Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Detección y caracterización de personas en espacios inteligentes con cámaras de color

Autor: Marcos Baptista Ríos

Tutores: Marta Marrón Romera y Cristina Losada Gutierrez

2015

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Detección y caracterización de personas en espacios inteligentes con cámaras de color

Autor: Marcos Baptista Ríos

Directores: Marta Marrón Romera y Cristina Losada Gutierrez

Tribunal:

Presidente: Javier Macías Guarasa

Vocal 1º: Luis Miguel Bergasa Pascual

Vocal 2º: Cristina Losada Gutiérrez

Calificación: .....

Fecha: .....

A mi madre y a mi padre.

Resumen

El objetivo de este proyecto es el estudio y análisis de un sistema de detección y seguimiento de personas en espacios inteligentes con cámaras de color. Para ello, se ha diseñado un sistema basado en dos etapas: detector de personas y seguimiento. El detector de personas se ha diseñado utilizando la tecnología de ventana deslizante con descriptores HOG y un clasificador SVM lineal. El sistema de seguimiento se ha elaborado con un banco de filtros de Kalman con modelo de velocidad constante.

Se ha demostrado que el detector de personas localiza y determina el tamaño de lo que considera persona. Esta información es entregada al sistema seguidor, el cual, descarta las detecciones que, por error del detector, no corresponden con persona y pone en proceso de seguimiento las que sí lo son. Además, el algoritmo que realiza el seguimiento hace robusto el sistema completo porque filtra los errores de la etapa de detección y resuelve otros como cruce de individuos u oclusión.

Palabras clave: Detección y seguimiento, ventana deslizanze, descriptores HOG, clasificador SVM lineal, Filtro de Kalman.

Abstract

The aim of the project is to study, develop and analyse a human detection and tracking system in intelligent spaces with colour cameras. To this purpose, a two stage system has been designed: human detection and tracking. The human detection stage is done by using the sliding window technique combined with the HOG feature descriptor and a linear SVM based classifier. The tracking system consists of a Kalman filter bank in which each filter implements the constant velocity model.

It has been shown that the human detector finds a person and determines its dimensions. This information is delivered to the tracker, which excludes those detections that, by mistake, do not correspond to a person and initiates the tracking process for those that do. Furthermore, the tracking algorithm makes the whole system robust because it filters the errors of the human detector and solves others such as crossing and occlusion.

Keywords: Detection and tracking, sliding window, HOG features descriptors, linear SVM classifier, Kalman filter.

Resumen extendido

Una de las apuestas de la investigación actual es la de crear sistemas que sean capaces de analizar la interacción entre las personas y su entorno natural (interior o exterior, dotado de inteligencia o no) y extraer información de alto nivel acerca de los comportamientos observados. Para conseguir esa información de alto nivel es necesario cumplir con la tarea básica de detectar la posición de la persona o personas y saber la trayectoria que siguen sobre la escena. Para ello, se elaboran, en el bajo nivel, los sistemas de detección y seguimiento.

Este proyecto en concreto, toma como punto de partida el trabajo presentado en [1] y su objetivo es el estudio, desarrollo y análisis de un **sistema de detección y seguimiento de personas** en espacios inteligentes con cámaras de color. Con este fin, se ha diseñado un sistema basado en dos etapas: **etapa de detección** de personas y **etapa de seguimiento**.

En primer lugar, se ha realizado un estudio con el fin de entender qué significa *espacio inteligente* y analizar las diferentes alternativas existentes en la literatura para implementar las dos etapas mencionadas.

En relación a la etapa de detección de personas, la investigación se ha centrado en las herramientas existentes para extraer características de una imagen, con especial detenimiento en el conjunto formado por **ventana deslizante** y **descriptores HOG**. También se ha realizado un amplio estudio teórico sobre los clasificadores basados en **máquinas de soporte vectorial**.

Con respecto a la etapa de seguimiento, se ha recopilado información sobre los distintos objetivos de utilizar un sistema de seguimiento en estas aplicaciones, así como las diferentes alternativas existentes. Se hace especial énfasis en entender lo que es el proceso de asociación y en comprender las fases del **filtro de Kalman**.

Apoyándose en la teoría, se ha diseñado un sistema cuya etapa de detección implementa el conjunto **ventana deslizante** y **descriptor HOG**, además de recurrir a una **SVM lineal** para realizar la clasificación. Es importante destacar que la imagen real que llega a la etapa de detección será escalada con el fin de aumentar las posibilidades de encontrar personas en campos de visión más lejanos. En la figura 2 se presenta un diagrama que resume el proceso que sigue la etapa de detección de personas.

Para el sistema seguidor, se ha decidido implementar un **banco de filtros de Kalman**. Cada persona que esté siendo seguida tendrá asociado un filtro del banco. El tipo de filtro utilizado trabaja según se presenta en la figura 3 pero aplicando el modelo de velocidad constante. Para asociar las medidas recibidas de detección con su respectivo filtro de Kalman se utiliza el algoritmo de asociación del vecino más próximo (NN: Nearest Neighbour).

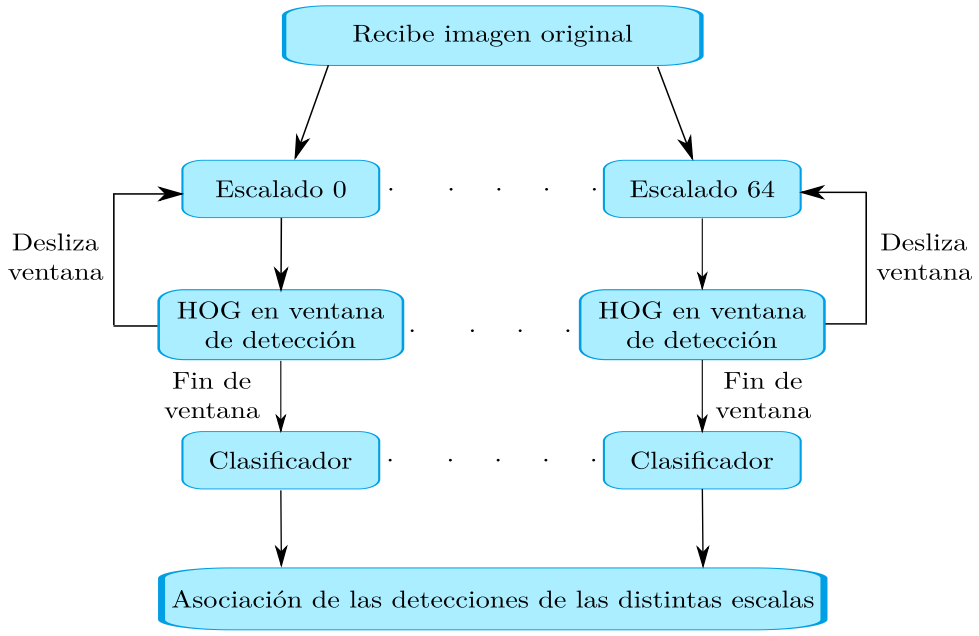


Figura 2: Funcionamiento general del detector de personas.

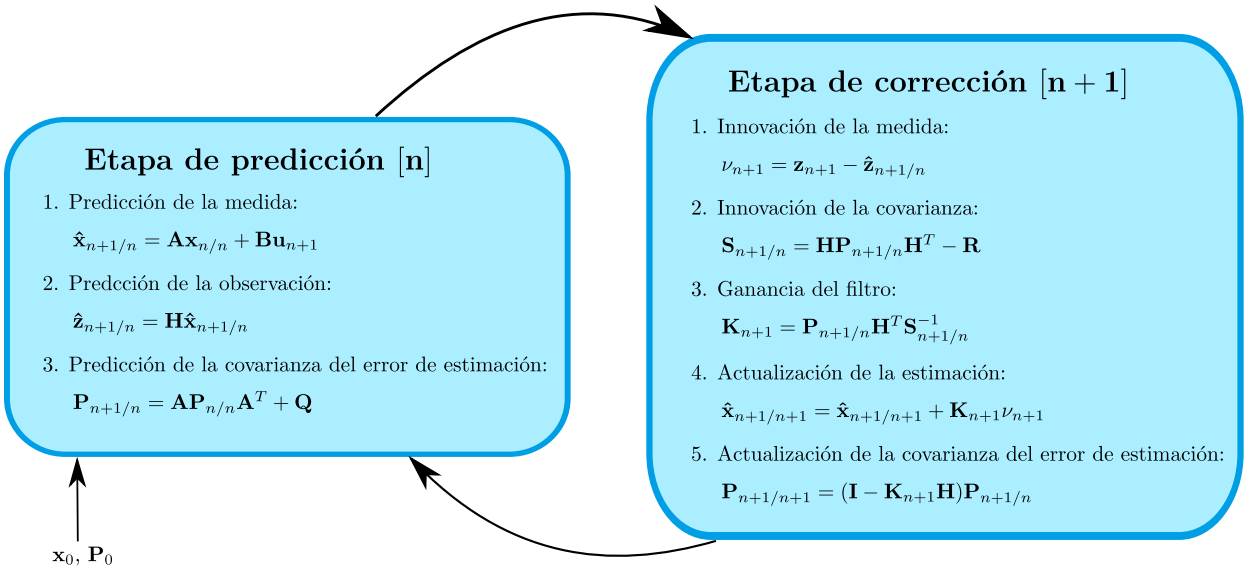


Figura 3: Ecuaciones de las etapas del filtro de Kalman.

Con todo implementado como se ha descrito, se obtiene un sistema completo de detección y seguimiento capaz de:

- Encontrar el lugar de la imagen donde se encuentra o encuentran las personas.
- Deducir el tamaño de la persona.
- Realizar un seguimiento de las personas que haya en la escena.
- Solventar con el banco de filtros los errores momentáneos que existan en la etapa de detección (continuación de seguimientos, oclusiones, cruce de personas,etc).

Para evaluar el algoritmo implementado se han realizado diversas pruebas experimentales utilizando tanto imágenes de una base de datos ampliamente utilizada en la literatura (CAVIAR Project [2]),

como otras grabadas para la aplicación desarrollada. En todos los casos, se ha comprobado el correcto funcionamiento del sistema.

En cuanto al tiempo de cómputo del sistema de detección y seguimiento completo, la ejecución de vídeos de la base de datos de referencia (CAVIAR Project [2]) la primera etapa tiene un tiempo medio de $t_{m_{detecCAVIAR}} = 187{,}63ms$ y el coste computacional del banco de filtros es $t_{m_{seg}} = 0{,}723ms$. Con estos datos, se puede ver que el sistema diseñado e implementado tiene un tiempo de cómputo menor (entre 15 y 20 veces menor) que el trabajo de referencia[1].

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Introducción al Trabajo Fin de Grado

Eres más valiente de lo que crees, más fuerte de lo que te ves y más inteligente de lo que piensas.

Winnie the Pooh

1.1 Introducción

El reconocimiento de actividad de personas es actualmente un objetivo fundamental en visión artificial, motivado por el potencial de sus múltiples aplicaciones: identificación de personas, control de aforos, detección de eventos anómalos o videovigilancia [6], [7]. Los sensores típicamente utilizados son cámaras de color y de profundidad (múltiples en algunos casos), y las características extraídas para alcanzar el reconocimiento abarcan tanto el bajo, como el medio y alto nivel. Normalmente éstas son la entrada a un clasificador, cuyo rendimiento es evaluado en un amplio rango de bases de datos disponibles al efecto [8]. Trabajos más recientes también abordan el reconocimiento de actividad en entornos reales [9], siendo este, todavía un tema sin resolver.

En este trabajo fin de grado se ha puesto en marcha un sistema de extracción de características que ayuden a realizar el reconocimiento de la actividad de las personas. En resumen, se plantea implementar un algoritmo de detección de personas en imágenes en color en tiempo real, y extracción de algunas de sus características físicas (posición, orientación, número de personas, velocidad, tamaño, etc.).

En esta línea, el trabajo aquí propuesto continua uno previo realizado dentro del Grupo de Ingeniería Electrónica Aplicada a Espacios Inteligentes y Transporte (GEINTRA) por José Ángel Cruz Lozano como trabajo fin de máster [1]. La mejora incluida en el trabajo fin de grado aquí expuesto, respecto al trabajo previo es doble: evaluar el sistema desarrollado, solucionar los errores que se encuentren y adaptarlo para su ejecución en tiempo real.

1.2 Objetivos

El objetivo global del proyecto es la extracción de características de secuencias de imágenes capturadas en tiempo real con una cámara de color de un espacio interior que permitan concluir la presencia de personas, el número de ellas, su pose, y otras características físicas.

Para alcanzar este objetivo general será necesario, por tanto, cubrir varios objetivos específicos que se detallan a continuación:

1. Detectar si aparecen personas en la imagen y cuántas.
2. Obtener su pose (posición y orientación) en la imagen.
3. Seguir la posición de las personas en distintas imágenes secuenciales, extrayendo así información acerca de su movimiento.
4. Estimar el tamaño de la persona.

Como ya se ha comentado, se propone alcanzar estos objetivos mediante la modificación del trabajo desarrollado en [1]. En la figura 1.1 se muestra un diagrama de bloques de las partes fundamentales del trabajo referido.

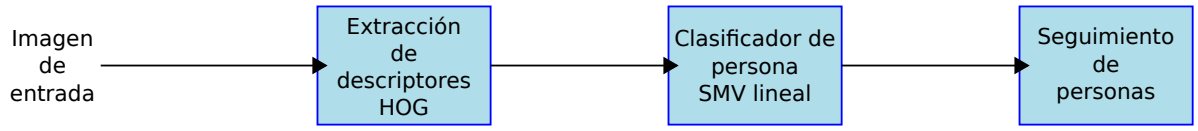


Figura 1.1: Esquema simplificado del trabajo desarrollado en [1].

1.3 Solución propuesta

La solución que se propone en este trabajo para conseguir los objetivos descritos consiste en desarrollar un sistema con las mismas etapas que [1] (ver figura 1.1), pero incluyendo modificaciones en cada una de ellas.

Para la detección de personas se ha decido implementar un sistema de extracción de características basado en ventana deslizante y descriptores HOG (histograma de gradientes orientados) que además, realiza un análisis en distintas escalas de la imagen de entrada al sistema. En concreto, el detector trabaja con 65 escalas, a diferencia del propuesto en [1] que lo hacía con 28.

Como método para decidir si los descriptores extraídos definen a una persona o no, se hace uso de un clasificador SVM lineal. Dicho clasificador, además de indicar si existe persona, entrega a su salida el lugar en la imagen donde se encuentra la persona.

Como sistema de seguimiento, se emplea un banco de seguidores basados en el filtro de Kalman, creando en concreto, un filtro por cada detección. Con este sistema se pretende resolver problemas de pérdidas puntuales de la detección de una persona bien por fallos en el detector o por oclusión de la persona con elementos de la escena, de modo que se consiga incrementar la robustez del conjunto.

Además, el nuevo sistema permite obtener una información más completa de las personas, ya que además de realizar la detección, se extraen características de bajo nivel (dirección de movimiento, velocidad, etc.)

Desarrollando esta estructura con las características comentadas se consigue, además, obtener unos tiempos de ejecución más bajos que los que se conseguían con la solución propuesta por [1].

1.4 Organización de la memoria

La memoria del presente Trabajo Fin de Grado, está estructurada en 5 secciones. Cada una de ellas se indica y se describe brevemente a continuación.

En el capítulo 1 se introduce y contextualiza el trabajo. Se presentan los objetivos a cumplir y la solución propuesta para la consecución de los mismos.

En el capítulo 2 se describe la teoría sobre la que se apoya el trabajo desarrollado. Se define brevemente el concepto de espacio inteligente (sección 2.1). También se explica el proceso de detección de personas en imágenes (sección 2.2) así como la teoría del filtro de Kalman para su uso en tareas de seguimiento (sección 2.3).

La forma en la que se ha implementado cada parte del sistema se explica en el capítulo 3. Se explican las soluciones adoptadas tanto para el diseño del detector (sección 3.1) de personas como para el del sistema seguidor (sección 3.2).

Los resultados de analizar los distintos bloques que forman el conjunto se disponen en el capítulo 4, tanto para la etapa de detección (sección 4.1) como para la de seguimiento (sección 4.2).

En el capítulo 5 se presentan las conclusiones finales obtenidas del desarrollo del trabajo (sección 5.1). Así mismo se proponen distintas líneas futuras del mismo (sección 5.2).

Capítulo 2

Estudio teórico

2.1 Espacio inteligente

Una parte amplia de la producción científica contemporánea, como es la de la robótica, los interfaces hombre-máquina o las redes de sensores, desarrolla en la actualidad con gran velocidad el objetivo de dotar de inteligencia y poder de decisión a los sistemas que ahora sustentan las redes de información. En este marco visionario se desarrollan los llamados *espacios inteligentes*, en los cuales la tecnología se encuentra inmersa en los espacios ocupados por el usuario, poniendo al servicio del mismo su interacción con el entorno. El *espacio inteligente* tiene a su disposición una red de sensores que le permiten obtener información del mundo que observa, y una red de actuadores (robots, pantallas de información, etc.) que habilitan su interacción con el usuario. La red de sensores y la inteligencia que los gobierna es por tanto imperceptible para el usuario, tomando decisiones y cooperando para la consecución de un servicio o tarea. Las aplicaciones de los *espacios inteligentes* son numerosas y de emergente aplicabilidad e implantación práctica. Abarcan todos los ámbitos de la robótica de servicios, tareas de seguridad y vigilancia automática, ayudas a personas con discapacidad y por supuesto los servicios domóticos orientados al gran público. La figura 2.1 muestra la estructura general de un *espacio inteligente*.

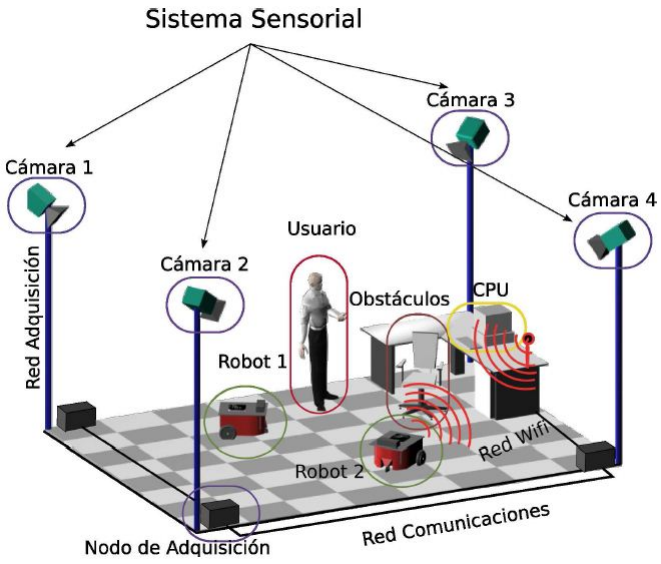


Figura 2.1: Estructura del espacio inteligente.

2.2 Detección de personas

El **detector de personas** visual, es el sistema que se encarga de determinar si en una imagen existen personas. En caso afirmativo, este deberá proporcionar tanto la posición como el tamaño en la imagen del individuo localizado. La estructura que tiene este bloque se muestra en la figura 2.2 y se compone de dos módulos: el sistema de extracción de información de la imagen y el clasificador.

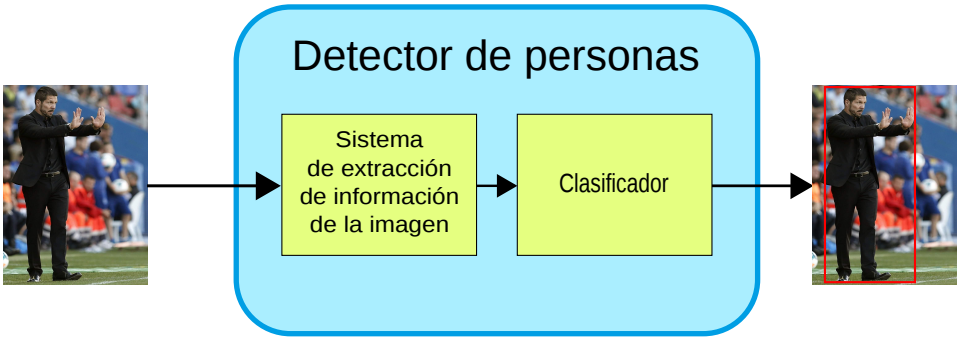


Figura 2.2: Estructura del detector de personas.

2.2.1 Extracción de información de la imagen

Según los distintos tipos de técnicas utilizadas para extraer información y características de imágenes, los detectores pueden ser: **basados en segmentación**, **basados en características de forma** y **basados en ventana deslizante**.

- Los **detectores basados en segmentación** [10] hacen uso de un conocimiento a priori del fondo para separarlo del primer plano que correspondería a las personas a detectar. Este método presenta múltiples inconvenientes y es muy poco robusto a cambios de iluminación, fondos dinámicos y movimientos de la cámara.

- Los **detectores basados en características de forma** [11], [12] recurren al modelo de forma implícita o ISM ("Implicit Shape Model") para obtener las localizaciones de las personas en base a votaciones de un conjunto de características similares a las de un diccionario previamente aprendido. Este método sólo funciona de manera adecuada para imágenes de alta resolución, obteniendo resultados muy pobres cuando la calidad de la imagen no es demasiado alta.

- Los **detectores basados en ventana deslizante** proporcionan un mejor rendimiento para baja y media resolución. Esta técnica consiste básicamente en recorrer una imagen seleccionando cada vez una determinada región local, a la que se denominará ventana, y para la cual se extraerá un descriptor de características que nos permitirá saber si dicha región delimita el cuerpo de una persona o no. Para conocer si una ventana contiene una persona, se recurrirá al uso de un clasificador previamente entrenado. Utilizando un recorrido de la ventana en todas las posiciones y escalas, se puede determinar un conjunto de detecciones válidas mediante la determinación de los máximos de densidad de detección por ventana. Uno de los primeros en utilizar este planteamiento fue Papageorgiou en [13] aplicando como características *wavelets Haar* a diferentes escalas y una máquina de soporte vectorial (SVM) como clasificador. Basados en esta idea Viola y Jones [14] construyeron un sistema que mejoraba el anterior en rendimiento y velocidad mediante el uso de imágenes integrales para el cálculo de las características y una estructura en cascada para el clasificador automatizado con *AdaBoost*.

Dentro del planteamiento de detección basada en ventana deslizante, la selección de unos descriptores de características adecuados es fundamental para obtener unos resultados aceptables. A modo de clasificación, las características utilizadas en la detección de personas se pueden organizar en un conjunto de categorías en función de la información que proporcionan. Así, las clases básicas de las características serán: las **basadas en histogramas de orientación de gradientes (HOG)** [15], [16], [17], [18], [19], las características de forma [20], [21], [22], [23], **las de movimiento** [24], [25], [26], **las compuestas** y, **las basadas en articulaciones y partes**. De entre todas ellas, HOG [16] ha resultado ser la que a nivel más básico ha proporcionado mejores resultados y sirve actualmente como referente para los recientes sistemas de detección.

2.2.1.1 Descriptores de características: Histograma de gradientes orientados (HOG)

Para abordar cualquier tarea de detección, se necesita una descripción adecuada del objeto a detectar. Para conseguir esta descripción se utilizan características que definan por completo al objeto. Estas características se agrupan en un vector que se conoce como **descriptor de características**. Hay que tener en cuenta que diferentes características serán adecuadas para diferentes aplicaciones. La clave está en hacer una elección adecuada de las características: serán mejores aquellas que acentúen las diferencias entre clases y que eliminen las diferencias dentro de una misma clase.

Los **descriptores basados en histogramas de orientación**, tales como **SIFT** [15] y **HOG** [16], han demostrado ser especialmente útiles en la determinación eficiente de las características de forma de las imágenes. Al ser técnicas que no tienen en cuenta los valores absolutos de intensidad, les permite una mayor invarianza a brillos, sombras y demás cambios de iluminación. Además, presentan un gran nivel de detalle en bordes y texturas de la imagen, al mismo tiempo que permiten cambios moderados de posición. En concreto, HOG [16] ha resultado ser el método que a nivel más básico ha proporcionado mejores resultados y sirve actualmente como referente para los recientes sistemas de detección.

Histograma de Gradientes Orientados: HOG

Como se describe en [1], el descriptor **HOG** fue desarrollado por Navneet Dalal [16] como un método de codificación eficiente de imágenes que permite la extracción de aquellas propiedades generalizables a una clase.

El descriptor HOG se basa en la idea de que la apariencia local de una imagen puede ser caracterizada de manera adecuada por una distribución de gradientes o direcciones de las aristas a nivel local, incluso con un conocimiento pobre de su posición. Tomando esto como partida, se desarrolla un tipo de descriptor consistente en histogramas locales de orientación de los gradientes de regiones solapadas y distribuidas de una imagen, o de una región de la misma, a la cual se denomina *ventana de detección*.

El proceso de extracción de los descriptores HOG se basa en evaluar un conjunto normalizado de histogramas locales de orientación de los gradientes. La figura 2.3 muestra la cadena de procesamiento completa del algoritmo de extracción de los descriptores HOG. Las etapas que el proceso requiere se encuentran descritas a continuación.

1. **Normalización color/gamma.**

   Consiste en la aplicación de una ecualización de la imagen local previa al resto de procesos, cuyo objetivo es reducir la influencia de la iluminación. El modelo utilizado como mecanismo de normalización consiste en una compresión gamma aplicada bien a cada uno de los canales de color o a la imagen en escala de grises. Este paso, no obstante es opcional, ya que no siempre proporciona mejoras a la hora de utilizar estos descriptores en tareas de detección.

2. **Cálculo de gradientes.**

   Se trata de un paso fundamental a la hora de obtener mejores o peores resultados en la clasificación basada en descriptores HOG. El objetivo de la etapa es capturar la información de contorno y silueta de la imagen mediante el cálculo de los gradientes de primer o segundo orden píxel a píxel. La manera de proceder, en el caso de disponer de imágenes con canales de color, es la asignación a cada píxel del gradiente de mayor norma de sus canales. Como paso previo al cálculo de los gradientes, se puede aplicar un filtrado gaussiano, aunque normalmente no sólo no proporciona mejoras, sino que puede ser contraproducente. De hecho, la realización más efectiva ha demostrado ser la más básica, es decir, sin filtrado gaussiano y aplicando máscaras unidimensionales de primer orden de la forma [-1 0 1]. La justificación más lógica a estos resultados es que la imagen contiene esencialmente información en las aristas y estas quedan peor definidas al aplicar métodos más elaborados.

3. **Codificación de los histogramas locales de orientación.**

   La formación de los histogramas locales constituye el paso no lineal fundamental de la extracción del descriptor HOG. La codificación propuesta para las características es localmente sensible al contenido de la imagen, aunque tolera pequeños cambios de pose o apariencia. El planteamiento para agrupar localmente la información de orientación de los gradientes es similar a la adoptada en el descriptor SIFT [15]. La ventana de detección se divide en pequeñas regiones espaciales, llamadas celdas. Para cada celda, se acumula un histograma unidimensional de orientación de los gradientes de todos los píxeles contenidos en dicha celda. Este histograma a nivel de celda es la representación básica del descriptor HOG. Cada histograma de orientación divide el rango de ángulos de los gradientes en un número predeterminado de contenedores. La magnitud de los gradientes de los píxeles de la celda se utilizan para votar en el histograma de orientación.

4. **Normalización del bloque.**

   La intensidad de los gradientes tiene un amplio rango de variación debido a los cambios locales de iluminación y, al contraste entre el objeto y el fondo. La etapa de normalización del bloque permite mejorar la invarianza a la iluminación y al sombreado, así como el contraste de los bordes. Como paso inicial en la normalización, se calcula la energía acumulada en un grupo local de celdas, al que se denominaría bloque. La energía acumulada se utiliza para normalizar cada celda del bloque. Normalmente cada celda individual es compartida por varios bloques; sin embargo, su normalización depende del bloque de pertenencia y, por lo tanto, difiere al calcularlo de un bloque a otro. Este uso de información redundante en apariencia mejora el rendimiento del sistema. Algo a tener en cuenta es que el término *histograma de orientación de gradientes (HOG)* hace referencia justamente a estos descriptores de bloque normalizados. Existen diferentes esquemas para la normalización de los bloques. Sea $\mathbf{v}$ el descriptor de características sin normalizar; $\|\mathbf{v}\|_k$ la norma Lk para $k \in 1, 2$ y ϵ una constante de normalización pequeña y positiva, que evita dividir entre cero. Los esquemas de normalización propuestos son:

   - El basado en la **normaL2**, definido como

   $$\mathbf{v}_{L2} = \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + \epsilon^2}} \tag{2.1}$$

   - El basado en la **norma L2 truncada** o **norma L2-Hyst**. Consiste en realizar una normalización basada en la norma L2 seguida de un truncamiento (limitando los valores máximos de $\mathbf{v}$ a 0,2) y una renormalización. Este tipo de normalización fue propuesta por Lowe [15].

- El basado en la **norma L1**, definido como

$$\mathbf{v}_{L2} = \frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon} \tag{2.2}$$

- El basado en la **raíz cuadrada de la norma L1**. Consiste en realizar una normalización basada en la norma L1 seguida de la aplicación de la raíz cuadrada. Se define como

$$\mathbf{v}_{L2} = \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}} \tag{2.3}$$

5. **Agrupación de los descriptores.**

   El paso final consiste en concatenar los descriptores HOG de todos los bloques que generalmente se superponen entre sí mediante una rejilla que cubre toda la ventana de detección. El descriptor final contendrá una descripción de la ventana de detección completa y será utilizado en el clasificador posteriormente.

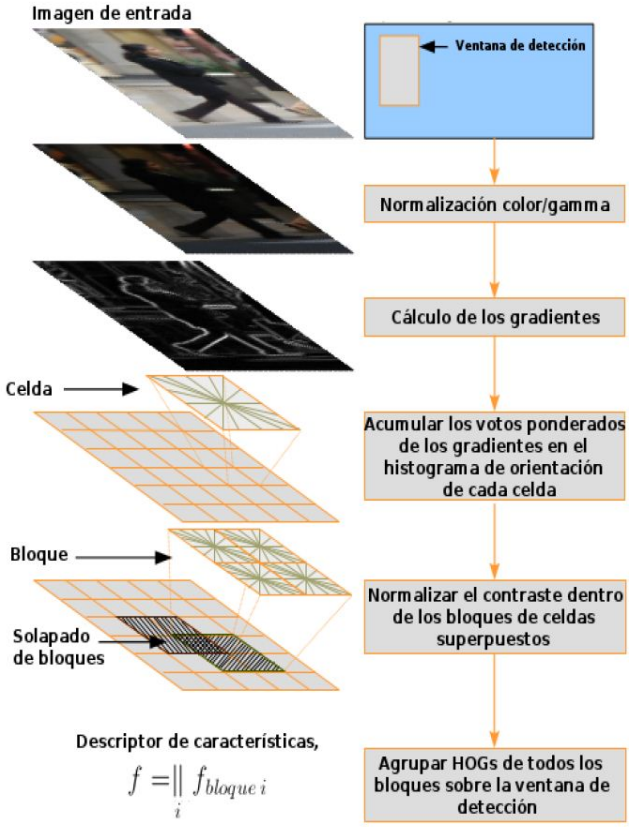


Figura 2.3: Proceso de extracción de los descriptores HOG.

Como se comentó anteriormente, el conjunto de características HOG proporcionan una descripción completa y redundante de la ventana de detección. Algo a tener en cuenta es que existirán tantas variantes de descriptores HOG como formas haya de seleccionar los bloques a normalizar. Cada una de estas variantes presentará una codificación diferente y dispondrá de sus propios parámetros clave; pero por lo demás, todas tendrán la misma cadena de procesamiento explicada (figura 2.3). En lo que sigue se se explica los distintos tipos de descriptores HOG en función de la **disposición espacial de las celdas** que forman los bloques. Una revisión más extensa puede encontrarse en [1].

- **HOG Rectangular (R-HOG).**

  Los bloques del descriptor **HOG rectangular** se definen sobre una rejilla cuadrada o rectangular. Este descriptor es el que se considera por defecto al tratar con HOG. Los bloques se definen como una rejilla de $C_V \times C_H$ celdas (los subíndices V y H hacen referencia a la dimensión vertical y horizontal de la rejilla, respectivamente); cada una de ellas de $P_V \times P_H$ píxeles. Dicha rejilla contiene Y contenedores de orientaciones.

- **HOG circular (C-HOG).**

  En los bloques del **HOG circular**, las celdas se disponen en una rejilla de forma logarítmica-polar. La imagen de entrada se convierte utilizando una rejilla rectangular de centros. En cada centro se divide la región local en un determinado número de contenedores de ángulos y radios. Los contenedores angulares se distribuyen uniformemente sobre toda la región que determina un círculo, mientras que los contenedores radiales se calculan sobre una escala logarítmica, de manera que se incrementa el tamaño del contenedor conforme aumenta la distancia al centro. Esto implica que una mayor cantidad de píxeles son tomados en las celdas más alejadas del centro.

- **HOG centro-periferia.**

  Este descriptor, evita el cálculo de bloques de tamaño fijo y los sustituye por una región de interés denominada *centro* y sus regiones vecinas denominadas periferia. De este modo, se pueden definir bloques con disposiciones y tamaños diferentes, a diferencia de las variantes anteriores de HOG.

2.2.2 Clasificación usando máquinas de soporte vectorial ("Support Vector Machines")

Un clasificador es el método mediante el cual, a partir de un descriptor de características, se determina la clase a la que pertenece la región de la imagen analizada. Los clasificadores pueden trabajar con dos (binarios) o más clases (multiclase). Cuando se habla de detección de personas, el clasificador con el que se trabaja es binario. Únicamente distinguirá entre las clases *persona* y *no persona*.

Como se explica en [1], los clasificadores se entrenan haciendo uso de un conjunto de ejemplos etiquetados de modo que se pueda determinar una división entre las posibles clases de pertenencia. El conjunto de entrenamiento a elegir es una tarea importante ya que de él buen funcionamiento del sistema de detección depende en gran medida de cómo sea de bueno el clasificador utilizado.

En tareas de detección de objetos en imágenes, existen distintos tipos de clasificadores. Sin embargo, en el presente trabajo se hará un estudio más centrado en aquellos basados en **máquinas de soporte vectorial (SVM)**.

Las **máquinas de soporte vectorial ("Support Vector Machines, SVM")** son una técnica de aprendizaje supervisado desarrolladas en los laboratorios *AT&T* por Vladimir Vapnik y su equipo. A partir de una serie de muestras (subconjunto de un conjunto mayor, espacio), cada una de ellas pertenecientes a una de las clases posibles del espacio, mediante la técnica de **SVM** se crea un modelo que es capaz de **predecir la clase a la que pertenece** una nueva muestra (cuya clase no es conocida). Las muestras de entrada son definidas como un vector de características de dimensiones p.

La técnica SVM busca por lo tanto un **hiperplano** que separe de forma óptima los puntos de las distintas clases. Estos puntos pueden ser proyectados previamente a un espacio de dimensión superior.

En la figura 2.4, se muestran una serie de muestras pertenecientes a dos clases posibles (rojo y azul) y tres de los múltiples planos de separación entre muestras posibles.

De los tres planos indicados en la figuran 2.4, se puede comprobar que:

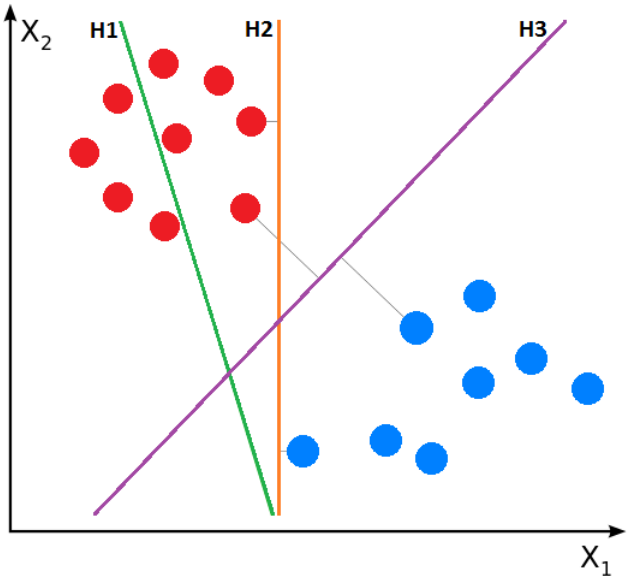


Figura 2.4: Ejemplos de posibles planos de separación en un clasificador lineal [3] .

- H_1 no logra separar correctamente los dos grupos de muestras.
- H_2 consigue separarlos correctamente. Sin embargo se puede ver como es probable que una nueva muestra puede ser confundida, ya que la distancia entre el plano H_2 y las muestras más cercanas es muy pequeña.
- H_3 separa correctamente los grupos de muestras y su distancia a las muestras más cercanas (margen), es máxima.

Este concepto de **margen máximo** es la característica fundamental de las SVM. En este tipo de algoritmos se busca un **hiperplano de separación** que tenga la máxima distancia (margen) a las muestras más cercanas a él. Es por ello por lo que las SVM también son conocidas como **clasificadores de margen máximo**.

En la figura 2.5 se presenta de forma general el funcionamiento de las SVM.

Una vez que se obtiene el hiperplano de separación las muestras a uno de los lados pertenecerá a una clase y las que estén al otro a la otra clase.

2.2.2.1 Clasificador SVM lineal

Se considera clasificador lineal óptimo a aquel cuyo hiperplano de separación entre clases, H_0, maximiza la distancia a las muestras más próximas de dichas clases. Estas muestras se conocen como **vectores soporte**, sobre los cuales se disponen dos hiperplanos, H_1 y H_2, paralelos al plano de separación H_0. La distancia que separa estos dos planos (H_1 y H_2), se conoce como **margen**. En la figura 2.6, se muestra este proceso de forma gráfica.

La capacidad de generalización de este clasificador viene marcada por el margen (distancia entre H_1 y H_2). Debido a ello, presentan menos problemas de sobreaprendizaje, que otros métodos como las redes neuronales, ya que en el entrenamiento solamente se aprenden los vectores soporte.

Por lo tanto el problema de optimización lineal, puede definirse de las siguiente manera: a partir de un conjunto de muestras o vectores de entrada $x \in \mathbb{R}^n$, pertenecientes a las diferentes clases $y \in \mathbb{N}$, se desea encontrar el hiperplano de parámetros $\{w, b\}$, que realice la mayor separación entre las clases.

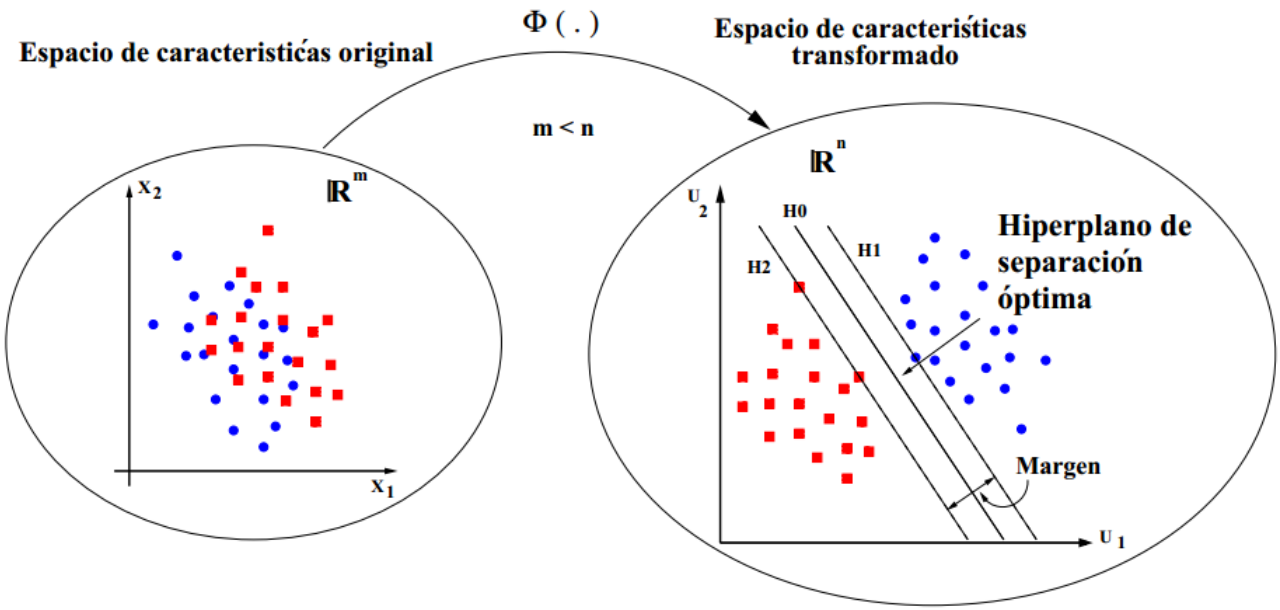


Figura 2.5: Esquema del funcionamiento general de las SVM [3].

Matemáticamente puede expresarse según la ecuación 2.4.

$$wx_i + b \lesseqgtr y_i \quad / \quad \begin{cases} y_i = -1 \quad \forall x_i \in C_i = B \\ y_i = 1 \quad \forall x_i \in C_i = A \end{cases} \Rightarrow \exists (w^*, b^*) \quad / \quad w^* x + b^* = 0 \rightarrow H_0 \tag{2.4}$$

Aunque para el desarrollo matemático solo se empleen dos clases $\{A, B\}$, puede realizarse clasificación multiclase, empleando las estrategias:

- "*One-against-all*": Se diseña una SVM por clase, entrenada para distinguir entre dicha clase y todas las demás. Se decide por maximización de las salidas.
- "*One-against-one*": Diseño de $\frac{1}{2}C(C-1)$ clasificadores binarios. Cada uno distingue entre cada par de clases. Se decide el resultado mediante votación. En caso de empate, decide el clasificador de las clases empatadas, o bien mediante "*one-against-all*".

De todos los hiperplanos de clasificación, el mejor de ellos es aquel cuya distancia de separación a las muestras o vectores vecinos de ambas clases es máxima. Este hiperplano se denomina H_0 y es para el que el margen es máximo. El margen es calculado como la distancia entre H_1 y H_2, paralelos a H_0 y que contienen al menos un vector de cada clase, denominados vectores soporte (Figura 2.6).

El cálculo matemático de H_1 y H_2, se realiza mediante las expresiones de 2.5.

$$\begin{cases} H_1 \quad : \quad wx - b = 1 \\ H_2 \quad : \quad wx - b = -1 \end{cases} \tag{2.5}$$

Si los vectores a la entrada son linealmente separables, es posible obtener dos planos H_1 y H_2, entre los que no haya muestras y maximizar la separación entre ellos.

La distancia entre el hiperplano y el origen del sistema viene dada por la ecuación 2.6.

$$d = \frac{b}{|w|} \tag{2.6}$$

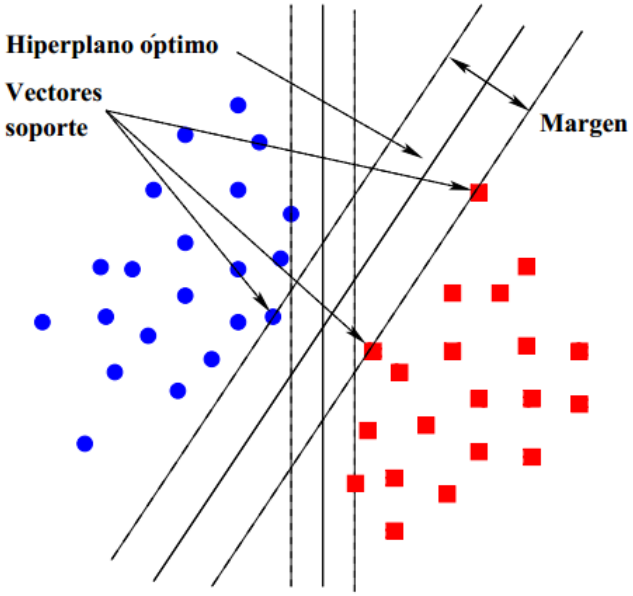


Figura 2.6: Selección del hiperplano óptimo [3].

Teniendo en cuenta que el vector w es perpendicular al hiperplano que define y la ecuación 2.6, se puede obtener la distancia entre H_1 y H_2, según la ecuación 2.7.

$$d = \frac{2}{|w|} \tag{2.7}$$

En la figura 2.7, se muestra de forma gráfica un caso de clasificación, indicando los distintos hiperplanos y las diferentes distancias definidas.

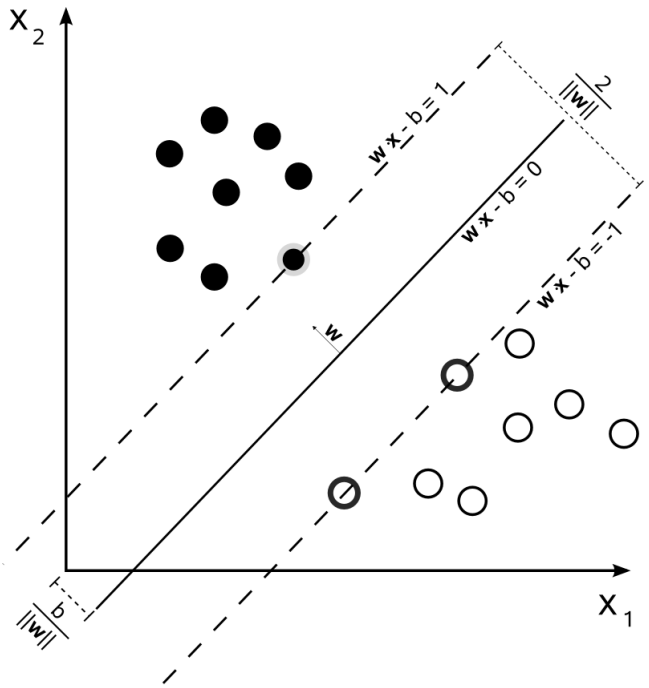


Figura 2.7: Hiperplano de separación óptimo, con distancias indicadas [3].

Con la ecuación 2.7, se puede comprobar como maximizar la separación entre H_1 y H_2 equivale a minimizar $|w|$. Además habrá que evitar que haya muestras o vectores en la región entre ambos hiperplanos,

ecuación 2.8.

$$y_i(wx_i - b) \geq 1, \quad 1 \leq i \leq n \tag{2.8}$$

Lo que da lugar al siguiente problema de optimización: minimizar $\Omega(w) = \frac{1}{2}|w|^2$, considerando la restricción $y_i(wx_i - b) \geq 1, \quad 1 \leq i \leq n$.

Para su resolución, se llega a la función lagrangiana de la ecuación 2.9 (empleando multiplicadores de *Lagrange*).

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}w^T w - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - 1] \tag{2.9}$$

El desarrollo de la ecuación 2.9, da lugar al sistema de ecuaciones 2.10. Dicho sistema es resoluble si los datos de entrada son linealmente separables, obteniéndose un mínimo global.

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} & \rightarrow \quad w = \sum_{i=1}^{N} \alpha_i y_i x_i \\ \frac{\partial \mathcal{L}}{\partial w} & \rightarrow \quad w = \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{2.10}$$

El sistema 2.10, da lugar a la ecuación 2.11.

$$w^T w = w^T \sum_{i=1}^{N} \alpha_i y_i x_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.11}$$

Sustituyendo la ecuación 2.11 en 2.9, se obtiene que la función $Q(\alpha)$ a maximizar es la que se presenta a continuación (ecuación 2.12).

$$J(w, b, \alpha) = Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.12}$$

Con la restricción $\sum_{i=1}^{N} \alpha_i y_i = 0$, para $\alpha_i \geq 0, \quad i = {}^{o}, \ldots, N$. Siendo los α_i correspondientes a los vectores soporte, los únicos distintos a cero.

Por lo tanto el hiperplano óptimo de separación H_0, depende únicamente de los vectores soporte del "set" de entrenamiento. Una vez obtenidos los valores de α_i^*, se obtienen los coeficientes del hiperplano, según la ecuación 2.13.

$$\alpha_i \Rightarrow w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i \rightarrow b^* = 1 - w^{*T} x_s \tag{2.13}$$

2.2.2.2 Margen relajado ("Soft Margin")

En el apartado anterior (sección 2.2.2.1), no se permitía que hubiera vectores mal clasificados. Es decir, que en la región entre H_1 y H_2 no ha de haber ningún vector. Esta condición es muy exigente para la aplicación en muchos casos prácticos, por ello se propuso emplear "soft margin" (margen blando).

Consiste en establecer un margen máximo, para que en el caso de no poder obtener un hiperplano H_0 de separación óptimo, se elegirá aquel hiperplano que separando las muestras o vectores lo mejor posible, permita errores de clasificación al mismo tiempo que maximice la distancia al resto de muestras correctamente clasificadas (Figura 2.8).

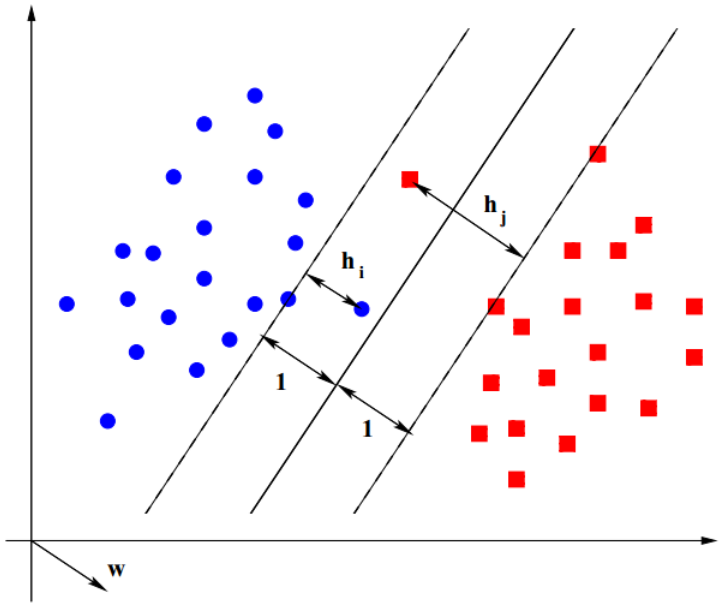


Figura 2.8: SVM aplicando "soft margin" [3].

Para ello se introducen un conjunto de variables "de holgura" $h_i, i = 1, \ldots, N$, que permitan modular la "rigidez" del criterio de clasificación de cada vector (Ecuación 2.14).

$$y_i(w^T x_i + b) \geq 1 - h_i, \quad i = 1, \ldots, N \tag{2.14}$$

Donde:

$$h_i \geq 0 \ \ \forall i \Rightarrow \begin{cases} 0 \leq h_i \leq 1 \ \ \rightarrow clasificaci\acute{o}ncorrecta \\ h_i > 1 \ \ \rightarrow clasificaci\acute{o}nincorrecta \end{cases} \tag{2.15}$$

Esto origina que en la nueva función a optimizar, $\Omega(w, h)$, aparezca un término que sirva para hacer balance entre el mayor margen y el error. Para el caso en el que la función de penalización sea lineal, se obtiene que la función a minimizar con "soft margin" es $\Omega(w, h) = \frac{1}{2}|w|^2 + C\sum_{i=1}^{N} h_i$, con la restricción $y_i(wx_i - b) \geq 1, \quad 1 \leq i \leq n$.

Siguiendo un desarrollo análogo al realizado anteriormente (sección 2.2.2.1), se obtendría que las variables de ajuste desaparecen, quedándose únicamente la constante C como restricción adicional en los multiplicadores de *Lagrange*.

Esta constante se encarga de controlar la relación de compromiso entre la complejidad del modelo y el número de errores obtenidos (cantidad de datos no separables). También es posible usar funciones de penalización no lineales, con la ventaja de poder minimizar el efecto de muestras anómalas, pero con la desventaja de aumentar la complejidad y con más dificultades de generalizar el clasificador.

2.2.2.3 Clasificador SVM no lineal

Para aquellos problemas de clasificación de conjuntos de datos no separables linealmente, en los que no es suficiente con aplicar el algoritmo de clasificación lineal descrito (sección 2.2.2.1) o la modificación de "soft margin" presentada (sección 2.2.2.2), se ha propuesto un tipo de clasificador no lineal.

Esto se obtiene mediante la transformación de los conjuntos de datos de entrada a conjuntos de datos en un espacio de características transformado (mediante el uso de una función no lineal $\Phi(\cdot)$). En este

espacio de dimensión superior al original, los vectores podrán separarse linealmente, empleando el método de separación mediante hiperplano óptimo H_0. Por lo que el clasificador basado en un hiperplano en el espacio de características transformado, puede ser no lineal en el espacio de características original.

El algoritmo resultante es de la misma forma que el descrito anteriormente (sección 2.2.2.1), sustituyendo el producto escalar original por una función de kernel no lineal K.

En la ecuación 2.16, se muestra la expresión matemática de la transformación no lineal $\Phi(\cdot)$, de los datos de entrada, a un espacio de dimensión superior en el que poder realizar una separación lineal de los datos transformados.

$$\mathbb{R}^n \xrightarrow{\Phi} \mathbb{R}^m, \quad m > n \tag{2.16}$$

En la figura 2.9, se muestra de forma esquemática el proceso explicado anteriormente.

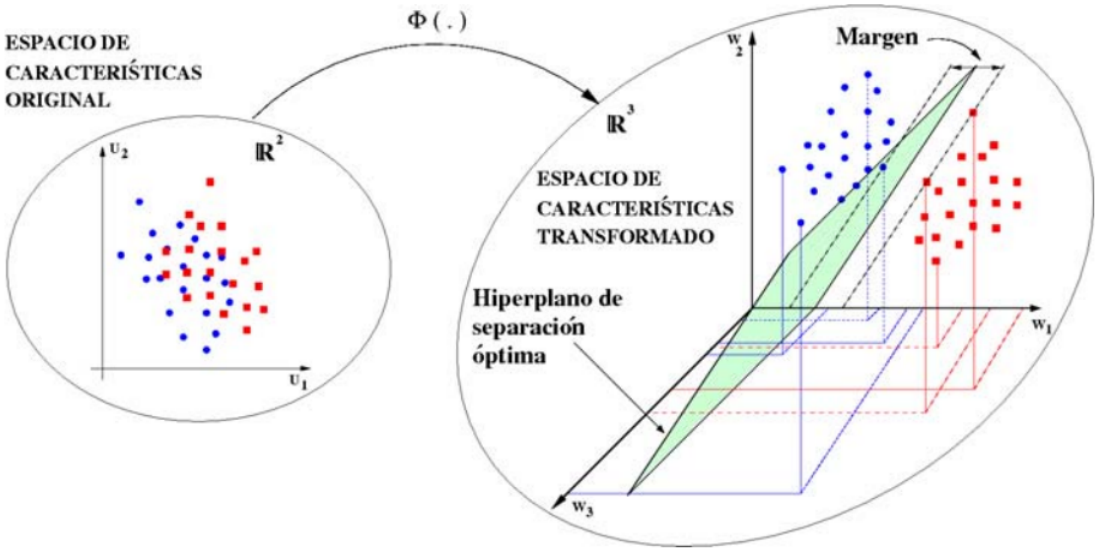


Figura 2.9: Esquema general para la clasificación no lineal [3].

Teniendo en cuenta el procedimiento anterior y la técnica de "soft margin", se obtiene una función objetivo a minimizar $\Omega(w, h)$, equivalente a la del caso anterior.

$$\Omega(w, h) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} h_i \tag{2.17}$$

Considerando las restricciones equivalentes de "soft margin", la función de transformación $\Phi(x)$, sustituye al vector x, obteniéndose la expresión 2.18.

$$\begin{aligned} &y_i(w^T \Phi(x_i) + b) \geq 1 - h_i, \quad i = 1, \ldots, N \\ &h_i \geq 0 \quad \forall i \end{aligned} \tag{2.18}$$

Pudiendo obtener los parámetros $\{w, b\}$ del hiperplano, en el nuevo espacio transformado $\mathbb{R}^m$.

La transformación explícita, $\Phi(x)$, de cada dato es costosa para m elevados. Por ello se emplean las denominadas funciones Kernel K.

El equivalente, a la ecuación a maximizar del caso lineal, para la clasificación no lineal se presenta en la ecuación 2.19.

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j) \tag{2.19}$$

Por lo tanto, el objetivo es encontrar una función en la que el producto escalar de las transformaciones $\Phi(x)$ de cada dato, sea igual a la transformación del producto escalar de los datos (ecuación 2.20). Esto implica que no es necesario transformar explícitamente cada uno de los datos de entrada ni conocer de forma exacta la expresión de la función de transformación $\Phi(x)$.

$$K(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{2.20}$$

Para conocer si una función Kernel cumple la propiedad del producto escalar (2.20), se emplea el *teorema de Mercer* (ecuaciones 2.21, 2.22). Sin embargo, este teorema no indica la forma de construir las funciones $\Phi(x)$.

$$K(u, v) = \sum_{m}^{\infty} a_m \Phi_m(u) \Phi_m(v), \quad a_m \geq 0 \tag{2.21}$$

$$\int \int K(u, v) g(u) g(v) du dv > 0, g \in L_2 \tag{2.22}$$

En la tabla 2.1, se presentan algunos de los tipos de funciones Kernel más empleados.

Tipo MSV	Función kernel	Comentario
Polinómica	$(x^T y + 1)^p$	El valor de p lo especifica el usuario
RBF Gaussianas	$e^{-(\frac{\|x-x_i\|^2}{2\sigma^2})}$	El valor de σ lo especifica el usuario y es común a todas las funciones kernel
RBF Exponeciales	$e^{-(\frac{\|x-x_i\|}{2\sigma^2})}$	El valor de σ lo especifica el usuario y es común a todas las funciones kernel
Perceptrón de 2 capas	$tanh(\beta_0 x^T x_i + \beta_1)$	Sólo ciertos valores de β son válidos
Serie de Fourier	$\frac{sin(N+\frac{1}{2})(x-y)}{sin(\frac{1}{2}(x-y))}$	El valor de N lo especifica el usuario
Splines	$\sum_{r=0}^{N} x^r y^r + \sum_{s=1}^{N} (x-\tau_s)_+^k (y-\tau_s)_+^k$	El orden de la curva k y el número de los N puntos localizados en τ_s lo especifica el usuario
B splines	$B_{2N+1}(x-y)$	El valor de N lo especifica el usuario

Tabla 2.1: Tipos de funciones Kernel [3].

Por último, hay que indicar que las funciones compuestas por sumas y/o productos de otras funciones Kernel, también son funciones Kernel.

2.3 Seguimiento de Personas

A pesar de las mejoras introducidas en los detectores, en el caso de los vídeos, la solución "frame" a "frame" no produce resultados completamente satisfactorios. Debido a ello, se hace necesario el uso de un sistema de seguimiento que mejore las detecciones obtenidas y a su vez proporcione un mecanismo de asociación con objetivos detectados previamente. Este problema se aborda con los llamados **sistemas de seguimiento mediante detección**. Para conseguir esto, es necesario recurrir a algún modelo de observación que permita combinar las informaciones de detección, seguimiento y apariencia específica de cada objetivo.

En su forma más simple, la tarea de seguimiento se puede entender como la obtención de la estimación de la trayectoria de un objeto según se desplaza por la escena. Además, el sistema seguidor podría aportar información sobre otros aspectos como orientación, área o forma del objeto.

Para dar solución al problema de seguimiento de un objeto, existen diferentes **métodos probabilísticos de estimación (estimadores)**. Algunos de los más utilizados son: **mínimos cuadrados recursivos (LMS)**, **filtro de Kalman (KF)** o **filtro de partículas (PF)** [4].

En las posteriores secciones de este apartado, se explicará el filtro de Kalman, pues ha sido el estimador utilizado en este trabajo.

2.3.1 Filtro de Kalman

El filtro de Kalman trabaja con modelos en variables de estado. Como cualquier estimador, el objetivo básico es obtener la estimación de un vector de estado variante en el tiempo a partir de observaciones con incertidumbre de dicho vector [4].

Dicho filtro, estima el vector de estado del instante siguiente de un proceso dinámico y discreto, mediante el siguiente conjunto de ecuaciones en diferencias:

$$\hat{\mathbf{x}}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n+1] + \mathbf{w}[n] \tag{2.23}$$

$$\hat{\mathbf{z}}[n+1] = \mathbf{H}\mathbf{x}[n+1] + \mathbf{v}[n+1] \tag{2.24}$$

Las ecuaciones 2.23 y 2.24 forman el modelo de sistema y su modelo de observación respectivamente, donde:

- $\mathbf{w}[n]$ y $\mathbf{v}[n]$ representan el ruido de proceso y el ruido de medida respectivamente. Ambos independientes, blancos y con una distribución de probabilidad normal (ruido blanco gaussiano) de modo que permitan al filtro alcanzar una solución estable:

$$p(\mathbf{w}) \approx N(0, \mathbf{Q}) \tag{2.25}$$

$$p(\mathbf{v}) \approx N(0, \mathbf{R}) \tag{2.26}$$

- $\mathbf{A}(r \times r)$: Matriz de actualización de estado en función del estado anterior.
- $\mathbf{B}(r \times l)$: Matriz de actualización de estado en función de la entrada.
- $\mathbf{H}(s \times r)$: Matriz que relaciona el estado con la medida.

Para el caso general, las matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{Q}$ y $\mathbf{R}$ se consideran constantes (propios de sistemas invariantes y ruidos estacionarios). La estimación por el filtro de Kalman se basa en un algoritmo recursivo con dos etapas: **predicción** y **corrección** (figura 2.10).

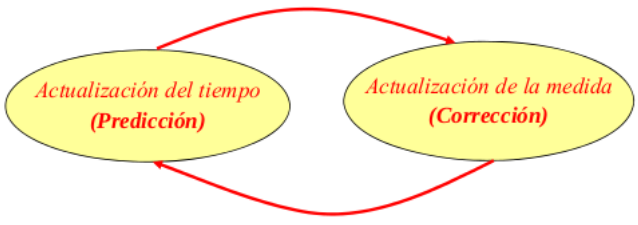


Figura 2.10: Etapas del filtro de Kalman [4].

Las ecuaciones de la etapa de predicción, proyectan hacia delante el estado actual y error de covarianza estimados para obtener su estimación a priori para el nuevo paso, mientras que las ecuaciones de la etapa de corrección incorporan una nueva medida en la estimación a priori para obtener una estimación mejorada a posteriori.

En este punto conviene explicar la nomenclatura empleada en lo que sigue, incluida la figura 2.11. Se definirá $m_{n+1/n}$ como la estimación de la variable m en el instante $n+1$ a partir de la información existente en el instante n y $m_{n+1/n+1}$ como la estimación de la variable m en el instante $n+1$ a partir de la información existente en el instante $n+1$, su corrección con la información de observación.

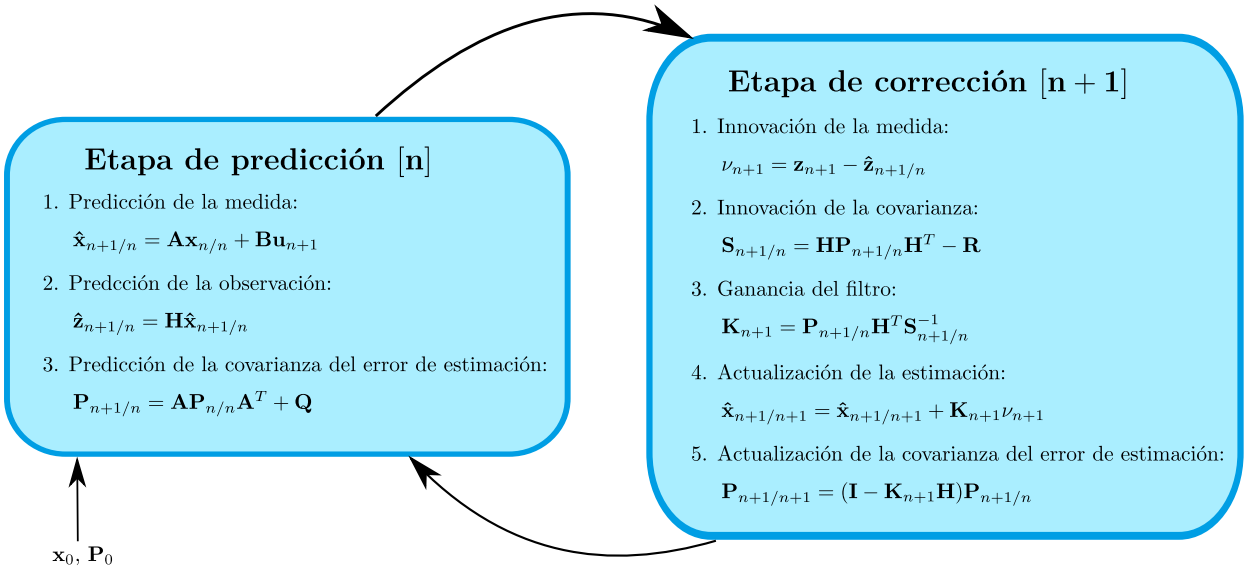


Figura 2.11: Ecuaciones de las etapas del filtro de Kalman.

En la **etapa de predicción** (en el instante n) se obtiene el valor del vector de estados estimado ($\hat{\mathbf{x}}_{n+1/n}$) empleando para ello el modelo matemático del sistema. Además se obtiene la matriz de covarianza del error de estimación a priori ($\mathbf{P}_{n+1/n}$) que incluye la matriz de covarianza del error del sistema $\mathbf{Q}$, incorporando así, el error de modelado.

La **etapa de corrección** (instante $n+1$) comienza con la obtención de la ganancia de Kalman a partir de las matrices $\mathbf{P}_{n+1/n}$, $\mathbf{H}$ y $\mathbf{R}$. Con ella se corrige la estimación del estado ($\hat{\mathbf{x}}_{n+1/n+1}$) del sistema y finalmente se actualiza la matriz de covarianza del error de estimación ($\mathbf{P}_{n+1/n+1}$). Merece la pena destacar que este paso consiste en fusionar la información obtenida por el modelo del sistema con la información externa de la medida.

De este modo, quedaría detallado el funcionamiento del estimador. No obstante, conviene indicar cómo se debe realizar la inicialización del sistema. Como se desprende de la figura 2.11, sería necesario conocer el valor inicial del vector de estado ($\mathbf{x}_0$) y de la matriz de covarianza del error de predicción ($\mathbf{P}_0$). Si bien la primera suele ser fácil de fijar, la segunda no lo es tanto. Existen varias técnicas para asignarle un valor inicial:

- Si se conoce $\mathbf{x}_0$, $\mathbf{P}_0$ será nulo, ya que representa la incertidumbre del valor de estimación inicial.
- Si no se conoce $\mathbf{x}_0$, $\mathbf{P}_0$ deberá fijarse en función de su significado, pero en cualquier caso una técnica muy recurrida es la de ejecutar el estimador "off-line" (mediante una simulación) y ajustarlo así por prueba y error.
- Si no se conoce $\mathbf{x}_0$ y $\mathbf{P}_0$ no se ajusta en simulación, conviene darle valores elevados para que el filtro pueda "engancharse".

En cualquier caso, si el sistema es lineal e invariante, el filtro de Kalman suele converger siempre (más o menos rápidamente) independientemente del valor de $\mathbf{P}_0$. Será por tanto más crítica la elección de este parámetro a la hora de implementar el estimador para sistemas no lineales.

En lo que a las matrices $\mathbf{Q}$ y $\mathbf{R}$ se refiere, al ser la covarianza del error de sistema y de medidas se pueden obtener fácilmente, al menos de forma aparente:

$$\mathbf{Q} = E[\mathbf{w}_n \cdot \mathbf{w}_n^T] \tag{2.27}$$

$$\mathbf{R} = E[\mathbf{v}_n \cdot \mathbf{v}_n^T] \tag{2.28}$$

Aunque $\mathbf{v}_n$ y $\mathbf{w}_n$ aparecen dependientes del tiempo, Q y R suelen considerarse constantes.

Sin embargo, si bien la matriz $\mathbf{R}$ puede obtenerse de forma empírica realizando medidas "off-line" de la salida y obteniendo dicho parámetro estadístico, no ocurre de igual modo con la matriz $\mathbf{Q}$, ya que no suele ser habitual tener acceso a las variables de estado del sistema . En caso de no poder obtener $\mathbf{Q}$ de forma matemática se fijará por tanteo (generalmente también mediante sucesivas pruebas del estimador realizadas "off-line"). En ese caso es importante tener en cuenta ciertos aspectos sobre el valor de estas matrices:

- La relación de magnitud entre $\mathbf{Q}$ y $\mathbf{R}$ es indicativa de la fiabilidad de los dos elementos de medida o modelado. Así, si $\mathbf{Q} < \mathbf{R}$ significa que el modelo proporciona más fiabilidad sobre el vector de estado que la matriz de salida. En ese caso la matriz $\mathbf{P}$ tomará valores pequeños, así como la ganancia de Kalman. Y si $\mathbf{Q} > \mathbf{R}$ significará que el modelo es menos fiable que los sensores de salida ($\mathbf{P}$ y $\mathbf{K}$ serán pequeñas en la evolución del algoritmo).
- Ambas matrices son cuadradas y diagonales (pues el ruido no ha de estar autocorrelado, ha de ser blanco). En caso de que los experimentos de obtención de las matrices devuelvan algún valor no nulo fuera de la diagonal principal el ruido será coloreado, y será necesario emplear alguna técnica para corregir este aspecto. Si alguna de las fuentes de ruido presenta fuentes sistemáticas, la media de estas variables aleatorias no será nula, por lo que será necesario eliminarlos de algún modo. Ambos aspectos se verán de forma más detallada en el capítulo siguiente al especificarlos para las fuentes sensoriales de interés en este trabajo.
- Finalmente, las dos fuentes de ruido ($\mathbf{v}_n$ y $\mathbf{w}_n$) han de estar incorreladas entre sí.

2.3.1.1 Modelo de velocidad constante

Para los casos en los que se está estimando la posición de un objeto que se desplaza con la misma velocidad, (o aparentemente la misma) se puede asumir un modelo en que la velocidad sea constante (**modelo de velocidad constante**). En este caso, se incluye dentro del vector de estados la velocidad del objeto (aunque puede que no sea observable). Para los casos en los que no se puede asegurar que la velocidad siempre vaya a ser constante, se añade al sistema un vector de ruido para caracterizar las variaciones de velocidad del objeto. Las ecuaciones en este caso particular quedan como se muestra en 2.29 y 2.30.

$$\hat{\mathbf{x}}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{w}[n] \tag{2.29}$$

$$\hat{\mathbf{z}}[n+1] = \mathbf{H}\mathbf{x}[n+1] + \mathbf{v}[n+1] \tag{2.30}$$

Este modelo también se puede encontrar con el nombre de **modelo de segundo orden**. Hacer uso del filtro de Kalman caracterizando el sistema con dicho modelo, permite obtener el valor de la velocidad del objeto seguido.

2.3.2 Proceso de asociación

Para implementar un sistema de seguimiento de múltiples objetos, al margen del proceso de estimación de la posición, es necesario definir un proceso de asociación que permita relacionar cada una de las medidas recibidas (**hipótesis**) junto con las estimaciones posición del objeto obtenidas por el estimador. Esto es, asignar las nuevas detecciones a sus correspondientes seguidores existentes, para de este modo, saber que se está siguiendo al mismo objeto.

La asociación de datos es un problema importante y complejo en aplicaciones de seguimiento. Importante porque la asociación de alguna hipótesis con el objeto incorrecto puede hacer divergir la estimación de la posición de ese y el resto de los objetos; complejo porque el algoritmo ha de buscar la solución óptima entre todas las posibles.

Las propuestas para resolver el problema de asociación son variadas. En este apartado se describirán de manera breve algunas de las más utilizadas.

- **Seguidor de múltiples hipótesis (MHT, "Multiple Hypotheses Tracker")**.

  Se trata de un algoritmo de asociación determinístico que calcula las posibles soluciones de asociación que se generan en cada instante de llegada de nuevas hipótesis. Supone que cada medida puede proceder de alguno de los objetos seguidos. Este algoritmo conlleva una gran carga computacional pues ha de ejecutarse por cada solución generada por el MHT.

- **Vecino más próximo (NN, "Nearest Neighbour")**.

  Este algoritmo puede verse como una simplificación del MHT en la que sólo se mantiene una hipótesis por cada objeto. En concreto, relaciona la hipótesis con la estimación de un objeto más cercana. Dicho método, es utilizado en problemas de asociación sencillos o situaciones en las que se precise de una carga computacional baja.

- **Asociación de datos probabilística (PDA,"Probabilistic DATA Association")**.

  Consiste en un algoritmo probabilístico que obtiene la probabilidad de que una hipótesis esté relacionada con alguna estimación.

Capítulo 3

Desarrollo

Por mucho que sople el viento, una montaña nunca se arrodillará a su paso.

Mulán

El sistema implementado en este trabajo tiene la estructura que se muestra en la figura 3.1.

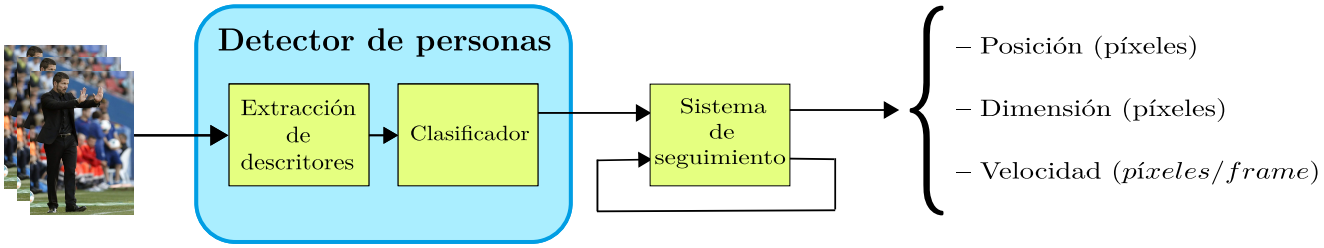


Figura 3.1: Sistema implementado.

Dicho sistema, recoge una secuencia de imágenes (vídeo) que es analizada una por una (análisis "frame" a "frame"). El detector de personas, mediante el sistema de extracción de descriptores y el clasificador, determina si existen personas y en qué lugar de cada imagen. Después, el sistema de seguimiento tiene la función de seguir a la misma persona o personas a lo largo de la secuencia.

En los siguientes apartados se describe cómo se ha implementado cada uno del los bloques que forman el algoritmo de detección y seguimiento de personas. Además, se justifican las diferentes decisiones de diseño.

En rasgos generales, es importante destacar que todo el sistema se ha implementado haciendo uso del lenguaje de programación C++, y en la medida de lo posible, utilizando funciones de la librería de visión computacional *OpenCV*.

3.1 Detector de personas

El sistema detector de personas (sección 2.2) trabaja con dos módulos: el **extractor de características** y el **clasificador**. El módulo que obtiene las características de la imagen emplea para ello el método de la ventana deslizante con descriptores HOG. Para el clasificador se utiliza una SVM lineal entrenada para la detección de personas.

Este sistema recibe a su entrada la imagen ("frame"), el extractor de características realiza el análisis HOG de la ventana de detección dentro de la imagen con el fin de obtener sus descriptores. El vector de características obtenido se entrega al clasificador, que los comparará con el vector obtenido en el entrenamiento para determinar si la zona analizada en la ventana corresponde con una persona o no.

Es importante destacar que la imagen es analizada a distintas escalas, es decir, la imagen de entrada al detector de personas será una versión escalada de la real. Esto permite detectar personas de diferentes tamaños. Es importante cuando se tienen escenas con personas situadas a distintas distancias de la cámara.

Esta configuración para el diseño del detector de personas fue propuesta por [16]. La razón por la que elegir esta configuración es que actualmente, para el entorno de trabajo descrito, es la que mejor resultados ofrece. Una descripción más detallada de los módulos mencionados se realiza en los posteriores subapartados. Además, un diagrama cómo funciona el detector de personas descrito se presenta en 3.2

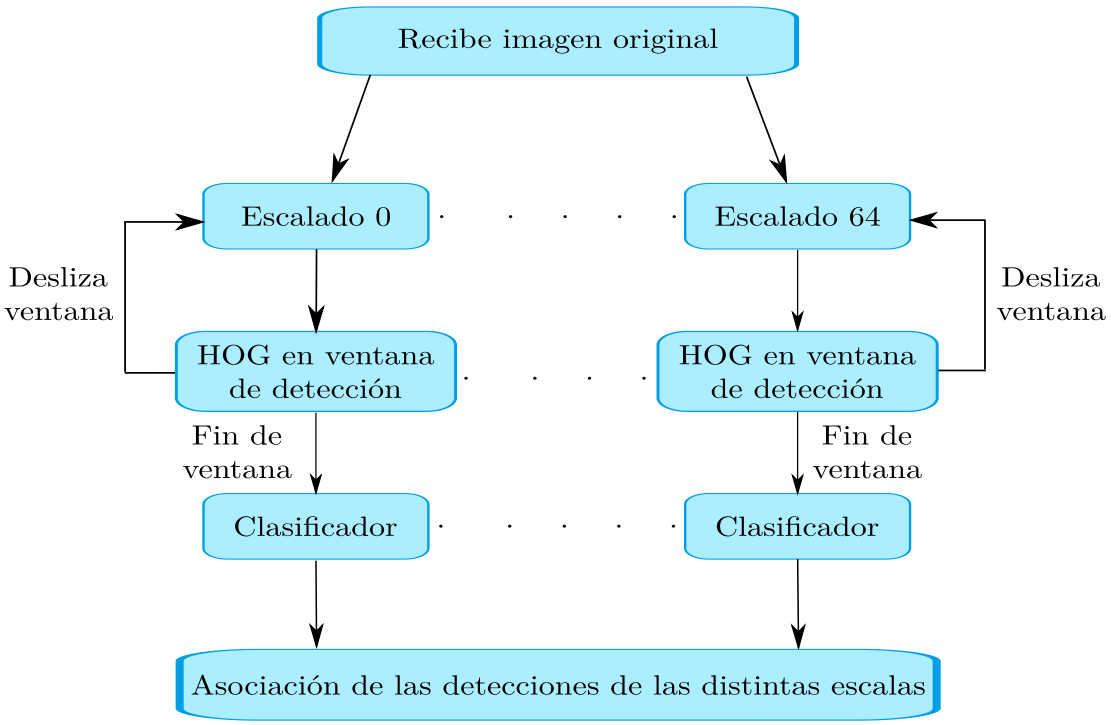


Figura 3.2: Diagrama de bloques del funcionamiento general del detector de personas.

3.1.1 Cálculo de los descriptores HOG

Como se ha comentado, para extraer las características de la imagen se hace uso del método de la ventana deslizante con descriptores HOG. Para trabajar con este método se acude a las funciones que *OpenCV* proporciona. En concreto, la mencionada librería, desarrolla el algoritmo propuesto por [16] para la extracción de descriptores.

Dichas funciones permiten implementar un detector con una ventana deslizante de 64×128 píxeles. La ventana a su vez se divide en bloques de 16×16 píxeles que contienen celdas de 8×8 píxeles. Los bloques se pueden solapar a la mitad, es decir, un nuevo bloque trabaja con la mitad de las celdas del bloque anterior. La configuración descrita hace que la ventana sea de 7×15 bloques (105 bloques por ventana) y que a su vez los bloques sean de 2×2 celdas (4 celdas por bloque).

El algoritmo obtiene para cada celda el módulo y la orientación de los gradientes y los ordena en un

histograma de nueve intervalos. El rango total del histograma es de 0° a 180°, además está normalizado aplicando L2 truncado. Teniendo en cuenta esto y las relaciones entre celdas bloques y tamaño de ventana, se puede deducir que el vector de características generado tendrá 3780 valores. Este vector, será el dato de entrada al siguiente bloque: el clasificador. El proceso que realiza el algoritmo es el descrito en la sección 2.2.1.1 y mostrado en la figura 2.3. También en la figura 3.3 se puede observar un ejemplo de como se distribuye la ventana (en verde), el bloque (en rojo) y la celda (en amarillo).

Figura 3.3: Imagen real (aumentada) con la distribución de venta, bloque y celda.

3.1.2 Clasificador SVM lineal

Para realizar la tarea de clasificación, al igual que para el cálculo de descriptores, también se emplea *OpenCV*. En concreto, el detector hace uso del clasificador para detección de personas que implementa dicha librería.

El clasificador está basado en una máquina de soporte vectorial (SVM) lineal (sección 2.2.2.1) de margen relajado (sección 2.2.2.2). Además, ha sido entrenado Para ventanas de detección de 64×128 píxeles con la base de datos de *INRIA* (**INRIA Person Dataset**)[5]. La base de datos mencionada, consiste en un conjunto de imágenes de 64×128 píxeles adecuadamente centradas y normalizadas, válidas para entrenar y comprobar un sistema de detección de personas. Un ejemplo del tipo de imágenes que son se puede ver en la figura 3.4.

El hecho de que el clasificador se entrene para el tamaño de ventana de detección anteriormente dicho, hace que su vector de coeficientes tenga 3780 valores. Se puede comprobar que el clasificador tiene el mismo número de coeficientes que el vector de características que se obtiene en el cálculo de los descriptores (sección 3.1.1), de este modo, se puede calcular la distancia al hiperplano de separación de las clases.

(a) Ejemplo 1

(b) Ejemplo 2

Figura 3.4: Ejemplos de imágenes de entrenamiento de la base de datos de INRIA [5].

El clasificador proporciona a su salida la distancia al hiperplano de la SVM (peso de detección) además de las características del "bounding box" que definen el tamaño y posición de la persona o personas: coordenadas de la esquina superior izquierda y alto y ancho. Los tres parámetros expresados en píxeles. Un ejemplo se muestra en la figura 3.5 expuesta a continuación.

Figura 3.5: Especificaciones del "bounding box".

3.1.3 Asociación de las detecciones de las distintas escalas

Al hacerse un análisis de la imagen a diferentes escalas, pueden darse detecciones correspondientes a diferentes personas en las distintas escalas, o bien, la misma persona puede ser detectada en varias escalas. Cuando ocurre esto, es necesario fusionar todas las detecciones para presentarlas en la imagen original. Esto significa que, las detecciones únicas tienen que ser escaladas al tamaño que tendrían en la imagen real y las detecciones en distintas escalas, necesitan ser escaladas y después asociadas de modo que sólo se muestre como una única detección.

En este trabajo se aplica el algoritmo disponible en la librería de *OpenCV* que obtiene todos los "bounding box" de todas las detecciones en todas las escalas además de el peso de su peso de detección

asociado. Tras esto, aplica un algoritmo de particionamiento mediante el cual, reconoce en la imagen todos los conjuntos de "bounding box" que hacen referencia a la misma persona y se queda con el más representativo y con el peso de detección mayor.

3.2 Sistema de seguimiento

Como se explica en la sección 2.3, los sistemas de seguimiento se utilizan tanto para robustecer el sistema completo de detección como para tener un histórico de la trayectoria en la escena que sigue el objeto seguido. En tareas de detección como las que requiere este trabajo, un sistema de seguimiento debe resolver los siguientes problemas:

- **Validación de medidas.** Normalmente, cuando la detección corresponde con un persona, suele mantenerse a lo largo de la secuencia, pudiendo trabajar el seguidor con su trayectoria. Sin embargo, Puede haber ocasiones en las que el detector, de manera inesperada y en un sólo un "frame" de la secuencia, localice una persona en un lugar de la escena donde no corresponde. Al no poder ser seguida porque la detección no se mantiene en el tiempo, el seguidor será el encargado de ignorar este error.

- **Oclusión.** El fenómeno de oclusión se produce cuando un elemento de la escena se interpone entre el objeto seguido y la cámara. Cuando pasa esto a lo largo de la trayectoria de un objeto seguido, por un período de tiempo, el detector no entregará medidas de su localización, por tanto, el seguidor tratará de trabajar con las estimaciones de la posición hasta que vuelva a ser detectado.

- **Cruce de objetos.** Cuando dos objetos que están siendo seguidos se cruzan en la escena, las medidas que el detector proporciona de cada uno de ellos se pueden confundir. Para evitar esto, el sistema seguidor de cada objeto tendrá de algún modo la posibilidad de obtener la dirección en la que se desplaza, en consecuencia, podrá solventar la confusión.

Se ha creído conveniente trabajar con seguidores basados en el **filtro de Kalman** (sección 2.3.1) debido a que ofrece buenos resultados en este tipo de trabajos al mismo tiempo que tiene un coste computacional no muy alto y su implementación es relativamente sencilla. En concreto, se emplea el **modelo de velocidad constante** (sección 2.3.1.1) ya que se supone que la variación de la velocidad de movimiento de las personas a lo largo de su paso por la escena es reducida. Condición que permite asumir dicho modelo.

Es importante destacar que en la escena, como caso habitual, se encontrarán múltiples personas a seguir. Existen métodos que trabajan con un sólo estimador para para predecir la posición de todos los objetos. No obstante, el filtro de Kalman en su forma más básica, como es este caso, no es capaz de conseguir eso. Esto obliga que cada persona en la escena tenga su propio filtro, conformando así un banco de filtros de Kalman. Al trabajar de esta manera, el sistema está obligado a controlar los siguientes momentos posibles del banco.

- **Creación de seguidores**. Cuando aparece en la escena una nueva detección, el sistema seguidor tiene que ser capaz de elegir el momento adecuado para iniciar el seguimiento (lanzar un filtro) de esa nuevo objetivo aparecido.

- **Asociación de medidas con seguidores**. Cuando el detector de personas encuentra un objetivo, entregará sus respectivas medidas de posición al bloque seguidor. Este último, debe entregar las medidas al filtro que está siguiendo a ese objetivo (Proceso de asociación).

- **Iteración de seguidores**. El algoritmo debe iterar cada filtro existente cumpliendo con las etapas de predicción y corrección. En caso de que haya una persona desaparecida por oclusión, por algún fallo del detector o por abandono del campo de visión, y por tanto, no se tenga medidas observadas de ella, el algoritmo seguirá manteniendo sus estimaciones hasta que vuelva a ser encontrada en la escena.

- **Eliminación de seguidores**. Cuando un objetivo ha desaparecido de la escena y no va a volver a aparecer (no existen nuevas medidas observadas), se ha de finalizar el estimador asociado a dicha persona.

Los valores de las matrices **Q** y **R** se consideran constantes y son los mostrados en 3.1 y 3.2. Las diagonales de las matrices corresponden con las varianzas del ruido de proceso y el ruido de medida respectivamente, cuyos valores reales no se conocen. Experimentalmente, se ha visto que con los valores elegidos los filtros funcionan correctamente. Por ello todos los filtros del banco de filtros utilizarán estos valores.

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.2}$$

3.2.1 Definición de los modelos de sistema y observación

Tal como se determina en la sección 3.1, la salida del detector de personas es la caja que delimita la persona ("bounding box"), definida por su alto, ancho y las coordenadas de la esquina superior izquierda en píxeles. Con estos datos, hay que encontrar las variables que formarán el vector de estado del con el que trabajará el filtro de Kalman.

Si se sabe el ancho y el alto además de la coordenada de la esquina superior izquierda, es sencillo calcular las coordenadas del centroide del rectángulo. El filtro de Kalman hace uso de ese centroide y de sus variaciones (velocidades) para realizar el seguimiento de la persona. Las coordenadas del centroide son dadas en píxeles y la velocidad se expresa en $\frac{píxeles}{frame}$. Conocido esto, el modelo de sistema queda definido por la ecuación 3.3:

$$\hat{\mathbf{x}}_{\mathbf{n+1}} = \begin{bmatrix} x_{n+1} \\ y_{n+1} \\ v_{x_{n+1}} \\ v_{y_{n+1}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ v_{x_n} \\ v_{y_n} \end{bmatrix} + \mathbf{w}_n \tag{3.3}$$

Por tanto, se entiende que el filtro, en su etapa de predicción, es capaz de estimar la posición de la persona y la velocidad con la que se desplaza. Para la etapa de corrección, el seguidor necesita conocer las nuevas medidas (observaciones) que llegan del sensor en el instante siguiente para compararlas con sus estimaciones del instante anterior. Las medidas únicamente observables son las de posición. Debido a esto, la ecuación que define el modelo de sistema es la mostrada en 3.4.

$$\hat{\mathbf{z}}_{\mathbf{n+1}} = \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \\ v_{x_{n+1}} \\ v_{y_{n+1}} \end{bmatrix} + \mathbf{v}_n \tag{3.4}$$

3.2.2 Creación de un nuevo filtro

Cuando se va a iniciar un filtro de Kalman para alguna detección, se debe tener cierta seguridad de que el objetivo que se va a comenzar a seguir corresponde con una persona, es decir, el algoritmo de seguimiento debe ser capaz de entender que lo que está siguiendo es una persona y no un fallo esporádico del sistema detector. Para solucionar este problema, existen al menos dos alternativas posibles.

1. La primera alternativa consiste en retrasar el inicio del filtro hasta que se haya podido relacionar varias detecciones consecutivas en la escena. La mayor ventaja de aplicar esta alternativa es que al no lanzar filtros que más adelante tuvieran que ser eliminados, se reduce el coste computacional. Sin embargo, se puede entender como inconveniente el hecho de que si el filtro sigue un objetivo real, cuanto antes se hubiera lanzado, antes se tendría controlado.

2. La segunda solución consiste en iniciar el filtro con la primera detección que se tenga. En el caso de que el filtro se haya iniciado para un objetivo correcto, se habría ganado tiempo. No obstante, si tras pasado un pequeño espacio de tiempo se deduce que no es una persona, habría que eliminar un filtro que no hubiera sido necesario crear.

Basándose en las ventajas e inconvenientes que ofrece uno método frente a otro, en este trabajo se ha optado por implementar la primera alternativa.

Al iniciar un filtro de Kalman para seguir a la persona, es necesario determinar los valores iniciales de $\mathbf{x}_0$ y $\mathbf{P}_0$ (sección 2.3.1). El vector de estado necesita las coordenadas de la posición de la persona y las velocidades de desplazamiento en cada eje. Las coordenadas se las proporciona el detector. Como para iniciar un filtro, el sistema ha tenido que comprobar durante varios instantes ("frames") que existen varias medidas observadas de una misma persona, en el momento de lanzar su filtro, es posible conocer las velocidades.

Es importante destacar que para estos sistemas que trabajan con píxeles, las medidas de posición y las de velocidad son bastante ruidosas. Por tanto, si al vector de estado inicial se le asignan medidas que tienen incertidumbre, la matriz $\mathbf{P}_0$ no puede ser 0 para la primera estimación. Como no se puede conocer correctamente su valor, se ha dado un valor alto, comprobando experimentalmente que inicializando $\mathbf{P}_0$ de esta forma el filtro converge. La matriz $\mathbf{P}_0$ se muestra en 3.5.

$$\mathbf{P}_0 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \tag{3.5}$$

En el instante de creación del filtro, se realiza la primera predicción del vector de estado para que al siguiente instante el filtro ya funcione en los pasos de corrección y predicción. El proceso de buscar

medidas para iniciar sus seguimientos se resume en el algoritmo 3.1.

Data: D : Conjunto de detecciones sin iniciar seguimiento;
S : Conjunto de seguidores activos;
ns : Nuevo seguidor;
f : Asociado no asociado con detecciones sin seguimiento del instante anterior (f = 1, tiene medidas asociadas. f = 0, no tiene);
$intIniCont$: Contador de instantes de asociación sin seguimiento;
$maxintIniCont$: número máximo de instantes de asociación sin seguimiento para considerar que ya se puede iniciar el filtro;

```
begin
    for i ∈ D do
        f = ComprobarSiAsociado(i);
        if f = 1 then
            intIniCont = ObtenerContador(i);
            if intIniCont < maxintIniCont then
                IncrementarContador(i);
            else
                IniciarFiltro(i, ns);
                S ⟵ ns;
                Prediccion(ns);
        else if f = 0 then
            DescartarMedida(i);
```

Algoritmo 3.1: Creación de filtros

3.2.3 Asociación de datos

El proceso de asociación es una de las tareas básicas del bloque seguidor. En este trabajo se implementa un algoritmo de asociación basado en el vecino más próximo, estudiado en la sección 2.3.2. El algoritmo calcula la distancia euclídea entre dos datos para primero ver si esa distancia es menor que un umbral de cercanía.

En un instante dado, en el sistema puede haber coordenadas de detección del instante anterior y estimaciones de los distintos filtros del banco de filtros. En ese mismo instante se reciben las nuevas medidas. El algoritmo de asociación, para cada medida, calcula la distancia euclídea con todas las hipótesis posibles. Tras esto, compara las distancias con el umbral de cercanía. En este punto, toda hipótesis que cumpla ese umbral es considerada candidato. Cuando están todos los candidatos posibles, se elige al más

cercano. La propuesta descrita se presenta en el algoritmo 3.2.

Data: m : Medida recibida;
D : Conjunto de hipótesis posibles (estimaciones y detecciones sin iniciar seguimiento del frame anterior);
u : Umbral de distancia;
Result: Candidato asociado
begin
  $distmin = u$;
  for $i \in D$ **do**
    $dist = DistanciaEuclidea(m, x)$;
    if $dist < distmin$ **then**
      $candidato = i$;

Algoritmo 3.2: Búsqueda del candidato final

Al realizar el algoritmo para todas las medidas recibidas, el sistema consigue diferenciar aquellas que proceden de personas que ya están siendo seguidas por un filtro de Kalman de las que proceden de detecciones que no tienen que iniciar el filtro aun y con ello poder trabajar correctamente con cada una de ellas.

3.2.4 Iteración de seguidores

Una vez que el algoritmo de asociación ha terminado de asociar las medidas recibidas, cada filtro puede estar en dos situaciones.

1. El filtro tiene asociadas nuevas medidas a sus estimaciones.

2. El filtro no tiene medidas asociadas a sus estimaciones.

El primer caso es el caso común cuando la persona está siendo seguida. En esta situación, el filtro debe realizar la corrección de su estimación. Tras esto, realiza una nueva predicción del vector de estado. Durante este proceso de dos fases se van ajustando las matrices **P** y **K**. Tras varios ciclos normales del filtro, dichas matrices convergerán a su valor final. Este proceso matemático se encuentra descrito en la sección 2.3.1.

Cuando se da el segundo caso, puede ser que la persona seguida se haya ido de la escena o haya desaparecido tras algún objeto (oclusión) y el filtro que esta estimando su posición no disponga de medidas observadas durante algunos instantes de tiempo ("frame"). En estas situaciones, el estimador no dispone de datos para realizar la etapa de corrección, por tanto, únicamente realizará la etapa de estimación con las medidas anteriormente estimadas. Al no tener etapa de corrección, la matriz **P** aumenta su valor, indicando la creciente inexactitud de las medidas estimadas. Cuando la persona vuelva a ser encontrada, se volverá a realizar el proceso normal (corrección, predicción).

El proceso para realizar esta tarea queda expuesto en el algoritmo 3.3.

Data: S : Conjunto de filtros activos;
f : Asociado no asociado (f = 1, tiene medidas asociadas. f = 0, no tiene);
begin
  for $i \in S$ **do**
    $f = ComprobarSiAsociado(i)$;
    if $f = 1$ **then**
      $Correccion(i)$;
      $Prediccion(i)$;
    else if $f = 0$ **then**
      $Prediccion(i)$;

Algoritmo 3.3: Iteración de los filtros

3.2.5 Eliminación de filtros

En el apartado 3.2.4 se describe el caso en el que un filtro no tiene observación asociada. En los supuestos descrito (abandono temporal de la escena y oclusión), la persona vuelve a ser detectada. No obstante, cuando a un filtro no se le asocia ninguna medida puede ser porque la persona haya salido para siempre del campo de visión.

Cuando ocurre esta situación, el sistema de seguimiento esta programado para que itere el filtro durante unos instantes (varios "frames"). Si en esos instantes al filtro no se le vuelve asociar ninguna medida, significa que la persona ya no está en la escena. En consecuencia, el filtro será eliminado.

Con esta nueva situación descrita, el algoritmo 3.3 se hace más completo quedando como el que se muestra a continuación en el algoritmo 3.4.

Data: S : Conjunto de filtros activos;
f : Asociado no asociado (f = 1, tiene medidas asociadas. f = 0, no tiene);
$insCont$: Contador de instantes sin tener medidas asociadas.;
$maxCont$: máximo número de instantes sin tener medidas asociadas.;
begin
  for $i \in D$ **do**
    $f = ComprobarSiAsociado(i)$;
    if $f = 1$ **then**
      $Correccion(i)$;
      $Prediccion(i)$;
    else if $f = 0$ **then**
      $insCont = ObtenerContador(i)$;
      if $insCont < maxCont$ **then**
        $prediccion(i)$;
      else
        $ElimanarFiltro(i)$;

Algoritmo 3.4: Iteración de los filtros incluyendo eliminación

Capítulo 4

Resultados

Mira Simba, toda la tierra que baña la luz es nuestro reino.

El Rey León

En este capítulo se presentan los resultados obtenidos para el trabajo descrito. Las secuencias de imágenes utilizadas para probar el sistema desarrollado forman parte de la base de datos de **CAVIAR Project**[2]. Los vídeos de esta base de datos han sido grabados en un centro comercial de Portugal, con una calidad de imagen de 384×288 píxeles a 25 "frames" por segundo. Además de esta base de datos, el sistema se ha probado con secuencias grabadas con una cámara de mayor calidad, en concreto, la cámara *GoPro HERO3*[27]. Los vídeos adquiridos con esta cámara, han sido grabados con gran angular(lente curvada), con una resolución de 1280×720 píxeles y a 50 "frames" por segundo.

En los siguientes apartados se podrá comprobar el funcionamiento de cada bloque del sistema por separado, mostrando tiempos de cómputo y ejemplos de "frames" de algunas secuencias. De este modo se puede analizar el efecto que tienen en el sistema completo los distintos bloques que lo forman.

Para el análisis de tiempos de ejecución, las pruebas experimentales se han realizado en un ordenador con procesador *Intel Core 2*, y 2GB de memoria RAM.

4.1 Análisis y resultados del detector de personas

Como se ha explicado en 2.2, el detector de personas se encarga de localizar las personas en las imágenes. Para ello, trabaja con múltiples escalas y en dos fases: extrae características y clasifica (sección 3.1). En las figuras 4.1 y 4.2 se muestran varias situaciones que se pueden dar tras la fase de detección.

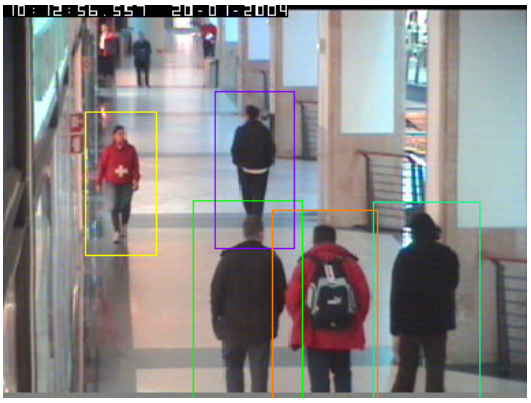
En el ejemplos 4.1a, 4.1b y 4.2a, se puede ver como el detector ha sido capaz de obtener la posición de todas las personas existentes en la escena. Esta situación, es la ideal, no obstante, por errores del detector o por fenómenos de oclusión puede haber casos en los que no se obtengan todas las localizaciones de las personas y además existan detecciones que corresponden con zonas en las que no hay persona. Esto queda mostrado en 4.1c, 4.1d y 4.2b.

Cabe destacar los colores que tienen las cajas que rodean a las personas, los cuales están relacionados con los pesos (distancia al hiperplano) que ha obtenido el clasificador para una determinada detección. Como ya se comentó, cuanto mayor sea el peso, mayor es la probabilidad de que la detección sea correcta.

Los colores utilizados son (ordenados de menor peso a mayor peso): rojo, naranja, amarillo, verde, azul, magenta. En verde y azul se utilizan tres tonos.

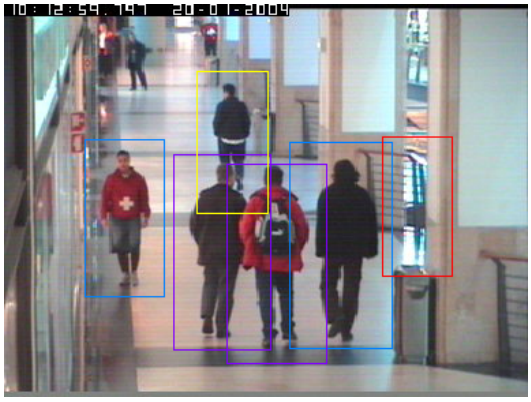
Si se observan las figura 4.1c, 4.1d y 4.2b, las detecciones que no corresponden con nada relevante en la imagen están marcadas en color rojo, es decir, tienen muy poco peso. Esto es lo que suele suceder con este tipo de errores: ocurren en un "frame" esporádico y su peso es muy pequeño. Sabiendo esta información, se puede dar solución a este caso en la etapa de seguimiento.

(a) Ejemplo 1.



(b) Ejemplo 2.

(c) Ejemplo 3.



(d) Ejemplo 4.

Figura 4.1: Distintas situaciones posibles en la fase de detección (CAVIAR Project [2]).

Además de todo esto, es importante analizar el tiempo de cómputo del algoritmo de detección de personas. Respecto al tiempo de cómputo, existen tres aspectos muy importantes que influyen en él: la resolución de imagen, el número de escalas analizadas y la calidad del hardware utilizado.

- **Resolución de la imagen**. Cuanto más grande sea la imagen, más tiempo estará deslizándose la ventana de detección, lo que implica un mayor tiempo de ejecución.
- **Número de escalas analizadas**. Escalar la imagen significa cambiar su resolución. El detector de personas trabaja con varias escalas de cada "frame" de manera que, estará trabajando con distintas resoluciones de imagen. De esto se deduce que cuanto más escalados realice el algoritmo, mayor será su tiempo de ejecución.
- **Calidad del hardware utilizado**. Es muy importante remarcar que los tiempos de ejecución del algoritmo de detección están fuertemente ligados con la calidad del hardware sobre el que se ejecute la aplicación. Siempre que la calidad sea mayor, los tiempos de ejecución serán menores.

(a) Ejemplo 1.

(b) Ejemplo 2.

Figura 4.2: Distintas situaciones posibles en la fase de detección (GoPro).

Hay que recordar que la función de la librería *OpenCV* que se utiliza en este trabajo para realizar la fase de detección, integra la extracción de características (HOG enventanado a distintas escalas) y la clasificación (clasificador SVM lineal). Por tanto, el tiempo que se puede analizar es el del sistema de detección completo.

Para analizar este tiempo, se ha hecho funcionar el programa con 5 secuencias de vídeo de aproximadamente 300 "frames" cada uno de ellos, tanto de la base de datos de *CAVIAR Project*[2] como los vídeos de GoPro. Se ha guardado el tiempo que ha tardado la fase de detección en cada "frame" de cada vídeo para después hacer la media. El tiempo medio de la fase de detección con la base de *CAVIAR Project*[2] y con GoPro para el ordenador utilizado en este proyecto son $t_{m_{detecCAVIAR}} = 187{,}63ms$ y $t_{m_{detecGoPro}} = 624{,}56ms$ respectivamente.

4.2 Análisis y resultados del sistema de seguimiento

En este trabajo, se ha hecho una descripción completa de cuáles son las características y los beneficios de añadir a un sistema de detección de personas un bloque de seguimiento (secciones 2.3 y 3.2). Para desarrollar las tareas de seguimiento se implementado un banco de filtros de Kalman aplicando el modelo de velocidad constante.

Para poder comprobar el correcto funcionamiento del sistema de seguimiento habrá que determinar si la implementación del filtro de Kalman es correcta y si el sistema de seguimiento aporta soluciones al mayor número de errores posibles que pueda tener el detector de personas en un instante dado.

4.2.1 Funcionamiento del modelo matemático del filtro de Kalman

Con la intención deducir si el filtro de Kalman implementado funciona correctamente, se han hecho pruebas con una secuencia en la que sólamente aparece un individuo y es detectado cláramente por el detector. Para poder determinar si el comportamiento del filtro es el adecuado es necesario observar lo siguiente:

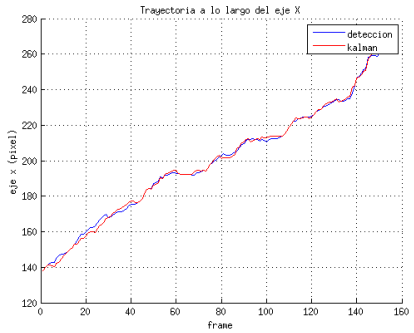
- El filtro converge. Las matrices **P** y **K** deben presentar en su respuesta un pequeño transitorio y un valor final. Esto demostrará que el filtro es capaz de estabilizarse.
- Debe actuar como filtro de los valores recogidos en detección, suavizando la trayectoria de las personas sobre la escena.

Para comparar la salida del detector con la del bloque seguidor se presenta la figura 4.3, donde se observan los datos que proporcionan los dos sistemas para un mismo "frame". En 4.3a se muestra el "bounding box"' que proporciona el detector para conocer la posición del individuo. La figura 4.3b enseña cómo el sistema seguidor está realizando un seguimiento del centroide del "bounding box". Las aspas de color rojo corresponden con los valores observados, mientras que las amarillas indican los valores estimados por el filtro. Además se puede ver que la estela de ambas aspas están casi solapadas entre sí, así que, en principio, se puede decir que el funcionamiento se ajusta a lo esperado. Lo comentado sobre las estelas se puede ver con más de detalle en las figura 4.4, la cual, contiene las trayectorias de la persona a lo largo de los ejes X e Y por separado.

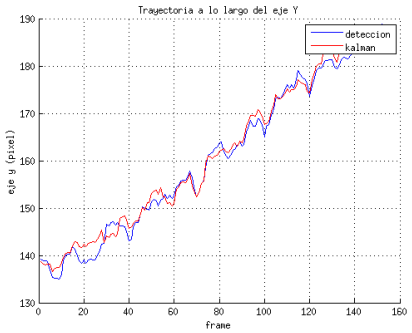
(a) Etapa de detección.

(b) Etapa de seguimiento.

Figura 4.3: Salida de los sistemas detector y seguidor para el mismo "frame".



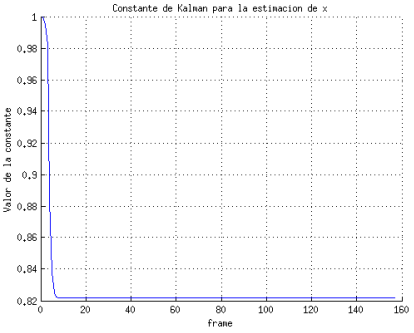
(a) Trayectoria sobre el eje X.



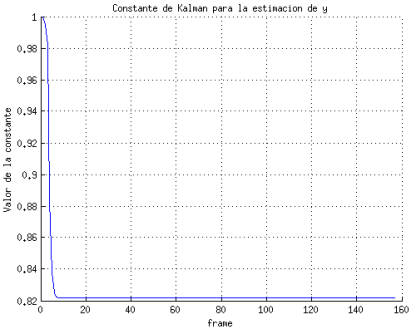
(b) Trayectoria sobre el eje Y.

Figura 4.4: Trayectorias de la persona sobre los ejes X e Y.

En las figuras 4.5 y 4.6 están mostradas las respuestas de las matrices **K** y **P**. Únicamente están los valores obtenidos para la estimación de las coordenadas de la posición. El comportamiento de los valores para la estimación de velocidad será el mismo.

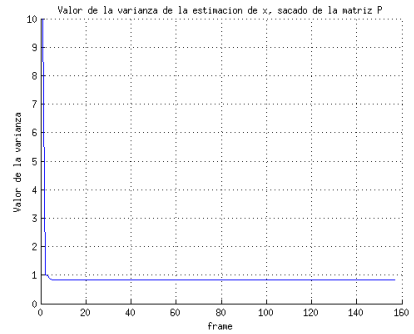


(a) Constante de Kalman para la estimación de la posición en el eje X.

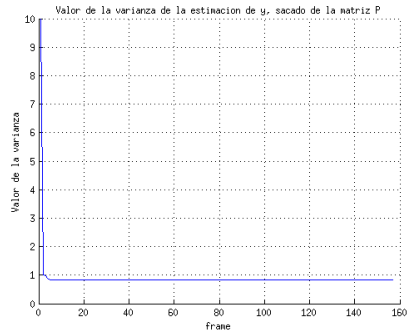


(b) Constante de Kalman para la estimación de la posición en el eje Y.

Figura 4.5: Respuesta de la constante de Kalman.



(a) Varianza del error de estimación de la posición en el eje X.



(b) Varianza del error de estimación de la posición en el eje Y.

Figura 4.6: Respuesta de la covarianza del error de estimación.

Es fácil ver que la respuesta de ambas matrices a lo largo de los "frames" es similar. Durante algunos instantes, los valores están ajustándose; esto se corresponde con el transitorio del filtro. Tras esto, las matrices alcanzan unos valores determinados y se mantienen en el tiempo. Esto indica que el filtro se estabiliza y converge.

Con estas pruebas, se consigue demostrar que el modelo matemático del filtro de Kalman implementado cumple con los requisitos anteriormente explicados y por tanto funciona adecuadamente.

4.2.2 Análisis del algoritmo de seguimiento

Una de las grandes tareas del sistema de seguimiento es robustecer el sistema completo mediante la eliminación de detecciones erróneas, continuación de detecciones perdidas, etc (sección 3.2). Con el fin de comprobar estas situaciones, se describen a continuación algunas pruebas con pequeños fragmentos de vídeo.

4.2.2.1 Validación de detecciones e inicio del filtro de Kalman

El proceso de validación de detecciones llevado a cabo por el sistema de seguimiento consiste en diferenciar entre medidas provenientes de personas en la escena y medidas erróneas del detector de personas, lo que se conoce como falsos positivos. Un ejemplo de este problema puede verse en la figura 4.7a. El detector ha marcado como persona dos elementos de la escena: uno que sí corresponde con persona y otro que corresponde con un fallo en detección. Esta información es entregada al sistema de seguimiento. Como son nuevas detecciones, este no se fiará aun de que sean verdaderas y durante un tiempo las mantendrá como provisionales. La figura 4.7b corresponde con la etapa de seguimiento. Las aspas azules significa que provisionalmente el sistema no considera válidas esas medidas. Para que esta etapa valide las medidas, estas se han tenido que mantener durante algunos instantes.

(a) Etapa de detección. (b) Etapa de seguimiento.

Figura 4.7: Estado provisional del sistema de seguimiento ante la llegada de nuevas detecciones.

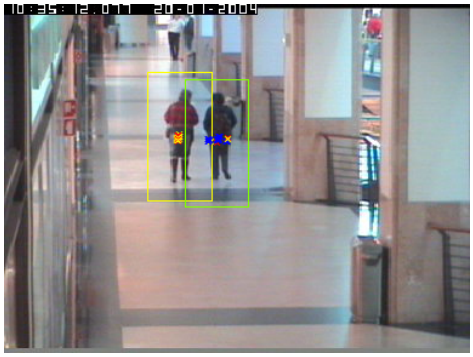
Si la medida que está en estado de espera, no permanece durante lo que la etapa de seguimiento considera aceptable, esta será descartada (no validada). Sin embargo, si se da el caso contrario, la medida será validada y por tanto se iniciará un filtro de Kalman para comenzar el seguimiento. Esta última situación se puede ver en la figura 4.8.

4.2.2.2 Continuación de detecciones pérdidas

A lo largo del seguimiento de una persona, puede ocurrir que el detector haya tenido algún error y en un momento dado no haya localizado a la persona aunque se encuentre en la escena o bien, esta esté en un fenómeno de oclusión o cruce entre personas y durante algunos instantes no está visible. En estas situaciones, la etapa de seguimiento es la encargada de solventar el error tal como se explicó en la sección

Figura 4.8: Validación de la medida e inicio del filtro de Kalman.

3.2.4 para estos casos concretos. En la figura 4.9 se pueden observar varios ejemplos en los que suceden las situaciones descritas.



(a) Detección perdida y recuperada.

(b) Instante en el que se pierde la detección.

(c) Instante en el que se pierde la detección (2).

(d) Personas cruzándose (oclusión).

Figura 4.9: Situaciones en las que el sistema de seguimiento debe continuar las detecciones a partir de sus estimaciones.

Es importante conocer que el sistema seguido muestra en azul las detecciones que fueron perdidas y solo mantenidas en estimación. La figura 4.9a presenta como gracias al algoritmo de seguimiento, la persona no ha sido perdida. En las figuras 4.9b y 4.9c es posible ver el instante en que se pierde la detección. Se puede ver un ejemplo en la figura 4.9d del momento en que dos personas se cruzan. Ambas personas siguen siendo estimadas (continuadas) para que cuando se separen puedan continuar siendo seguidas e identificadas como la misma.

4.2.3 Tiempo de cómputo del algoritmo de la etapa de seguimiento

Es importante conocer el tiempo que tarda el algoritmo de seguimiento en ejecutarse. Este tiempo incluye la fase de asociación y todas las iteraciones de cada filtro que haya en el instante dado. Por tanto, es lógico que cuantas mas personas estén siendo seguidas, más filtros habrá creados y en consecuencia el coste computacional será mayor.

En este caso, a diferencia que en el detector de personas, el tipo de base de datos utilizada no influye en el tiempo del algoritmo. Esto se debe a que, sea la calidad de imagen que sea, los filtros siempre trabajan con la misma cantidad y con el mismo tipo de datos.

Para analizar este parámetro se ha actuado de la misma manera que con la etapa de detección. Sin embargo, en vez de hacer el tiempo medio para cada base de datos, se indica el tiempo medio total (calculado con los tiempos obtenidos para las dos bases de datos). Este tiempo es de: $t_{m_{seg}} = 0{,}723ms$

4.3 Tiempo de ejecución del conjunto completo

Una vez calculados los tiempos de ejecución de las etapas, es importante estudiar el tiempo total de modo que se pueda descubrir en que etapa hay más margen de mejora. Estos tiempos calculados se comparan con los del trabajo que se tomó como punto de partida para este proyecto [1].

A continuación, en el gráfico de barras de la figura 4.10, se disponen los distintos tiempos obtenidos en las etapas analizadas. Además se compara el total de este trabajo desarrollado con el total que obtiene el trabajo referencia [1].

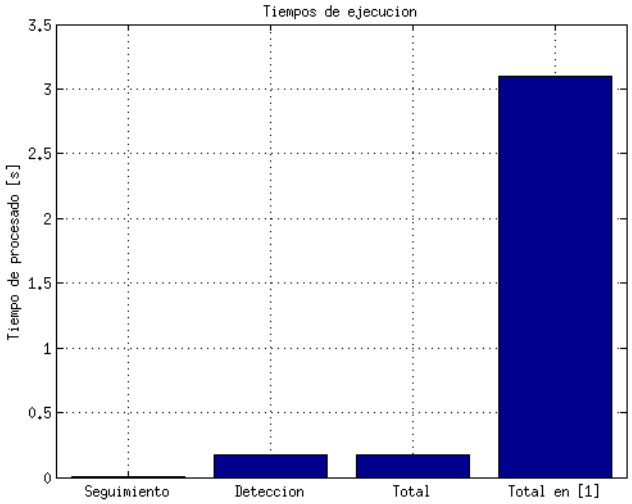


Figura 4.10: Costes computacionales por etapas y en total.

En comparación con [1], los tiempos de ejecución son mucho menores (entre 15 y 20 veces menores). Esto es un dato importante ya que el sistema implementado en este proyecto trabaja con un mayor número de escalas, lo que en un principio podría haber llevado a entender que el tiempo de cómputo sería elevado.

Con respecto a los tiempos de los algoritmos implementados, la etapa que tarda más es la de detección. Esto indica que podría ser la primera etapa a intentar mejorar.

Capítulo 5

Conclusiones y líneas futuras

Si se cree y se trabaja, se puede.

Cholo Simeone

En este capítulo se expondrán las conclusiones obtenidas tras haber analizado el trabajo aquí desarrollado. A partir de esas conclusiones se indicarán algunas líneas futuras de trabajo que puedan mejorar los resultados obtenidos.

5.1 Conclusiones

El Trabajo Fin de Grado desarrollado, ha consistido en la creación de un sistema de detección y seguimiento de personas. Para ello se ha utilizado la técnica de ventana deslizante con descriptores de características HOG, un clasificador SVM lineal y un banco de estimadores basados en el filtro de Kalman.

En primer lugar, se ha analizado el detector de personas, formado por el extractor de descriptores HOG en múltiples escalas y el clasificador SVM lineal. Es muy destacable que gracias a las múltiples escalas el detector consigue abarcar amplios campos de visión. Esta etapa ofrece a la siguiente, la posición y el tamaño de la persona encontrada. Sin embargo, hay ocasiones en las que tiene errores de detección: cuando muestra alguna localización donde no hay persona o cuando pierde en algún instante a una persona.

Con respecto al sistema de seguimiento, el análisis desarrollado sobre su funcionamiento permite concluir que consigue robustecer el sistema completo, ya que los errores de la etapa de detección consigue solucionarlos. Si es verdad, que hay algunas situaciones que aun no están bien controladas como cuando el cruce de personas en la escena no llega a ser cruce porque se quedan paradas haciendo que una tape a otra.

Para la evaluación del sistema desarrollado se han realizado múltiples pruebas experimentales tanto con imágenes pertenecientes a la base de datos de CAVIAR Project [2] (ampliamente utilizada en la literatura) como con imágenes grabadas para esta aplicación. En estas pruebas experimentales se ha podido validar el correcto funcionamiento del sistema.

Además, se han analizado los tiempos de procesamiento medios de las imágenes y se han comparado con los del trabajo tomado como punto de partida. Estos tiempos se han reducido de forma notable (siendo aproximadamente 20 veces inferiores para las mismas imágenes).

Como conclusión general, este trabajo ha servido para hacer un estudio de todas las técnicas aquí utilizadas (HOG, SVM, Asociación, Kalman), desde su base teórica hasta sus formas de implementación, valiendo así, como un punto de partida para futuros proyectos.

5.2 Líneas futuras

Tras haber realizado este trabajo, se proponen varias líneas futuras de desarrollo.

- **Obtención de métricas que permitan la evaluación cuantitativa del trabajo**. Sería imprescindible evaluar el trabajo realizado con métricas relevantes. Cotejar los resultados obtenidos con los ficheros de "ground truth" que proporcionan las bases de datos y comparar con otros proyectos.
- **Combinación de técnicas de procesamiento de imágenes**. En este trabajo, la imagen se analizado únicamente sacando los descriptores HOG. No obstante, es probable que combinando los descriptores HOG con alguna otra técnica de procesado de imagen se consigan mejores resultados.
- **Realizar nuevos entrenamientos más completos**. Con un entrenamiento más completo el sistema podría reducir su tasa de error.
- **Migrar a tecnologías más eficientes**. Debido a que, como se ha visto, el proceso de extracción de descriptores HOG a diferentes escalas es fácilmente paralelizable, se considera que la utilización de la *GPU* para la ejecución en paralelo de esa etapa puede reducir de forma notable el tiempo de procesamiento, pudiéndose llegar a ejecutar en tiempo real el proceso de detección y seguimiento.

Bibliografía

[1] J. A. C. Lozano, "Detección y reconocimiento de personas en escenas naturales mediante visión artificial," Master's thesis, Universidad de Alcalá (UAH). Escuela Politécnica Superior (EPS). Departamento de Electrónica, 2014.

[2] "Página de la base de datos de caviar projetc," http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/.

[3] J. L. M. Pérez, "Comunicación con computador mediante señales cerebrales. aplicación a la tecnología de la rehabilitación," Tesis Doctoral, ETSII, UPM, Madrid, 2009.

[4] L. M. B. Pascual, *Seguimiento en imágenes*, ser. Sistemas de Visión Computacional. Departamento de Electrónica, UAH.

[5] "Página de la base de datos de inria," http://pascal.inrialpes.fr/data/human/.

[6] R. Arroyo, J. J. Yebes, L. M. Bergasa, I. G. Daza, and J. Almazán, "Expert video-surveillance system for real-time detection of suspicious behaviors in shopping malls," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7991 – 8005, 2015.

[7] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, Septiembre 2010.

[8] J. M. Chaquet, E. J. Carmona, and A. Fernández-Caballero, "A survey of video datasets for human action and activity recognition," *Computer Vision and Image Understanding*, vol. 117, no. 6, pp. 633 – 659, 2013.

[9] J. L. Jingen Liu and M. Shah, "Recognizing realistic actions from videos ïn the wild"," in *Computer Vision and Pattern Recognition*, 2011, pp. 2268–2271.

[10] C. Gu, J. Lim, P. Arbelaez, and J. Malik, "Recognition using regions," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, vol. 2, Junio 2009, pp. 1030–1037.

[11] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, Junio 2005, pp. 878–885.

[12] K. M. Edgar Seemann, Bastian Leibe and B. Bernt Schiele, "An evaluation of local shape-based features for pedestrian detection," in *British Machine Vision Conference*, Septiembre 2005.

[13] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, Junio 2000.

[14] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, Mayo 2004.

[15] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Noviembre 2004.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, Junio 2005, pp. 886–893.

[17] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 1491–1498.

[18] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, Junio 2005, pp. 829–836.

[19] A. Shashua, Y. Gdalyahu, and G. Hayun, "Pedestrian detection for driving assistance systems: single-frame classification and system level performance," in *Intelligent Vehicles Symposium, 2004 IEEE*, Junio 2004, pp. 1–6.

[20] D. Gavrila and V. Philomin, "Real-time object detection for ldquo;smart rdquo; vehicles," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 87–93.

[21] D. Gavrila, "A bayesian, exemplar-based approach to hierarchical shape matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 8, pp. 1408–1421, Aug 2007.

[22] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 247–266, Noviembre 2007.

[23] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *Computer Vision and Pattern Recognition, 2007. CVPR 2007. IEEE Conference on*, Junio 2007, pp. 1–8.

[24] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Octubre 2003, pp. 734–741.

[25] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proceedings of the 9th European Conference on Computer Vision*, vol. 2, 2006, pp. 428–441.

[26] C. Wojek, S. Walk, and B. Schiele, "Multi-cue onboard pedestrian detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Junio 2009, pp. 794–801.

[27] "Página oficial de la cámara gopro," http://es.shop.gopro.com/EMEA/cameras/?gclid=CIXK49qYi8gCFUe3Gwod1usCKg.

Apéndice A

Manual de usuario

Este apéndice ofrece un manual de usuario mediante el cual se puede hacer funcionar la aplicación desarrollada en este trabajo.

Dicha aplicación ha sido programada en el lenguaje C++. Para la etapa de detección de personas se hace uso de la librería de funciones *OpenCV* en la versión 2.4.8 y para la de seguimiento se ha elaborado una librería que controla el proceso de asociación y todo lo relacionado con el banco de filtros de Kalman.

Para la puesta en funcionamiento de la librería y para ejecutar la aplicación se debe usar el sistema operativo *Ubuntu 14.04 LTS*, o superior, con un editor de texto como *Emacs*.

A.1 Estructura de la librería

La librería está contenida en dos carpetas: *DetectionAndTrackingSystem* y *makefile*. La primera contiene los archivos fuente de la librería creada y contendrá el ejecutable de la aplicación una vez se compile. La segunda carpeta contiene los archivos *makefile* necesarios para la compilación.

A.2 Compilación y creación del ejecutable

Para construir el ejecutable es necesario compilar la librería. Para ello, basta con proceder como se indica a continuación:

```
$ make
```

El archivo ejecutable creado tiene de nombre *detectionAndTracking*.

A.3 Ejecución de la aplicación

La aplicación está diseñada de manera que ofrece la opción de mostrar en la consola una descripción de su funcionamiento. Para ello, basta con ejecutarla con el atributo *-h* tal como se muestra a continuación.

```
$ ./detectionAndTracking -h
```

Tras ejecutar como se ha descrito, la consola muestra la siguiente información.

```
$ ./detectionAndTracking -h

NAME
detectionAndTracking

SYNOPSIS
./detectionAndTracking [OPTIONS] -l LIST_FILE

DESCRIPTION
It implements a human detection and tracking application. The system
has two stages: human detection and tracking. The human detection stage
is done by using the sliding window technique combined with the HOG
feature descriptor and a linear SVM based classifier. The tracking stage
consists of a Kalman filter bank in which each filter implements the
constant velocity model.

OPTIONS
-h: Print this usage message.
-s: A scoring file will be generated. If no name is indicated with option -n,
    the default name will be used.
-n SCORING_FILE: SCORING_FILE will be the name of the scoring file.
```

Para la ejecución del algoritmo de detección y seguimiento sobre una secuencia de video es necesario indicar un fichero (*LIST_FILE*) con extensión .list en el que se indiquen las rutas a cada uno de los "frames" que componen un vídeo. A continuación se dispone un fragmento del fichero *OneStopEnter1cor.list*:

/usr/share/geintra/databases/video/shoppingCentre/OneStopEnter1cor/FRAMES/OneStopEnter1cor0000.jpg
/usr/share/geintra/databases/video/shoppingCentre/OneStopEnter1cor/FRAMES/OneStopEnter1cor0001.jpg
/usr/share/geintra/databases/video/shoppingCentre/OneStopEnter1cor/FRAMES/OneStopEnter1cor0002.jpg
/usr/share/geintra/databases/video/shoppingCentre/OneStopEnter1cor/FRAMES/OneStopEnter1cor0003.jpg
/usr/share/geintra/databases/video/shoppingCentre/OneStopEnter1cor/FRAMES/OneStopEnter1cor0004.jpg
/usr/share/geintra/databases/video/shoppingCentre/OneStopEnter1cor/FRAMES/OneStopEnter1cor0005.jpg

Como ejemplo, se muestra la sentencia que permite ejecutar la aplicación para la secuencia *OneShopEnter1cor*:

```
$ ./detectionAndTracking -l OneShopEnter1cor.list
```

A.4 Resultados de la ejecución

Al ejecutarla, la aplicación muestra por pantalla cada una de las imágenes de la secuencia y sobre ella los resultados de la detección y el seguimiento. Un ejemplo de ello se puede ver en la figura A.1.

Cabe destacar los colores que tienen las cajas que rodean a las personas, los cuales están relacionados con los pesos (distancia al hiperplano) que ha obtenido el clasificador para una determinada detección. Como ya se comentó, cuanto mayor sea el peso, mayor es la probabilidad de que la detección sea correcta.

Figura A.1: Ejemplo de resultados que pueden visualizarse por pantalla.

Los colores utilizados son (ordenados de menor peso a mayor peso): rojo, naranja, amarillo, verde, azul, magenta.

Además, los resultados de la etapa de detección (número de "frame", resolución, tiempo de cómputo del algoritmo de detección, distancia al hiperplano, coordenadas de la esquina superior izquierda del "bounding box" y dimensiones del mismo) se almacenan en un fichero de texto con extensión *.log*. En la figura A.2 se presenta un ejemplo del contenido del contenido de resultados.

```
Thu Sep 24 10:29:06 2015

Frame           Resolucion      t_detec         Num_Detec       d_Hiperplano    corner_x        corner_y        ancho           alto
0               384 x 288       153.002         1               6.45294         99              74              61              122
1               384 x 288       191.946         1               5.68635         102             74              60              121
2               384 x 288       147.673         1               5.58282         103             74              60              121
3               384 x 288       152.495         1               6.26456         104             75              58              118
4               384 x 288       147.898         1               5.57451         104             75              59              118
5               384 x 288       153.094         1               5.92586         104             76              59              119
6               384 x 288       143.913         1               6.0445          105             77              59              118
7               384 x 288       144.262         1               5.81679         107             78              58              118
8               384 x 288       147.893         1               5.81679         107             78              58              118
```

Figura A.2: Ejemplo de fichero de resultados.

Apéndice B

Pliego de condiciones

Para la correcta utilización del sistema desarrollado en este trabajo se debe disponer de un hardware y un software que cumpla unos requisitos mínimos.

B.1 Requisitos de Hardware

- Procesador de 32 ó 64 bits.
- 2GB de memoria RAM o superior.
- Al menos 300MB de memoria libres en el disco duro para funciones y datos.
- Al menos 3.8GB de memoria libres en el disco duro para la base de datos de *CAVIAR Project* [2].
- Cámara de vídeo GoPro HERO3 o similar.

B.2 Requisitos de Software

- Sistema operativo *Linux Ubuntu 14.04.1 LTS*
- Libreria *OpenCV 2.4.9*
- Al menos 300MB de memoria libres en el disco duro para funciones y datos.
- Compilador GNU *GCC*

Apéndice C

Presupuesto

C.1 Costes de equipamiento

- Equipamiento hardware utilizado:

Concepto	Cantidad	Coste Unitario	Subtotal(€)
PC Mac Mini Core 2 Duo I3	1	500€	500€
GoPro HERO 3	1	400€	400€
Coste total HW			900€

Tabla C.1: Coste equipamiento hardware utilizado

- Recursos software utilizados:

Concepto	Cantidad	Coste Unitario	Subtotal(€)
Ubuntu 14.04.1 LTS	1	0€	0€
Librería OpenCV 2.4.8	1	0€	0€
Software LaTeX	1	0€	0€
Coste total SW			0€

Tabla C.2: Coste equipamiento software utilizado

C.2 Costes de mano de obra

Concepto	Cantidad	Coste Unitario	Subtotal(€)
Desarrollo SW	250	65€/hora	16250€
Mecanografiado del documento	50	15€/hora	750€
Coste total mano de obra			17000€

Tabla C.3: Coste debido a mano de obra

C.3 Costes total del presupuesto

Concepto	Subtotal(€)
Equipamiento hardware	900€
Recursos software	0€
Mano de obra	17000€
Coste total presupuesto	17900€

Tabla C.4: Coste total del presupuesto

El importe total del presupuesto asciende a la cantidad de: DIECISIETE MIL NOVECIENTOS EUROS

En Alcalá de Henares a ___ de _____ de 20___

Marcos Baptista Ríos

Graduado en Ingeniería en Tecnologías de Telecomunicación.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá