. Vis. Image Underst. 133 (2015) 30–50.
[18] P. Ochs, J. Malik, T. Brox, Segmentation of moving objects by long term video analysis, IEEE Trans. Pattern Anal. Mach. Intell. 36 (6) (2014) 1187–1200, doi:10.1109/TPAMI.2013.242.
[19] C. Papachristou, A. Delopoulos, A method for the evaluation of projective geometric consistency in weakly calibrated stereo with application to point matching, Comput. Vis. Image Underst. 119 (2014) 81–101.
[20] A. Papazoglou, V. Ferrari, Fast object segmentation in unconstrained video, in: Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV), 2013, pp. 1777–1784, doi:10.1109/ICCV.2013.223.
[21] Y. Sheikh, M. Shah, Bayesian modeling of dynamic scenes for object detection, IEEE Trans. Pattern Anal. Mach. Intell. 27 (11) (2005) 1778–1792.
[22] J. Shen, W. Yang, Z. Lu, Q. Liao, Information integration for accurate foreground segmentation in complex scenes, IET Image Process. 6 (5) (2012) 596–605.
[23] S.-W. Sun, Y.-C.F. Wang, F. Huang, H.-Y.M. Liao, Moving foreground object detection via robust SIFT trajectories, J. Vis. Commun. Image Represent. 24 (2013) 232–243.
[24] N. Werneck, A. Costa, Corisco: robust edgel-based orientation estimation for generic camera models, Image Vis. Comput. 31 (12) (2013) 969–981.
[25] K.M. Yi, K. Yun, S.W. Kim, H.J. Chang, H. Jeong, J.Y. Choi, Detection of moving objects with non-stationary cameras in 5.8ms: bringing motion detection to your mobile device, in: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2013, pp. 27–34.

# Bibliografía

[1] S.-F. Chang, "The holy grail of content-based media analysis," *MultiMedia, IEEE*, vol. 9, no. 2, pp. 6–10, Apr 2002.

[2] S. Mitra and T. Acharya, "Gesture recognition: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 3, pp. 311–324, May 2007.

[3] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 6, pp. 797–819, Nov 2011.

[4] B. Morris and M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 8, pp. 1114–1127, Aug 2008.

[5] S. Vishwakarma and A. Agrawal, "A survey on activity recognition and behavior understanding in video surveillance," *The Visual Computer*, vol. 29, pp. 983–1009, 2013.

[6] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance- a survey," in *Pervasive Computing (ICPC), 2015 International Conference on*, Jan 2015, pp. 1–6.

[7] K. N. Plataniotis and A. N. Venetsanopoulos, *Color image processing and applications*. Springer-Verlag New York, Inc., 2000.

[8] M. Tkalcic and J. Tasic, "Colour spaces: perceptual, historical and applicational background," in *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, 2003.

[9] T. Smith and J. Guild, "The c.i.e. colorimetric standards and their use," *Transactions of the Optical Society*, vol. 33, p. 73, 1931.

[10] D. L. MacAdam, *Color measurement*. Springer-Verlag Berlin Heidelberg, 1985.

[11] B. Hill, T. Roger, and F. W. Vorhagen, "Comparative analysis of the quantization of color spaces on the basis of the cielab color-difference formula," *ACM Trans. Graph.*, vol. 16, pp. 109–154, 1997.

[12] D. Pascale, "A review of rgb color spaces," The BabelColor Company, Tech. Rep., 2003.

[13] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes, "Proposal for a standard default color space for the internet-srgb," *Color and imaging conference*, vol. 1996, pp. 238–245, 1996.

[14] S. Süsstrunk, R. Buckley, and S. Swen, "Standard rgb color spaces," *Color and Imaging Conference*, vol. 1999, pp. 127–134, 1999.

[15] C. Poynton, *Digital video and HDTV algorithms and interfaces*. Morgan Kaufmann Publishers Inc., 2003.

[16] Y.-I. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," *Computer Graphics and Image Processing*, vol. 13, pp. 222 – 241, 1980.

[17] H. Palus, "Representations of colour images in different colour spaces," in *The Colour Image Processing Handbook*. Springer US, 1998.

[18] A. Ford and A. Roberts, "Colour space conversions," *Westminster University, London*, vol. 1998, pp. 1–31, 1998.

[19] M. Fairchild, *Color appearance models*. Wiley, 2013.

[20] C. Poynton, "Frequently asked questions about color," Tech. Rep., 1999.

[21] ——, "A guided tour of color space," in *New Foundations for Video Technology*, 1995.

[22] A. Pentland, "Smart rooms, smart clothes," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, 1998.

[23] F. El Baf, T. Bouwmans, and B. Vachon, "Comparison of background subtraction methods for a multimedia application," in *Systems, Signals and Image Processing, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. 14th International Workshop on*, 2007.

[24] S. Sarkar, P. Phillips, Z. Liu, I. Vega, P. Grother, and K. Bowyer, "The humanid gait challenge problem: data sets, performance, and analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 162–177, 2005.

[25] R. Colque and G. Camara-Chavez, "Progressive background image generation of surveillance traffic videos based on a temporal histogram ruled by a reward/penalty function," in *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, 2011.

[26] F. El Baf, T. Bouwmans, and B. Vachon, "Fuzzy integral for moving object detection," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on*, 2008.

[27] F. Porikli and O. Tuzel, "Human body tracking by adaptive background models and mean-shift analysis," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.

[28] Y. Kameda and M. Minoh, "A human motion estimation method using 3-successive video frames," *Proceedings of International Conference on Virtual Systems*, pp. 135–140, 1996.

[29] B. Lee and M. Hedley, "Background estimation for video surveillance," in *Image and Vision Computing New Zealand*, 2002.

[30] N. McFarlane and C. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications*, vol. 8, pp. 187–193, 1995.

[31] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, 2001.

[32] G. Cucchiara and P. Piccardi, "Detecting moving objects, ghosts and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1337–1342, 2003.

[33] S. Calderara, R. Melli, A. Prati, and R. Cucchiara, "Reliable background suppression for complex scenes," in *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, 2006.

[34] J. Zheng, Y. Wang, N. Nihan, and M. Hallenbeck, "Extracting roadway background image: a mode-based approach," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1944, pp. 82–88, 2006.

[35] D. Zhao, D. Liu, and Y. Yang, "An improved pic algorithm of background reconstruction for detecting moving object," in *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, 2008.

[36] L. Yu and Y. guang kong, "An algorithm of background reconstruction based on morphology and pic," in *International Workshop on Information Security and Application*, 2009.

[37] S. Fu, G. Jiang, and M. Yu, "An effective background subtraction method based on pixel change classification," in *Electrical and Control Engineering (ICECE), 2010 International Conference on*, 2010.

[38] A. Lipton, H. Fujiyoshi, and R. Patil, "Moving target classification and tracking from real-time video," in *Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on*, 1998.

[39] B. Bose and E. Grimson, "Improving object classification in far-field video," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2004.

[40] C. Stauffer, "Automatic hierarchical classification using time-based co-occurrences," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999.

[41] O. Javed and M. Shah, "Tracking and object classification for automated surveillance," in *Proceedings of the 7th European Conference on Computer Vision-Part IV*, 2002.

[42] L. M. Brown, "View independent vehicle/person classification," in *Proceedings of the ACM 2Nd International Workshop on Video Surveillance &Amp; Sensor Networks*, 2004.

[43] Y. Kuno, T. Watanabe, Y. Shimosakoda, and S. Nakagawa, "Automated detection of human for visual surveillance system," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, 1996.

[44] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A system for video surveillance and monitoring," Robotics Institute, Tech. Rep., 2000.

[45] X. Ma and W. Grimson, "Edge-based rich representation for vehicle classification," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005.

[46] L. Zhang, S. Li, X. Yuan, and S. Xiang, "Real-time object classification in video surveillance based on appearance learning," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[47] M. Tsuchiya and H. Fujiyoshi, "Evaluating feature importance for object classification in visual surveillance," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006.

[48] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, pp. 679–698, 1986.

[49] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, 1994.

[50] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, pp. 674–693, 1989.

[51] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[52] A. Yilmaz, X. Li, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1531–1536, 2004.

[53] D. H. Ballard and C. M. Brown, *Computer Vision*. Prentice-Hall, 1982.

[54] A. Ali and J. Aggarwal, "Segmentation and recognition of continuous human activity," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, 2001.

[55] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Comput. Surv.*, vol. 38, 2006.

[56] S. Zhu, T. Lee, and A. Yuille, "Region competition: unifying snakes, region growing, energy/bayes/mdl for multi-band image segmentation," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, 1995.

[57] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentl, "Pfinder: real-time tracking of the human body," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, 1997.

[58] S. J. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image and Vision Computing*, vol. 17, pp. 225 – 231, 1999.

[59] N. Paragios and R. Deriche, "Geodesic active regions and level set methods for supervised texture segmentation," *International Journal of Computer Vision*, vol. 46, pp. 223–247, 2002.

[60] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, pp. 1151–1163, 2002.

[61] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 564–577, 2003.

[62] Y. Zhong, A. Jain, and M.-P. Dubuisson-Jolly, "Object tracking using deformable templates," in *Computer Vision, 1998. Sixth International Conference on*, 1998.

[63] H. Nguyen and A. Smeulders, "Fast occluded object tracking by a robust appearance filter," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1099–1104, 2004.

[64] G. Edwards, C. Taylor, and T. Cootes, "Interpreting face images using active appearance models," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, 1998.

[65] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, pp. 696–710, 1997.

[66] M. Black and A. Jepson, "Eigentracking: robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, pp. 63–84, 1998.

[67] S. Avidan, "Support vector tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1064–1072, 2004.

[68] S. Park and J. Aggarwal, "A hierarchical bayesian network for event recognition of human actions and interactions," *Multimedia Systems*, vol. 10, pp. 164–179, 2004.

[69] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 51–65, 2005.

[70] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, pp. 58:1–58:48, 2013.

[71] C. Veenman, M. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 54–72, 2001.

[72] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina, Chapel Hill, NC, USA, Tech. Rep., 1995.

[73] G. Kitagawa, "Non-gaussian state-space modeling of nonstationary time series," *Journal of the American Statistical Association*, vol. 82, pp. 1032–1041, 1987.

[74] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, 1998.

[75] H. Schweitzer, J. Bell, and F. Wu, "Very fast template matching," in *Computer Vision - ECCV 2002*. Springer Berlin Heidelberg, 2002.

[76] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185 – 203, 1981.

[77] M. Bertalmio, G. Sapiro, and G. Randall, "Morphing active contours," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 733–737, 2000.

[78] M. Yokoyama and T. Poggio, "A contour-based moving object detection and tracking," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, 2005.

[79] A.-R. Mansouri, "Region tracking via level set pdes without motion computation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 947–961, 2002.

[80] I. Haritaoglu, D. Harwood, and L. S. David, "W4: real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 809–830, 2000.

[81] J. Kang, I. Cohen, and G. Medioni, "Object reacquisition using invariant appearance model," in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4 - Volume 04*, 2004.

[82] K. Sato and J. Aggarwal, "Temporal spatio-velocity transform and its application to tracking and interaction," *Computer Vision and Image Understanding*, vol. 96, pp. 100 – 128, 2004.

[83] R. Girisha and S. Murali, "Tracking humans using novel optical flow algorithm for surveillance videos," in *Proceedings of the Fourth Annual ACM Bangalore Conference*, 2011.

[84] J. Aggarwal and M. Ryoo, "Human activity analysis: a review," *ACM Comput. Surv.*, vol. 43, pp. 16:1–16:43, 2011.

[85] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, pp. 224 – 241, 2011.

[86] M. Rodriguez, J. Ahmed, and M. Shah, "Action mach a spatio-temporal maximum average correlation height filter for action recognition," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008.

[87] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004.

[88] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 257–267, 2001.

[89] Y. Sheikh, M. Sheikh, and M. Shah, "Exploring the space of a human action," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005.

[90] A. Yilmaz and M. Shah, "Recognizing human actions in videos acquired by uncalibrated moving cameras," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005.

[91] J. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," *International Journal of Computer Vision*, vol. 79, pp. 299–318, 2008.

[92] M. Ryoo and J. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009.

[93] T. Darrell and A. Pentland, "Space-time gestures," in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, 1993.

[94] D. M. Gavrila and L. S. Davis, "Towards 3-d model-based tracking and recognition of human movement: a multi-view approach," in *In International Workshop on Automatic Face- and Gesture-Recognition. IEEE Computer Society*, 1995.

[95] A. Veeraraghavan, R. Chellappa, and A. Roy-Chowdhury, "The function space of an activity," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006.

[96] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, 1992.

[97] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, "Modeling individual and group actions in meetings with layered hmms," *Multimedia, IEEE Transactions on*, vol. 8, pp. 509–520, 2006.

[98] P. Natarajan and R. Nevatia, "Coupled hidden semi markov models for activity recognition," in *Motion and Video Computing, 2007. WMVC '07. IEEE Workshop on*, 2007.

[99] N. Oliver, B. Rosario, and A. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[100] M. S. Ryoo and J. K. Aggarwal, "Semantic representation and recognition of continued and recursive human activities," *Int. J. Comput. Vision*, vol. 82, no. 1, pp. 1–24, 2009.

[101] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," in *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, 2002.

[102] E. Yu and J. K. Aggarwal, "Detection of fence climbing from monocular video," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006.

[103] P. Dai, H. Di, L. Dong, L. Tao, and G. Xu, "Group interaction analysis in dynamic context," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, pp. 34–42, 2009.

[104] D. Damen and D. Hogg, "Recognizing linked events: Searching the space of feasible explanations," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009.

[105] Y. Ivanov and A. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 852–872, 2000.

[106] D. Minnen, I. Essa, and T. Starner, "Expectation grammars: leveraging high-level expectations for activity recognition," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003.

[107] S.-W. Joo and R. Chellappa, "Attribute grammar-based event recognition and anomaly detection," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, 2006.

[108] V.-T. Vu, F. Bremond, and M. Thonnat, "Automatic video interpretation: a novel algorithm for temporal scenario recognition," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.

[109] S. Tran and L. Davis, "Event modeling and recognition using markov logic networks," in *Computer Vision - ECCV 2008*. Springer Berlin Heidelberg, 2008.

[110] A. Gupta, P. Srinivasan, J. Shi, and L. Davis, "Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009.

[111] T. Ko, "A survey on behavior analysis in video surveillance for homeland security applications," in *Applied Imagery Pattern Recognition Workshop, 2008. AIPR '08. 37th IEEE*, 2008.

[112] N. Cuntoor, B. Yegnanarayana, and R. Chellappa, "Activity modeling using event probability sequences," *Image Processing, IEEE Transactions on*, vol. 17, pp. 594–607, 2008.

[113] S. M. Khan and M. Shah, "Detecting group activities using rigidity of formation," in *Proceedings of the 13th Annual ACM International Conference on Multimedia*, 2005.

[114] T. Xiang and S. Gong, "Beyond tracking: Modelling activity and understanding behaviour," *International Journal of Computer Vision*, vol. 67, pp. 21–51, 2006.

[115] O. Boiman and M. Irani, "Detecting irregularities in images and in video," *International Journal of Computer Vision*, vol. 74, pp. 17–31, 2007.

[116] D. Moore, I. Essa, and I. Hayes, M.H., "Exploiting human actions and object context for recognition tasks," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999.

[117] A. Gupta and L. Davis, "Objects in action: An approach for combining action understanding and object perception," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[118] M. Ryoo and J. Aggarwal, "Hierarchical recognition of human activities interacting with objects," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[119] ——, "Stochastic representation and recognition of high-level group activities," *International Journal of Computer Vision*, vol. 93, pp. 183–200, 2011.

[120] N. Vaswani, A. Chowdhury, and R. Chellappa, "Activity recognition using the dynamics of the configuration of interacting objects," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003.

[121] L. Li, W. Huang, I.-H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *Image Processing, IEEE Transactions on*, vol. 13, pp. 1459–1472, 2004.

[122] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999.

[123] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure, "Robust foreground extraction technique using gaussian family model and multiple thresholds," in *Computer Vision - ACCV 2007*. Springer Berlin Heidelberg, 2007.

[124] N. Friedman and S. Russell, "Image segmentation in video sequences: a probabilistic approach," in *Proceedings of the thirteenth conference on uncertainty in artificial intelligence*, 1997.

[125] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, 1999.

[126] P. KaewTraKulPong and R. Rowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," *Proceedings of the Second European Workshop on Advanced Video Based Surveillance Systems*, pp. 149–158, 2001.

[127] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ser. ICPR '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 28–31.

[128] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[129] D. Mukherjee and Q. JonathanWu, "Real-time video segmentation using student's t mixture model," *Procedia Computer Science*, vol. 10, pp. 153 – 160, 2012.

[130] T. Haines and T. Xiang, "Background subtraction with dirichlet process mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, pp. 670–683, 2014.

[131] H.-H. Lin, T.-L. Liu, and J.-H. Chuang, "Learning a scene background model via classification," *Signal Processing, IEEE Transactions on*, vol. 57, pp. 1641–1654, 2009.

[132] J. Wang, G. Bebis, M. Nicolescu, M. Nicolescu, and R. Miller, "Improving target detection by coupling it with tracking," *Machine Vision and Applications*, vol. 20, pp. 205–223, 2009.

[133] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[134] J. Ma, J. Theiler, and S. Perkins, "Accurate on-line support vector regression," *Neural Computation*, vol. 15, pp. 2683–2703, 2003.

[135] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ser. ECCV '00. London, UK, UK: Springer-Verlag, 2000, pp. 751–767.

[136] R. Luque, D. López-Rodríguez, E. Dominguez, and E. Palomo, "A dipolar competitive neural network for video segmentation," in *Advances in Artificial Intelligence - IBERAMIA 2008*. Springer Berlin Heidelberg, 2008.

[137] R. Luque, D. Lopez-Rodriguez, E. Merida-Casermeiro, and E. Palomo, "Video object segmentation with multivalued neural networks," in *Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on*, 2008.

[138] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on Image Processing*, vol. 17, pp. 1168–1177, 2008.

[139] E. López-Rubio, R. M. Luque-Baena, and E. Domínguez, "Foreground detection in video sequences with probabilistic self-organizing maps," *International Journal of Neural Systems*, vol. 21, pp. 225–246, 2011.

[140] W. Kim and C. Kim, "Background subtraction for dynamic texture scenes using fuzzy color histograms," *Signal Processing Letters, IEEE*, vol. 19, pp. 127–130, 2012.

[141] F. El Baf, T. Bouwmans, and B. Vachon, "Type-2 fuzzy mixture of gaussians model: application to background modeling," in *Advances in Visual Computing*. Springer Berlin Heidelberg, 2008.

[142] ——, "A fuzzy approach for background subtraction," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, 2008.

[143] D. Manjula and M. Sivabalakrishnan, "Adaptive background subtraction in dynamic environments using fuzzy logic," *International Journal of Video & Image Processing and Network Security*, vol. 10, pp. 13–16, 2010.

[144] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 657–662, 2006.

[145] L. Maddalena and A. Petrosino, "A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection." *Neural Computing and Applications*, vol. 19, no. 2, pp. 179–186, 2010.

[146] D. Parks and S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *Advanced Video and Signal Based Surveillance, 2008. AVSS '08. IEEE Fifth International Conference on*, 2008.

[147] T. Bouwmans, "Recent advanced statistical background modeling for foreground detection - a systematic survey," *Recent Patents on Computer Science*, vol. 4, pp. 147–176, 2011.

[148] S. Brutzer, B. Hoferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011.

[149] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 2012.

[150] T. Bouwmans, F. El Baf, and B. Vachon, "Background modeling using mixture of gaussians for foreground detection - a survey," *Recent Patents on Computer Science*, vol. 1, pp. 219–237, 2008.

[151] M. Greiffenhagen, V. Ramesh, and H. Niemann, "The systematic design and analysis cycle of a vision system: a case study in video surveillance," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001.

[152] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, pp. 172–185, 2005.

[153] F. Campbell-West, P. Miller, and H. Wang, "Independent moving object detection using a colour background model," in *Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, 2006.

[154] X. Gao, T. Boult, F. Coetzee, and V. Ramesh, "Error analysis of background adaption," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000.

[155] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *19th International Conference on Pattern Recognition*, 2008.

[156] P. Chen, X. Chen, B. Jin, and X. Zhu, "Online EM algorithm for background subtraction," *Procedia Engineering*, vol. 29, pp. 164 – 169, 2012.

[157] J. Pilet, C. Strecha, and P. Fua, "Making background subtraction robust to sudden illumination changes," in *Proceedings of the European Conference on Computer Vision*, 2008.

[158] T. S. Haines and T. Xiang, "Background subtraction with dirichlet processes," in *Computer Vision ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 99–113.

[159] E. López-Rubio and R. M. Luque-Baena, "Stochastic approximation for background modelling," *Computer Vision and Image Understanding*, vol. 115, pp. 735–749, 2011.

[160] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[161] H. J. Kushner and G. G. Yin, *Stochastic approximation and Recursive Algorithms and Applications*. Springer-Verlag, 2003.

[162] T. Lai, "Stochastic approximation," *Annals of Statistics*, vol. 31, no. 2, pp. 391–406, 2003.

[163] S. Mukherjee and K. Das, "An adaptive GMM approach to background subtraction for application in real time surveillance," *International Journal of Research in Engineering and Technology*, vol. 2, pp. 25–29, 2013.

[164] Z. Zivkovic and F. Van der Heijden, "Recursive unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 651–656, 2004.

[165] A. Sobral, "BGSLibrary: An opencv c++ background subtraction library," in *IX Workshop de Visão Computacional (WVC'2013)*, 2013.

[166] A. Jensen, M. Loog, and A. Solberg, "Using multiscale spectra in regularizing covariance matrices for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, pp. 1851–1859, 2010.

[167] T. Schneider, "Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values," *Journal of Climate*, vol. 14, pp. 853–871, 2001.

[168] Y. Guo, T. Hastie, and R. Tibshirani, "Regularized linear discriminant analysis and its application in microarrays," *Biostatistics*, vol. 8, no. 1, pp. 86–100, 2007.

[169] B. Charles, "Image noise models," in *Handbook of Image and Video Processing (Second Edition)*. Academic Press, 2005.

[170] N. Alajlan, M. Kamel, and E. Jernigan, "Detail preserving impulsive noise removal," *Signal Processing: Image Communication*, vol. 19, pp. 993 – 1003, 2004.

[171] M. Figueiredo, J. Bioucas-Dias, and R. Nowak, "Majorization-minimization algorithms for wavelet-based image restoration," *Image Processing, IEEE Transactions on*, vol. 16, pp. 2980–2991, 2007.

[172] J.-S. Zhang, X.-F. Huang, and C.-H. Zhou, "An improved kernel regression method based on taylor expansion," *Applied Mathematics and Computation*, vol. 193, pp. 419 – 429, 2007.

[173] Z. Dengwen and C. Wengang, "Image denoising with an optimal threshold and neighbouring window," *Pattern Recognition Letters*, vol. 29, pp. 1694 – 1697, 2008.

[174] E. López-Rubio, "Restoration of images corrupted by gaussian and uniform impulsive noise," *Pattern Recognition*, vol. 43, pp. 1835 –1846, 2010.

[175] W. Kang, E. Lee, E. Chea, A. Katsaggelos, and J. Paik, "Compressive sensing-based image denoising using adaptive multiple sampling and optimal error tolerance," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[176] A. Srikrishna, B. Reddy, and M. Pompapathi, "Pixon based image denoising scheme by preserving exact edge locations," *Journal of The Institution of Engineers (India): Series B*, pp. 1–9, 2015.

[177] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 11-12, pp. 31 – 66, 2014.

[178] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4 – 21, 2014.

[179] L. Gammaitoni, P. Hänggi, P. Jung, and F. Marchesoni, "Stochastic resonance: A remarkable idea that changed our perception of noise," *The European Physical Journal B*, vol. 69, pp. 1–3, 2009.

[180] V. S. Rallabandi and P. K. Roy, "Magnetic resonance image enhancement using stochastic resonance in fourier domain," *Magnetic Resonance Imaging*, vol. 28, pp. 1361 – 1373, 2010.

[181] R. Chouhan and P. K. Biswas, "Image enhancement and dynamic range compression using novel intensity-specific stochastic resonance-based parametric image enhancement model," in *Image Processing (ICIP), 2014 IEEE International Conference on*, 2014.

[182] P. J. Bickel and E. Levina, "Regularized estimation of large covariance matrices," *Annals of Statistics*, vol. 36, pp. 199–227, 2008.

[183] J. Ning, Y. Yang, and F. Zhu, "Background modeling and fuzzy clustering for motion detection from video," *Journal of Multimedia*, vol. 8, pp. 626–631, 2013.

[184] M. Jlassi, A. Douik, and H. Messaoud, "Color images segmentation algorithms during a sports meeting: Application to soccer video images," *Journal of Circuits, Systems and Computers*, vol. 19, pp. 1307–1332, 2010.

[185] T. Gao, Z.-G. Liu, S.-H. Yue, J.-Q. Mei, and J. Zhang, "Traffic video-based moving vehicle detection and tracking in the complex environment," *Cybernetics and Systems*, vol. 40, no. 7, pp. 569–588, 2009.

[186] A. Doshi and M. Trivedi, "Satellite imagery based adaptive background models and shadow suppression," *Signal, Image and Video Processing*, vol. 1, no. 2, pp. 119–132, 2007.

[187] M. Balcilar, F. Karabiber, and A. Sonmez, "Performance analysis of lab2000hl color space for background subtraction," in *2013 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2013.

[188] C. M. Bishop and M. Svenson, "The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, pp. 215–234, 1998.

[189] M. M. Van Hulle, "Kernel-based topographic map formation by local density modeling," *Neural Computation*, vol. 14, no. 7, pp. 1561–1573, 2002.

[190] M. Van Hulle, "Maximum likelihood topographic map formation," *Neural Computation*, vol. 17, no. 3, pp. 503–513, 2005.

[191] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[192] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 11, pp. 1778–1792, 2005.

[193] M. Piccardi, "Background subtraction techniques: a review," in *IEEE Intl. Conf. on Systems, Man and Cybernetics*, 2004.

[194] J. Shen, W. Yang, Z. Lu, and Q. Liao, "Information integration for accurate foreground segmentation in complex scenes," *IET Image Processing*, vol. 6, no. 5, pp. 596–605, 2012.

[195] C. Benedek and T. Szirányi, "Study on color space selection for detecting cast shadows in video surveillance," *International Journal of Imaging Systems and Technology*, vol. 17, no. 3, pp. 190–201, 2007.

[196] J. A. Marchant and C. M. Onyango, "Shadow-invariant classification for scenes illuminated by daylight," *Journal of the Optical Society of America*, vol. 17, no. 11, pp. 1952–1961, 2000.

[197] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 1, Oct 2003, pp. 44–50.

[198] T. Parag, A. Elgammal, and A. Mittal, "A framework for feature selection for background subtraction," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1916–1923.

[199] B. Han and L. S. Davis, "Density-based multifeature background subtraction with support vector machine," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 1017–1023, 2012.

[200] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, pp. 511–518.

[201] X. Li and G. Du, "BSTBGA: A hybrid genetic algorithm for constrained multi-objective optimization problems," *Computers and Operations Research*, vol. 40, no. 1, pp. 282–302, 2013.

[202] C. Audet, J. Dennis Jr., and S. Le Digabel, "Trade-off studies in blackbox optimization," *Optimization Methods and Software*, vol. 27, no. 4-5, pp. 613–624, 2012.

[203] M. López-Ibáñez and T. Stützle, "An experimental analysis of design choices of multi-objective ant colony optimization algorithms," *Swarm Intelligence*, vol. 6, no. 3, pp. 207–232, 2012.

[204] O. Schütze, X. Esquivel, A. Lara, and C. Coello, "Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 504–522, 2012.

[205] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre, *Computer Vision - ACCV 2012 Workshops: ACCV 2012 International Workshops, Daejeon, Korea, November 5-6, 2012, Revised Selected Papers, Part I.* Springer Berlin Heidelberg, 2013, ch. A benchmark dataset for outdoor foreground/background extraction, pp. 291–300.

[206] Z. Chen and T. Ellis, "A self-adaptive gaussian mixture model," *Computer Vision and Image Understanding*, vol. 122, pp. 35 – 46, 2014.

[207] A.-L. Ellis and J. Ferryman, "Biologically-inspired robust motion segmentation using mutual information," *Computer Vision and Image Understanding*, vol. 122, pp. 47 – 64, 2014.

[208] L. Maddalena and A. Petrosino, "The 3dsobs+ algorithm for moving object detection," *Computer Vision and Image Understanding*, vol. 122, pp. 65 – 73, 2014.

[209] C. Spampinato, S. Palazzo, and I. Kavasidis, "A texton-based kernel density estimation approach for background modeling under extreme conditions," *Computer Vision and Image Understanding*, vol. 122, pp. 74 – 83, 2014.

[210] S. Yoshinaga, A. Shimada, H. Nagahara, and R. ichiro Taniguchi, "Object detection based on spatiotemporal background models," *Computer Vision and Image Understanding*, vol. 122, pp. 84 – 91, 2014.

[211] Y. Dong and G. N. DeSouza, "Adaptive learning of multi-subspace for foreground detection under illumination changes," *Computer Vision and Image Understanding*, vol. 115, pp. 31–49, January 2011.

[212] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 1999, pp. 1–19.

[213] J.-S. Hu and T.-M. Su, "Robust background subtraction with shadow and highlight removal for indoor surveillance," *EURASIP Journal on Applied Signal Processing*, vol. 2007, pp. 108–108, January 2007.

[214] A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 918–923, 2003.

[215] V. Reddy, C. Sanderson, A. Sanin, and B. Lovell, "Adaptive patch-based background modelling for improved foreground object segmentation and tracking," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, 2010, pp. 172–179.

[216] J. Li and Z. Miao, "Foreground segmentation for dynamic scenes with sudden illumination changes," *Image Processing, IET*, vol. 6, no. 5, pp. 606–615, 2012.

[217] D. Toth, T. Aach, V. Metzler, and B. E. X, "Illumination-invariant change detection," in *in IEEE Southwest Symposium on Image Analysis and Interpretation*, 2000, pp. 3–7.

[218] Y. Ivanov, A. Bobick, and J. Liu, "Fast lighting independent background subtraction," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 199–207, 2000.

[219] B. Xie, V. Ramesh, and T. Boult, "Sudden illumination change detection using order consistency," *Image and Vision Computing*, vol. 22, no. 2, pp. 117 – 125, 2004.

[220] A. Agrawal, R. Raskar, and R. Chellappa, "Edge suppression by gradient field transformation using cross-projection tensors," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 2301–2308.

[221] V. Reddy, C. Sanderson, and B. Lovell, "Improved foreground detection via block-based classifier cascade with probabilistic decision integration," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 1, pp. 83–93, Jan 2013.

[222] M. Bales, D. Forsthoefel, D. Wills, and L. Wills, "Illumination change compensation techniques to improve kinematic tracking," in *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, Jan 2011, pp. 434–439.

[223] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *AI 2006: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2006.

[224] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Jounal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.

[225] D. Farcas, C. Marghes, and T. Bouwmans, "Background subtraction via incremental maximum margin criterion: A discriminative subspace approach," *Machine Vision and Applications*, vol. 23, no. 6, pp. 1083–1101, 2012.

[226] I. Huerta, A. Amato, X. Roca, and J. González, "Exploiting multiple cues in motion segmentation based on background subtraction," *Neurocomputing*, vol. 100, pp. 183–196, 2013.

[227] Y. Xiangdong, Y. Jie, and W. Na, "Removal of disturbance of sudden illumination change based on color gradient fusion Gaussian model," *International Journal of Advancements in Computing Technology*, vol. 5, no. 2, pp. 86–92, 2013.

[228] J. Van Es, T. Vladusich, and F. Cornelissen, "Local and relational judgements of surface colour: Constancy indices and discrimination performance," *Spatial Vision*, vol. 20, no. 1-2, pp. 139–154, 2007.

[229] X.-Q. Yu, X.-N. Chen, and M.-Y. Jiang, "Detection of moving object in moving background based on feature vector field fuzzy segmentation and OTSU method," *Opto-Electronic Engineering*, vol. 39, pp. 94–102, 2012.

[230] M. Sivabalakrishnan and D. Manjula, "Performance analysis of fuzzy logic-based background subtraction in dynamic environments," *Imaging Science Journal*, vol. 60, pp. 39–46, 2012.

[231] Y. Ding, W.-H. Li, J.-T. Fan, and H.-M. Yang, "Robust moving object detection under complex background," *Computer Science and Information Systems*, vol. 7, pp. 201–210, 2010.

[232] A. Shimada and R.-I. Taniguchi, "Hybrid background modeling for long-term and short-term illumination changes," *IEEJ Transactions on Electronics, Information and Systems*, vol. 130, pp. 1524–1529+4, 2010.

[233] X. Lu, T. Izumi, L. Teng, T. Horie, and L. Wang, "A novel background subtraction method for moving vehicle detection," *IEEJ Transactions on Fundamentals and Materials*, vol. 132, pp. 857–863, 2012.

[234] L. Li, I.-H. Gu, M. Leung, and Q. Tian, "Adaptive background subtraction based on feedback from fuzzy classification," *Optical Engineering*, vol. 43, pp. 2381–2394, 2004.

[235] Y.-F. Mao and P.-F. Shi, "Gaussian kernel density estimation-based background modeling with noise and shadow suppression," *Xitong Fangzhen Xuebao / Journal of System Simulation*, vol. 17, pp. 1182–1184, 2005.

[236] L. Liu, N. Sang, and R. Huang, "Background subtraction using shape and colour information," *Electronics Letters*, vol. 46, pp. 41–43, 2010.

[237] K. Anderson and P. McOwan, "Robust real-time face tracker for cluttered environments," *Computer Vision and Image Understanding*, vol. 95, pp. 184–200, 2004.

[238] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *IEEE 12th International Conference on Computer Vision*, 2009.

[239] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *Computer Vision ECCV 2012*. Springer Berlin Heidelberg, 2012.

[240] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *Computer Vision ECCV 2010*. Springer Berlin Heidelberg, 2010.

[241] K. Patwardhan, G. Sapiro, and V. Morellas, "Robust foreground detection in video using pixel layers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 746–751, 2008.

[242] Y. Zhang, S. Kiselewich, W. Bauson, and R. Hammoud, "Robust moving object detection at distance in the visible spectrum and beyond using a moving camera," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, 2006.

[243] D. Chetverikov, S. Fazekas, and M. Haindl, "Dynamic texture as foreground and background," *Machine Vision and Applications*, vol. 22, pp. 741–750, 2011.

[244] T. Lim, B. Han, and J. H. Han, "Modeling and segmentation of floating foreground and background in videos," *Pattern Recognition*, vol. 45, no. 4, pp. 1696 – 1706, 2012.

[245] S. Kwak, T. Lim, W. Nam, B. Han, and J. H. Han, "Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[246] V. Mahadevan and N. Vasconcelos, "Spatiotemporal saliency in dynamic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 171–177, 2010.

[247] X. Cui, J. Huang, S. Zhang, and D. Metaxas, "Background subtraction using low rank and group sparsity constraints," in *Computer Vision ECCV 2012*. Springer Berlin Heidelberg, 2012.

[248] M. Narayana, A. Hanson, and E. Learned-Miller, "Coherent motion segmentation in moving camera videos using optical flow orientations," in *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[249] S. Kang, J.-K. Paik, A. Koschan, B. R. Abidi, and M. A. Abidi, "Real-time video tracking using ptz cameras," in *Sixth International Conference on Quality Control by Artificial Vision*, 2003.

[250] R. Cucchiara, A. Prati, and R. Vezzani, "Advanced video surveillance with pan tilt zoom cameras," in *Proceedings of the Workshop on Visual Surveillance (VS) at ECCV 2006*, 2006.

[251] L. Robinault, S. Bres, and S. Miguet, "Real time foreground object detection using ptz camera," in *International Conference on Computer Vision, Theory and Applications (VISAPP'09)*, 2009.

[252] P. Azzari and A. Bevilacqua, "Joint spatial and tonal mosaic alignment for motion detection with ptz camera," in *Image Analysis and Recognition*. Springer Berlin Heidelberg, 2006.

[253] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[254] S. Amri, W. Barhoumi, and E. Zagrouba, "A robust framework for joint background/foreground segmentation of complex video scenes filmed with freely moving camera," *Multimedia Tools and Applications*, vol. 46, no. 2-3, pp. 175–205, 2010.

[255] S. W. Kim, K. Yun, K. M. Yi, S. J. Kim, and J. Y. Choi, "Detection of moving objects with a moving camera using non-panoramic background model," *Machine Vision and Applications*, vol. 24, no. 3, pp. 1015–1028, 2013.

[256] S.-W. Sun, Y.-C. F. Wang, F. Huang, and H.-Y. M. Liao, "Moving foreground object detection via robust SIFT trajectories," *Journal of Visual Communication and Image Representation*, vol. 24, pp. 232–243, 2013.

[257] A. Alahi, P. Vandergheynst, M. Bierlaire, and M. Kunt, "Cascade of descriptors to detect and track objects across any network of cameras," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 624–640, 2010.

[258] Y.-S. Huang, Z.-H. Ou, H.-H. Yu, and H.-W. Hsieh, "Non-uniform SURF feature point detection and matching," *Applied Mechanics and Materials*, vol. 284-287, pp. 3184–3188, 2013.

[259] Y.-T. Wang, C.-T. Chi, and Y.-C. Feng, "Robot simultaneous localization and mapping using speeded-up robust features," *Applied Mechanics and Materials*, vol. 284-287, pp. 2142–2146, 2013.

[260] N. Werneck and A. Costa, "Corisco: Robust edgel-based orientation estimation for generic camera models," *Image and Vision Computing*, vol. 31, no. 12, pp. 969–981, 2013.

[261] C. Papachristou and A. Delopoulos, "A method for the evaluation of projective geometric consistency in weakly calibrated stereo with application to point matching," *Computer Vision and Image Understanding*, vol. 119, pp. 81–101, 2014.

[262] E. López-Rubio and J. M. Ortiz-de Lazcano-Lobato, "Soft clustering for nonparametric probability density function estimation," *Pattern Recognition Letters*, vol. 29, pp. 2085–2091, 2008.

[263] E. López-Rubio, "Robust location and spread measures for nonparametric probability density function estimation," *International Journal of Neural Systems*, vol. 19, no. 5, pp. 345–357, 2009.

[264] S. Bano and A. Cavallaro, "Discovery and organization of multi-camera user-generated videos of the same event," *Information Sciences*, vol. 302, pp. 108 – 121, 2015.

[265] T. Rapakoulia, K. Theofilatos, D. Kleftogiannis, S. Likothanasis, A. Tsakalidis, and S. Mavroudi, "EnsembleGASVR: A novel ensemble method for classifying missense single nucleotide polymorphisms," *Bioinformatics*, vol. 30, no. 16, pp. 2324–2333, 2014.

[266] O. Barnich and M. Van Droogenbroeck, "Vibe: A powerful random technique to estimate the background in video sequences," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 2009.

[267] A. Papazoglou and V. Ferrari, "Fast object segmentation in unconstrained video," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013.

[268] P. Ochs, J. Malik, and T. Brox, "Segmentation of moving objects by long term video analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 6, pp. 1187–1200, 2014.

[269] C. Coffin, J. Ventura, and T. Höllerer, "A repository for the evaluation of image-based orientation tracking solutions," in *Proc. 2nd Intl. Workshop on AR/MR Registration, Tracking and Benchmarking (TrakMark 2011), in conjunction with ISMAR*, 2011.

[270] D. A. Klein and A. B. Cremers, "Boosting scalable gradient features for adaptive real-time tracking," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[271] G. M. García, D. A. Klein, J. Stückler, S. Frintrop, and A. B. Cremers, "Adaptive multi-cue 3D tracking of arbitrary objects," in *Pattern Recognition*. Springer Berlin Heidelberg, 2012.

[272] P. Ochs and T. Brox, "Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions," in *IEEE International Conference on Computer Vision (ICCV)*, Nov 2011, pp. 1583–1590.

[273] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[274] K. M. Yi, K. Yun, S. W. Kim, H. J. Chang, H. Jeong, and J. Y. Choi, "Detection of moving objects with non-stationary cameras in 5.8ms: bringing motion detection to your mobile device," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013.

[275] F. J. López-Rubio, E. Domínguez, E. J. Palomo, E. López-Rubio, and R. M. Luque-Baena, "Selecting the color space for self-organizing map based foreground detection in video," *Neural Processing Letters*, vol. 43, no. 2, pp. 345–361, 2015.

[276] F. J. López-Rubio and E. López-Rubio, "Features for stochastic approximation based foreground detection," *Computer Vision and Image Understanding*, vol. 133, pp. 30–50, 2015.

[277] ——, "Local color transformation analysis for sudden illumination change detection," *Image and Vision Computing*, vol. 37, pp. 31 – 47, 2015.

[278] ——, "Foreground detection for moving cameras with stochastic approximation," *Pattern Recognition Letters*, vol. 68, Part 1, pp. 161 – 168, 2015.

[279] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," *IEEE Transactions on Image Processing*, vol. 17, pp. 1737–1754, 2008.

[280] J. Suo, Y. Deng, L. Bian, and Q. Dai, "Joint non-gaussian denoising and superresolving of raw high frame rate videos," *IEEE Transactions on Image Processing*, vol. 23, pp. 1154–1168, 2014.

# Nomenclatura

| | |
|---|---|
| AC | exactitud |
| $\boldsymbol{B}_t\,(p)$ | modelo de fondo del píxel cuya posición es $p$ en el instante $t$ |
| $\boldsymbol{C}$ | matriz de covarianzas asociada a un píxel |
| $card$ | cardinalidad o número de elementos de un conjunto |
| CMY | espacio de color utilizado para la impresión |
| CMYK | espacio de color similar al CMY, pero con una cuarta tinta de color negro |
| $col$ | número de columnas de un fotograma o secuencia |
| $D$ | número de canales o componentes de un fotograma o secuencia |
| $\delta^m$ | distancia de Mahalanobis |
| $\delta^e$ | distancia euclídea |
| det | determinante |
| $\boldsymbol{\Sigma}_{i,t}(p)$ | matriz de covarianzas de la distribución de probabilidad asociada a la i-ésima componente de una distribución de mixtura en el instante $t$ del píxel cuya posición es $p$ |
| $E\,[X]$ | esperanza de la variable aleatoria $X$ |
| $fil$ | número de filas de un fotograma o secuencia |
| FM | F-medida |
| FNR | tasa de falsos negativos |
| FPR | tasa de falsos positivos |
| fps | fotogramas por segundo |
| GT | *ground truth* |
| $G\,(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ | función de densidad de probabilidad gaussiana de media $\boldsymbol{\mu}$ y covarianza $\boldsymbol{\Sigma}$ |

| | |
|---|---|
| HSI | espacio de color fenoménico con una componente lumínica y dos cromáticas |
| HSL | denominación genérica de los espacios de color fenoménicos |
| HSV | espacio de color fenoménico con una componente lumínica y dos cromáticas |
| $\mathbf{I}$ | matriz identidad |
| $\boldsymbol{I}_t$ | imágen multicanal del fotograma de entrada en el instante $t$ |
| Lab | espacio de color uniforme propuesto por la CIE que consta de una componente lumínica y dos cromáticas |
| Luv | espacio de color uniforme propuesto por la CIE que consta de una componente lumínica y dos cromáticas |
| $\mathcal{M}$ | función de pertenencia al primer plano del método de detección de movimiento |
| $\widehat{\mathcal{M}}$ | función de pertenencia al primer plano del *ground truth* |
| $\boldsymbol{\mu}_{i,t}(p)$ | vector de medias de la distribución de probabilidad asociada a la i-ésima componente de una distribución de mixtura en el instante $t$ del píxel cuya posición es $p$ |
| N/A | no aplica |
| Ohta | espacio de color basado en la teoría de los colores opuestos |
| $p$ | posición de un píxel |
| $P_B(\boldsymbol{p}_t)$ | probabilidad a posteriori de que el vector de rasgos $\boldsymbol{p}_t$ pertenezca al fondo |
| $P_F(\boldsymbol{p}_t)$ | probabilidad a posteriori de que el vector de rasgos $\boldsymbol{p}_t$ pertenezca al primer plano |
| $\pi_{i,t}$ | probabilidad a priori de la i-ésima componente de una distribución de mixtura en el instante $t$ |
| $\mathrm{p}\left(\boldsymbol{p}_t\right)$ | densidad de probabilidad en el instante $t$ y en la posición $p$ |
| $\mathrm{p}\left(\boldsymbol{p}_t \mid B, i\right)$ | densidad de probabilidad de la i-ésima componente de la mixtura de distribución del fondo |
| PR | precisión |
| $\boldsymbol{\Psi}$ | matriz diagonal que modela los efectos del ruido de cuantización y compresión en los píxeles de entrada |

| | |
|---|---|
| $\boldsymbol{p}_t$ | vector de rasgos en el instante $t$ del píxel cuya posición es $p$ |
| $\boldsymbol{R}$ | matriz de correlación o matriz de covarianza no centrada asociada a un píxel |
| RC | recall |
| RGB | espacio de color con tres colores primarios: rojo, vede y azul |
| $R_{i,t}$ | responsabilidad o probabilidad a posteriori en el instante $t$ asociada a la componente i-ésima de una distribución de mixtura |
| $\sigma$ | desviación típica |
| $T$ | umbral |
| $t$ | instante de una secuencia |
| TPR | tasa de verdaderos positivos |
| $U(\boldsymbol{x})$ | función de densidad de probabilidad uniforme |
| $Vol$ | volumen del soporte de una función de densidad de probabilidad |
| YCbCr | espacio de color para la televisión con una componente lumínica y dos cromáticas relacionado con los interfaces digitales |
| $Y'IQ$ | espacio de color para la televisión con una componente lumínica y dos cromáticas asociado al sistema de codificación de vídeo NTSC |
| YPbPr | espacio de color para la televisión con una componente lumínica y dos cromáticas relacionado con los interfaces analógicos |
| $Y'UV$ | espacio de color para la televisión con una componente lumínica y dos cromáticas asociado tanto al sistema de codificación NTSC como al PAL |