ING. TEC. TELECOMUNICACIÓN ESPECIALIDAD
SONIDO E IMAGEN

PROYECTO FIN DE CARRERA

ESTUDIO DEL FILTRO DE PARTÍCULAS APLICADO AL SEGUIMIENTO DE OBJETOS EN SECUENCIAS DE IMÁGENES

ÁLVARO RODRÍGUEZ MOYA
OCTUBRE 2009

AGRADECIMIENTOS

La realización del Proyecto Fin de Carrera es una labor que requiere mucho trabajo, sacrificio y dedicación. Por unos motivos u otros, en mi caso no ha sido posible dedicar, en ciertos momentos, toda la atención que requiere un trabajo de estas características. Las pausas y los períodos de inactividad me han enseñado lo que cuesta retomar un proyecto, y han provocado que su realización se extienda más de lo que a cualquiera le gustaría.

Sin embargo, una vez terminado y con el duro trabajo de todos estos meses materializado en un libro, puedo decir que ha merecido la pena. Ha merecido la pena porque he aprendido mucho, no sólo sobre el tema acerca del que versa el proyecto, sino también sobre lo importante que es tener cerca a personas que te apoyen durante el trayecto, especialmente en los momentos difíciles.

Es por eso que quiero dedicar este trabajo a todas aquellas personas que, de una manera u otra, me han acompañado en este largo camino. Muchas han caminado conmigo de principio a fin, y otras sólo en algunos tramos. De cualquier manera, si he seguido adelante en todo momento ha sido gracias a todos ellos, y por eso parte de este proyecto les pertenece.

En primer lugar, quiero dar las gracias a mi familia. Todos y cada uno de ellos se han volcado con este trabajo, y siempre han estado pendientes de los avances dándome todo su apoyo. En especial, quiero agradecérselo a mis padres, ya que los valores que me han transmitido a lo largo de mi vida definen lo que soy.

También quiero dar las gracias a una persona muy especial. Paula, me has mostrado cuál es el camino a seguir y me has apoyado incondicionalmente, como nadie lo ha hecho.

A mis amigos, porque la verdadera amistad no se expresa con palabras, sino con hechos, y lo que vosotros habéis hecho por mí nunca os lo podré devolver. Pablo, Paul, Nacho, Carlos, Silvia, Lydia, Victor y Lety, gracias.

Al resto de Sonimágicos, también gracias, porque 4 años a vuestro lado no son suficientes y siempre querré más.

A toda la gente que está a mi alrededor, gracias, porque los momentos con vosotros hacen que cada día sea especial.

Por último, quiero agradecer a mis tutores su apoyo y seguimiento durante todo este tiempo. Lo que he aprendido os lo debo a vosotros.

PALABRAS CLAVE

Filtro de partículas, SIR, vídeo, seguimiento de objetivos, *tracking*, tratamiento de oclusiones, *multitracking*, ruido adaptativo, histograma múltiple, corrección pesos, remuestreo, Bhattacharyya, similitud de histogramas.

ABSTRACT

In recent years, particle filtering techniques to track objects in video sequences have captured the attention of many researchers in various communities, including those in signal processing, communication and image processing. Particle filtering is particularly useful in dealing with nonlinear state space models and non-Gaussian probability density functions. The underlying principle of the methodology is the approximation of relevant distributions with random measures composed of particles (samples from the space of the unknowns) and their associated weights.

This work makes a whole study of particle filtering applied to tracking, analyzing the influence in the behaviour of the algorithm of several parameters, and comparing different improvements.

Recientemente, las técnicas de filtrado de partículas aplicadas al seguimiento de objetos en secuencias de vídeo han captado la atención de muchos investigadores de diversas comunidades, como las de procesado de señal, comunicaciones y procesado de imagen. El filtro de partículas es particularmente útil para hacer frente a modelos de espacio de estados no lineales y f.d.p. no gaussianas. El principio subyacente de la metodología del filtro de partículas se basa en la aproximación de las distribuciones relevantes mediante medidas aleatorias compuestas por partículas (muestras del espacio no conocido) y sus pesos asociados.

El presente trabajo realiza un estudio exhaustivo del filtro de partículas aplicado al *tracking*, analizando la influencia de diversos parámetros en el comportamiento del algoritmo y comparando diferentes mejoras.

RESUMEN

This work makes a deep study in the field of particle filtering. Particularly, we analyze some free parameters of the basic algorithm, and look for the optimal values in terms of performance and computational cost. We improve this solution with several techniques like online parameter adaptation, covariance based object description or HSV histogramming. Finally we evaluate the accuracy and performance of these solutions through a variety of situations including target occlusions, camera flow and multiobject tracking.

Este trabajo hace un profundo estudio del campo del filtrado de partículas aplicado al seguimiento de objetos en secuencias de vídeo. En particular, analizamos algunos parámetros libres del algoritmo básico, buscando los valores óptimos en términos de rendimiento y coste computacional, y evaluando la influencia de cada uno en el comportamiento de la aplicación. Mejoramos esta solución básica con diversas técnicas como adaptación online de los parámetros, descripción de objeto basada en covarianza o utilización de histogramas HSV. Finalmente evaluamos la precisión y el rendimiento de estas soluciones en varias situaciones, incluyendo oclusiones del objetivo, movimiento de cámara y *tracking* de múltiples objetos.

ÍNDICE

1.-Introducción y objetivo 11
- 1.1. Marco o necesidad del proyecto 11
- 1.2. Objetivos 14
- 1.3. Fases y metodología 15
- 1.4. Estructura del documento 16

2.- Estado del arte del tracking 17
- 2.1. Representación de objetivos 17
  - 2.1.1 Representación de la forma del objeto 17
  - 2.1.2. Representación de apariencia del objeto 19
- 2.2. Selección de características 22
  - 2.2.1. Color 22
  - 2.2.2. Bordes 23
  - 2.2.3. Textura 23
- 2.3. Detección de objetos 24
  - 2.3.1. Detectores de puntos 25
  - 2.3.2. Sustracción de fondo 26
  - 2.3.3. Segmentación 26
  - 2.3.4. Aprendizaje supervisado 28
- 2.4. Métodos de tracking 32
  - 2.4.1. *Tracking* de punto 33
  - 2.4.2. *Tracking* de *kernel* 39
  - 2.4.3. *Tracking* de silueta 40

3.- El filtro de partículas 43
- 3.1. Introducción 43
- 3.2. Filtrado bayesiano recursivo 43
- 3.3. El filtro de Kalman 45
  - 3.3.1. Introducción al filtro de Kalman 45
  - 3.3.2. Algoritmo 47
  - 3.3.3. El filtro de Kalman Extendido (EKF) 49
- 3.4. El método Monte-Carlo 50
- 3.5. El filtro de partículas 51
  - 3.5.1. Introducción 51
  - 3.5.2. Diferencias con el filtro de Kalman 52
  - 3.5.3. Algoritmo 55
  - 3.5.4. Degeneración de los pesos 55
  - 3.5.5. Selección de la función de importancia 56
  - 3.5.6. Remuestreo 57

4.- Solución propuesta 59
- 4.1. Introducción a la solución 59
- 4.2. Solución básica 60
- 4.3. Ajuste de parámetros 63
  - 4.3.1. Numero de partículas 63
  - 4.3.2. Modo de selección de la partícula 64
  - 4.3.3. Desviación típica del ruido 64
  - 4.3.4. Número de *bins* del histograma 65

4.4. Extensiones 66
4.4.1. Extensión 1: corrección de los pesos 66
4.4.2. Extensión 2: ruido adaptativo 69
4.4.3. Extensión 3: histograma HSV 71
4.4.4. Extensión 4: histograma múltiple 72
4.4.5. Extensión 5: matriz de covarianzas 73
4.5. Conclusiones 76

5.- Pruebas y resultados 77
5.1. Entorno de trabajo 77
5.2. Medidas de calidad utilizadas en la evaluación 79
5.2.1. Calidad de la estimación 79
5.2.2. Desviación típica media de la secuencia 81
5.2.3. Tiempo medio de cálculo por *frame* de la secuencia 82
5.3. Pruebas realizadas 82
5.3.1. Establecimiento de los parámetros óptimos del algoritmo básico 82
5.3.2. Algoritmo básico 86
5.3.3. Extensión 1: corrección de los pesos 89
5.3.4. Extensión 2: ruido adaptativo 91
5.3.5. Extensión 3: histograma HSV 93
5.3.6. Extensión 4: histograma múltiple 95
5.3.7. Extensión 5: matriz de covarianzas 96
5.4. Algoritmo final 98
5.5. Multitracking 100

6.- Conclusiones y trabajo futuro 103

7.- Bibliografía 107

ÍNDICE DE FIGURAS

Figura 1. Esquema del procedimiento básico para el tratamiento digital de imágenes 12

Figura 2. Ejemplos de *tracking* y trayectorias obtenidas 13

Figura 3. Posibles representaciones de un objetivo 19

Figura 4. Histograma de color RGB de una imagen 20

Figura 5. Modelo de apariencia activa para detección de caras humanas 21

Figura 6. Modelo de apariencia multivista para detección de caras humanas 21

Figura 7. Distintas formas de seleccionar características de una imagen 23

Figura 8. Puntos de interés obtenidos al aplicar las técnicas (a) Harris, (b) KLT, y (c) SIFT 26

Figura 9. Modelo de mezcla de gaussianas para sustracción de fondo 26

Figura 10. Ejemplos de métodos de segmentación 28

Figura 11. Ejemplo de aplicación de los métodos de aprendizaje supervisado al *tracking*: detección de caras humanas 29

Figura 12. Funcionamiento de SVM 30

Figura 13. Funcionamiento de *Adaboost* 31

Figura 14. Esquema de una red neuronal 32

Figura 15. Diferentes aproximaciones al seguimiento de objetos 33

Figura 16. Esquema general de las diferentes técnicas de *tracking* revisadas en la Sección 2.4 33

Figura 17. Correspondencia entre puntos 34

Figura 18. Resultados de dos algoritmos de correspondencia de puntos 35

Figura 19. Diferentes restricciones de movimiento 35

Figura 20. Iteraciones de *tracking* utilizando *mean-shift* 40

Figura 21. Resultados del *tracking* de contorno 42

Figura 22. Ciclo de funcionamiento del filtro de Kalman 47

Figura 23. Diagrama completo de operación del filtro de Kalman, combinando el diagrama de alto nivel de la Figura 22 con las ecuaciones del algoritmo 1 49

Figura 24. Ilustración del proceso de Remuestreo 58

Figura 25. Ejemplos de *tracking* con diferente número de partículas 63

Figura 26. Diferencias en el cambio en el vector de estados de un *frame* al siguiente 65

Figura 27. Influencia de la resolución del histograma en el *tracking* 66

Figura 28. Importancia de los *outliers* 67

Figura 29. Función lineal aplicada al remuestreo 67

Figura 30. Función exponencial aplicada al remuestreo 68

Figura 31. Función sigmoide aplicada al remuestreo 69

Figura 32. Función sigmoide para varios valores de α y β 71

Figura 33. Representación del espacio de color HSV 72

Figura 34. Ejemplos de *tracking* empleando la técnica de histograma múltiple 73

Figura 35. Secuencias utilizadas en las pruebas 78

Figura 36. Medida de calidad *Q1* propuesta por Phillips y Chhabra [50] 80

Figura 37. Medida de calidad *Q2*, ejemplos de localización estimada precisa 81

Figura 38. 5 iteraciones del algoritmo para distintos valores de N_s: 100, 500, 2000 81

Figura 39. Comparación partícula media y partícula máxima 83

Figura 40. Comparativa de N_s para las distintas secuencias 84

Figura 41. Consecuencias de introducir ruido elevado 85

Figura 42. Resultados del algoritmo para valores de ruido bajos 86

Figura 43. Gráfica de estimación de los parámetros del objetivo (ground-truth en azul y salida del algoritmo en rojo) para la Secuencia 6 87

Figura 44. Diferencia entre utilizar objetivos completos o zonas características 88

Figura 45. Funcionamiento del algoritmo básico, secuencia 5 89

Figura 46. Comparación entre la versión básica y la extensión 1 90

Figura 47. Gráfica comparativa de Q entre la versión básica y las distintas implementaciones de la extensión 1 90

Figura 48. Influencia del ruido en las componentes (a) estática y (b)

dinámica 91

Figura 49. Resultados de la extensión 2 92

Figura 50. Inestabilidad de la extensión 3: histograma HSV 93

Figura 51. Resolución de la oclusión en la secuencia 5 94

Figura 52. Inestabilidad de la extensión 4 95

Figura 53. Estabilidad de la extensión 5 97

Figura 54. Gráfica comparativa de calidad Q de todas las implementaciones 99

Figura 55. Mejora resultante de la combinación de las extensiones 1 y 2 100

Figura 56. Asociación de los objetivos tras una oclusión en *multitracking* 100

Figura 57. Algoritmo final aplicado al *multitracking* 101

Figura 58. Limitación impuesta por representación rectangular del objeto 103

Figura 59. Importancia de la información espacial en *tracking* 104

Figura 60. Error derivado de cambio de apariencia (giro del objetivo) 105

ÍNDICE DE TABLAS

Tabla 1. Resumen de las distintas técnicas de detección de objetos agrupadas por categorías 25

Tabla 2. Secuencias de test empleadas en los experimentos 78

Tabla 3. Tablas comparativas de $\overline{Q1}$ y $\overline{Q2}$ para distintos valores de N_s 84

Tabla 4. Comparativa de las medidas obtenidas para distintos valores de K 85

Tabla 5. Resultados de la versión básica para las 4 secuencias de prueba 89

Tabla 6. Comparativa de las diferentes funciones utilizadas en la extensión 1 91

Tabla 7. Comparativa de la extensión 2 y la versión básica 93

Tabla 8. Comparativa de la extensión 3 y la versión básica 95

Tabla 9. Comparativa de la extensión 4 y la versión básica 96

Tabla 10. Comparativa de la extensión 5 y la versión básica 96

Tabla 11. Comparación general de los algoritmos desarrollados 98

Tabla 12. Resultados del algoritmo final aplicado a *multitracking* 101

1. INTRODUCCIÓN Y OBJETIVOS

1.1 MARCO DEL PROYECTO

La **Visión Computacional** consiste en el estudio de los procesos que intervienen en la visión, para entenderlos y diseñar máquinas o aplicaciones con capacidades similares.

La Visión Computacional, en un intento de reproducir el comportamiento del ser humano, define tradicionalmente cinco fases principales (ver Figura 1):

- La primera fase, que es puramente sensorial, consiste en la Captura o Adquisición de las imágenes digitales mediante algún tipo de sensor. Se muestrea, discretiza y almacena digitalmente la imagen.

- La segunda etapa consiste en el tratamiento digital de las imágenes, con objeto de facilitar las etapas posteriores. En esta etapa de Preprocesamiento es donde, mediante filtros o transformaciones geométricas, se eliminan partes indeseables de la imagen o se realzan partes interesantes de la misma.

- La Segmentación consiste en dividir la escena en regiones de atributos similares. De esta manera podemos aislar los distintos elementos que la componen y extraer los de interés.

- El siguiente paso es Extraer las características relevantes del objeto, de cara a clasificarlo adecuadamente.

- Por último se llega a la etapa de Reconocimiento o Clasificación. En ella se pretende distinguir los objetos segmentados, gracias al análisis de las características extraídas previamente para diferenciarlos.

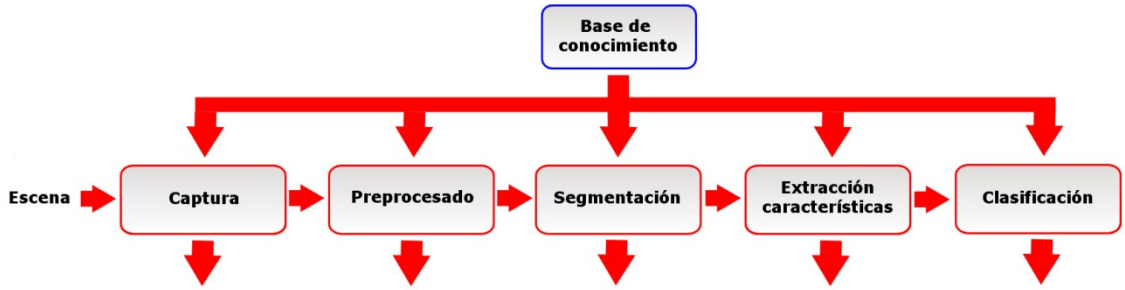


Figura 1. Esquema del procedimiento básico para el tratamiento digital de imágenes

Algunas de las causas del avance de las técnicas de procesamiento de imágenes son:

- Respecto a la tecnología: más calidad, más potencia (capacidad de procesamiento, memoria), menos precio (videocámaras, sensores, equipos informáticos).

- Avances teóricos en los principios y algoritmos para el procesamiento y análisis de imágenes.

- Necesidad creciente de análisis de vídeo automatizado: medicina, seguridad y supervisión, clasificación de contenidos, video online

Un problema clásico en visión computacional es el seguimiento de objetos en secuencias de imágenes, denominado comúnmente ***tracking***. Su resolución consta de tres etapas clave: detección de los objetos de interés, seguimiento de dichos objetos de un *frame* al siguiente y, por último, análisis del movimiento que permite reconocer algún patrón de comportamiento (ver Figura 2).

Durante las dos últimas décadas, el seguimiento de objetos en secuencias de vídeo ha recibido una especial atención por parte de la comunidad investigadora. Algunos de sus usos más importantes son:

- Reconocimiento basado en movimiento: identificación humana, detección automática de objetos…

- Supervisión o vigilancia automática: monitorear una escena para detectar actividades sospechosas o eventos no deseados.

- Indexado de video: anotación automática de parámetros y recuperación de vídeos en bases de datos multimedia [1].

- Interacción ordenador-humano: reconocimiento de gestos, seguimiento de los ojos, etc.

- Monitorización del tráfico: obtención de estadísticas del tráfico en tiempo real para dirigirlo adecuadamente.

- Navegación de vehículos: capacidades de planificación de rutas basada en vídeo y evitación de obstáculos

Todas ellas requieren un algoritmo de *tracking* robusto para funcionar de manera óptima. Para poder ser aplicado en sistemas de visión, el algoritmo debe ser capaz de controlar oclusiones parciales y totales, cambios de escala del objeto, cambios de iluminación de la escena, movimientos erráticos, medidas ruidosas… Además, para algunas aplicaciones como supervisión o interacción, los *trackers* deben proporcionar seguimientos fiables a lo largo de enormes secuencias de vídeo, en tiempo real y a altas frecuencias de refresco. Estas restricciones requieren de una aplicación estable y computacionalmente eficiente.

(a) (b)

Figura 2. Ejemplos de *tracking* y trayectorias obtenidas. (a) *Tracking* mono-objetivo. (b) *Tracking* multi-objetivo

Una técnica de *tracking* que ha recibido una atención considerable es la familia de los **filtros de partículas**. El filtro de partículas es una técnica basada en estimación Bayesiana recursiva de un estado del sistema desconocido. El interés que han despertado estas técnicas radica en su habilidad para trabajar con múltiples hipótesis, y en la posibilidad de desarrollar implementaciones simples y eficientes.

Un asunto crucial en *tracking* visual es la adaptación del algoritmo general a la aplicación de seguimiento particular. El principal problema es que, habitualmente, existen multitud de parámetros que deben ser fijados, y en muchas ocasiones no está claro cómo deben ser establecidos para cada problema en particular.

1.2. OBJETIVOS

El objetivo fundamental de este proyecto consiste en implementar el filtro de partículas aplicado al seguimiento de objetivos en secuencias de imágenes 2D, evaluando los resultados que se obtienen en la aplicación bajo diferentes condiciones experimentales.

El proyecto se va a dividir en dos grandes bloques: *tracking* mono-objetivo y *tracking* multi-objetivo. Para ambos se plantearán algunas aplicaciones de interés en las que comprobar la validez de los desarrollos.

Comenzaremos implementando la solución del filtro de partículas propuesta por [2]. Se trata de una versión básica del algoritmo SIR (*Sequential Importance Resampling*) que ha demostrado dar buenos resultados en *tracking* de objetivos en secuencias de imágenes.

Una vez creado el núcleo del filtro de partículas básico y las funciones complementarias, ajustaremos los valores de los parámetros libres, incidiendo en la influencia de cada uno respecto al rendimiento.

Definidos estos parámetros, realizaremos diversas modificaciones sobre el sistema básico con el fin de mejorar los resultados del *tracking*.

Probaremos el funcionamiento de la versión básica y las extensiones propuestas para uno y varios objetivos (*multitracking*). Finalmente analizaremos los resultados

obtenidos y evaluaremos las distintas soluciones, además de proponer otras posibles mejoras para optimizarlas. En este punto es interesante comparar las distintas alternativas entre sí, para analizar las fortalezas y debilidades de cada una. El escenario de prueba deberá ser lo más generalista posible, contemplando diferentes objetivos, situaciones y condiciones.

Para desarrollar la solución propuesta aprenderemos a utilizar el entorno de programación Matlab, incidiendo en las técnicas de cálculo algebraico y tratamiento de imágenes. También será necesario obtener los contenidos audiovisuales (vídeos con objetivos en movimiento y *ground-truth* asociado) sobre los que se evaluará el algoritmo.

1.3. FASES Y METODOLOGÍA

En primer lugar, se debe realizar un profundo análisis del estado del arte de métodos de *tracking*, en especial de las diferentes posibilidades existentes a la hora de implementar el filtro de partículas.

Una vez analizado el estado actual de las técnicas de seguimiento, se procede a elegir el sistema más adecuado a las necesidades del proyecto.

El siguiente paso es diseñar el algoritmo seleccionado. Para ello se utilizará el lenguaje de programación Matlab.

Una vez en funcionamiento la aplicación básica, y tras realizar unas mínimas pruebas de comprobación, se procede a desarrollar las extensiones.

Con la aplicación completamente desarrollada se realizan diversas pruebas experimentales para evaluar la influencia de cada parámetro en el rendimiento del algoritmo y comparar las diversas extensiones implementadas.

Por último, se establecen las conclusiones y líneas de trabajo futuras derivadas de los resultados obtenidos.

1.4. ESTRUCTURA DEL DOCUMENTO

La memoria de este proyecto está organizada en 7 capítulos. A continuación se presenta una breve descripción de cada uno:

Capítulo 1: Marco teórico en el que se desarrolla el proyecto, motivación y objetivos a alcanzar.

Capítulo 2: Breve análisis de las técnicas y tecnologías comúnmente empleadas en el seguimiento de objetos.

Capítulo 3: Descripción del filtro de partículas y las técnicas asociadas: método de Monte Carlo y filtro de Kalman. Desarrollo del algoritmo.

Capítulo 4: Desarrollo de la solución propuesta. Descripción de las funcionalidades implementadas tanto en la versión básica como en sus extensiones.

Capítulo 5: Aplicación del sistema implementado al seguimiento de objetos bajo diferentes condiciones: oclusiones, movimiento de cámara y múltiples objetos en escena. Análisis de la importancia de los parámetros libres. Comparación de las diferentes extensiones. Descripción y análisis de las medidas y resultados obtenidos de las pruebas.

Capítulo 6: Extracción de conclusiones en base a los resultados obtenidos. Limitaciones y debilidades de la aplicación. Análisis de la continuidad del proyecto a través de posibles mejoras y desarrollos futuros.

2. ESTADO DEL ARTE

En el contexto del *seguimiento de objetivos*, se puede definir un objetivo u objeto como cualquier región de la imagen que sea de interés para un análisis posterior. Por ejemplo, un barco en el mar, vehículos en la carretera, aviones en el aire o gente caminando por la calle son conjuntos de objetivos que podrían ser importantes para hacer seguimiento en un dominio específico.

Habitualmente, los sistemas de *tracking* se componen de cuatro etapas: representación del objetivo, extracción de características, detección del objetivo en la escena y seguimiento del objetivo (trayectoria seguida). Estas etapas están estrechamente vinculadas, ya que la técnica escogida para la representación del objetivo condiciona en gran parte su detección y posterior seguimiento. En los siguientes subapartados se estudiarán las técnicas comúnmente utilizadas para resolver cada una de las fases mencionadas.

2.1. REPRESENTACIÓN DE OBJETIVOS

Los objetivos pueden ser representados por su forma y apariencia. En esta sección se describirán inicialmente las representaciones de la forma de los objetos comúnmente utilizadas en el *tracking* para, después, estudiar representaciones más complejas que unen forma y apariencia.

2.1.1 Representación de la forma del objeto

a) Puntos

El objetivo es representado por un punto, esto es, el centroide o centro de masas (Figura 3(a)) [3], o un conjunto de puntos (Figura 3(b)) [4]. Esta representación es adecuada para seguimiento de objetos que ocupan regiones pequeñas en la imagen.

b) Formas geométricas primitivas

La forma del objetivo es representada por un rectángulo, elipse (Figura 3(c), (d)), etc. [5]. El movimiento del blanco para estas representaciones es modelado usualmente por transformaciones de translación, afinidad o proyección. A pesar de que estas formas primitivas son más apropiadas para representación de objetos rígidos simples, también son usadas para *tracking* de objetos no rígidos.

c) Silueta y contorno

La representación del contorno define los límites o la forma de un objeto (Figura 3(g), (h)). La región dentro del contorno es llamado silueta del objeto (Figura 3(i)). Esta representación es adecuada para seguimiento de formas no rígidas más complejas [6].

d) Modelos con formas articuladas

Los objetos articulados están compuestos de diversas partes unidas entre sí a través de uniones. Por ejemplo, el cuerpo humano es un objetivo articulado con cabeza, torso, brazos, piernas, manos y pies unidos mediante articulaciones. La relación entre las partes está gobernada por un modelo de movimiento cinemático, por ejemplo, ángulo de unión, etc. Para representar un objeto articulado, se pueden modelar las partes que lo constituyen usando cilindros o elipses (Figura 3(e)).

e) Modelos esqueléticos

El esqueleto de cualquier objeto puede ser extraído aplicando la transformación MAT (*Medial Axis Transformation*) a la silueta del objeto [7]. Este modelo se usa comúnmente en el reconocimiento de objetos [8]. La representación basada en esqueleto puede utilizarse para modelar tanto objetos rígidos como articulados (Figura 3(f)).

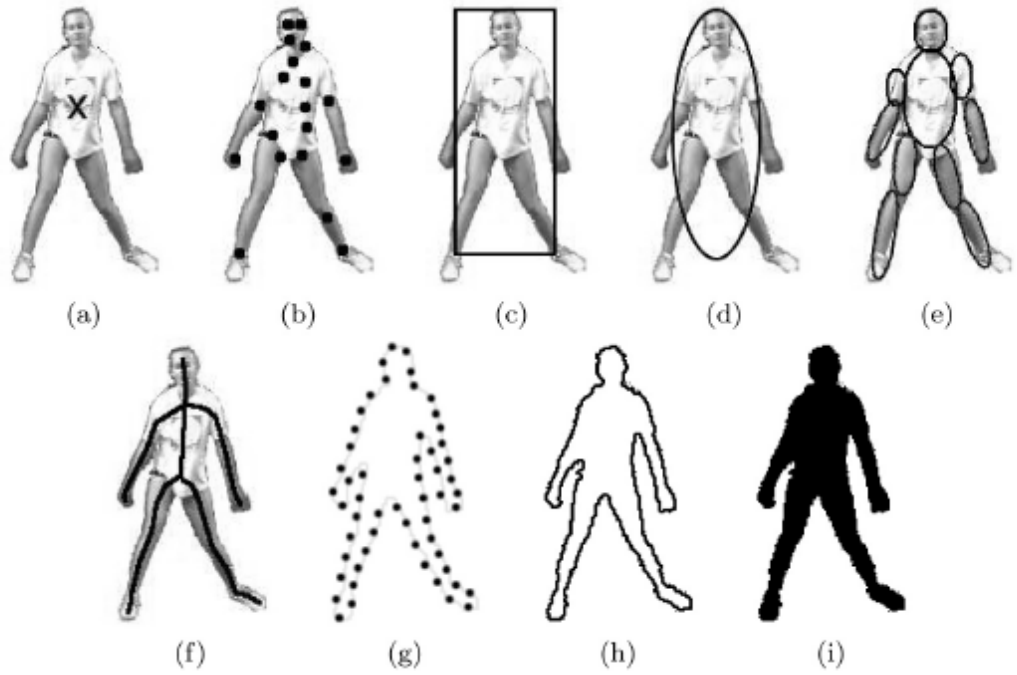


Figura 3. Posibles Representaciones de un objetivo. (a) Centroide, (b) múltiples puntos, (c) parche rectangular, (d) parche elíptico, (e) parches en múltiples partes, (f) esqueleto, (g) puntos de control en el contorno, (h) contorno completo del objeto, (i) silueta del objeto. (Figura extraída de [9]).

2.1.2. Representación de apariencia del objeto

Las representaciones de la forma del objeto vistas anteriormente se suelen combinar con información sobre otras características del objeto, como la textura y el color. Se denomina representación de apariencia a la forma de representar tales características. Las más comunes en el contexto del *tracking* son:

a) Densidad de probabilidad de la apariencia del objeto

La estimación de la densidad de probabilidad de la apariencia del objeto puede ser paramétrica, como una Gaussiana o una mezcla de Gaussianas, o no paramétrica, como las ventanas de Parzen y los histogramas (Figura 4). Las densidades de probabilidad de las características de apariencia del objeto (color, textura) pueden ser calculadas a partir de las regiones de la imagen especificadas por los modelos de forma (regiones interiores de elipses o contornos) [5].

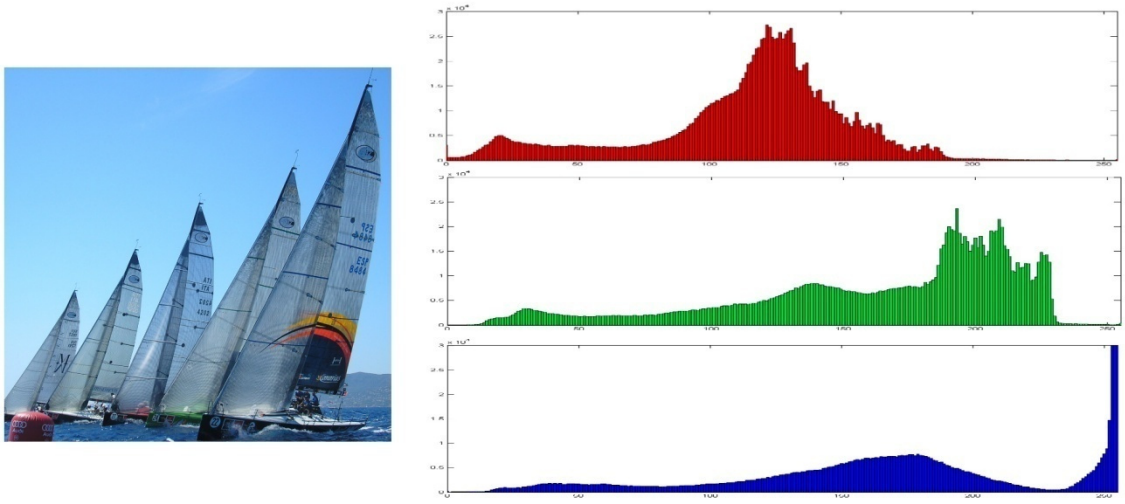


Figura 4. Histograma de color RGB de una imagen. Se puede apreciar que gran parte de los píxeles son azul intenso (valores cercanos a 255) y no tienen componente roja ni verde: corresponden al cielo en la imagen.

b) Plantillas

Las plantillas se diseñan usando formas geométricas simples o siluetas [10]. Una ventaja de las plantillas es que proporcionan información tanto espacial como de apariencia. Sin embargo, las plantillas sólo codifican la apariencia de un objeto generada desde una sola vista. Por ello, sólo son adecuadas para el *tracking* de objetos cuya pose no varía considerablemente durante el transcurso del *tracking*.

c) Modelos de apariencia activa

Son generados modelando simultáneamente la forma del objeto y la apariencia [11]. En general, la forma del objeto es definida por un conjunto de marcas (Figura 5). Al igual que en la representación basada en contorno, las marcas pueden residir en el borde del objeto o, alternativamente, dentro de la región del mismo. Para cada marca, un vector de apariencia es almacenado en forma de magnitud de color, textura o gradiente. Los modelos de apariencia activa requieren una fase de entrenamiento donde la forma y su apariencia asociada son aprendidas de un conjunto de muestras usando, por ejemplo, PCA (*Principal Component Anaylisis*).

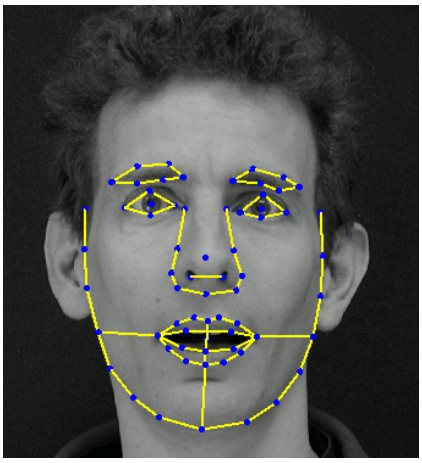


Figura 5. Modelo de apariencia activa para detección de caras humanas. Las 68 marcas azules definen la forma del objeto, y cada una almacena un vector de apariencia. (Figura extraída de [11]).

d) Modelos de apariencia multi-vista

Estos modelos codifican diferentes vistas de un objeto (Figura 6). Una aproximación para representar las diferentes vistas del objeto es generar un subespacio a partir de las vistas dadas. Aproximación de subespacios como PCA o ICA (*Independent Component Anaylisis*) son comúnmente empleadas para la representación conjunta de forma y apariencia.

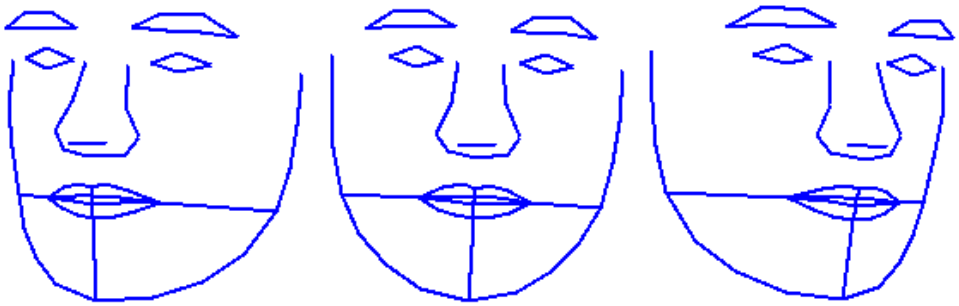


Figura 6. Modelo de apariencia multivista para detección de caras humanas. Se emplean en este caso 3 vistas para la detección de la cara con un ángulo de rotación aproximado de ± 30º.

En general, existe una estrecha relación entre la representación del objeto y los algoritmos de *tracking*. La representación de los objetos es escogida normalmente de acuerdo al dominio de aplicación; por ejemplo, para objetos pequeños la representación por puntos, como ya se ha dicho, es muy apropiada. Para objetos

con formas complejas, como humanos, una representación por contorno o silueta resulta más adecuada.

2.2. SELECCIÓN DE CARACTERÍSTICAS

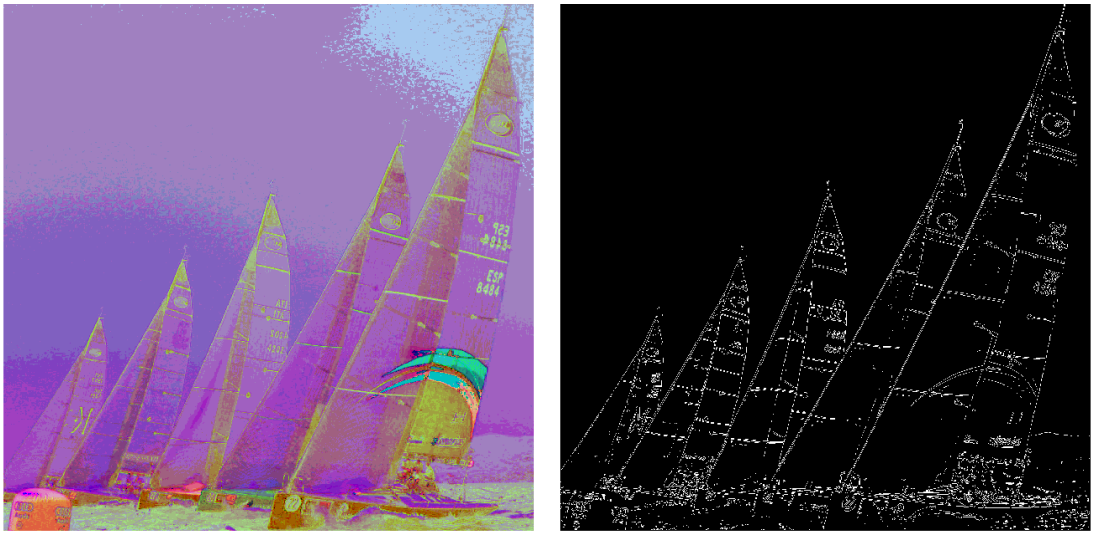
Seleccionar las características correctas juega un papel crítico en *tracking*. En general, la propiedad más deseable de una característica visual es su singularidad, de forma que los objetos puedan ser fácilmente discriminados en el espacio de características. La selección de características está estrechamente relacionada con la representación de los objetos. Por ejemplo, el color es usado como una característica por las representaciones basadas en histograma mientras que, para las basadas en contornos, se suelen utilizar los bordes del objeto. En general, la mayoría de algoritmos de *tracking* usan una combinación de estas características. Los detalles de las características visuales más comunes se comentan a continuación.

2.2.1. Color

El color aparente de un objeto es influenciado principalmente por dos factores físicos, 1) la distribución espectral de potencia de la iluminación y 2) las propiedades de reflectancia de la superficie del objeto. En procesamiento de imágenes, el espacio de color RGB (rojo, verde, azul) es comúnmente usado para representar el color. Sin embargo, el espacio RGB no es un espacio de color perceptualmente uniforme, es decir que las diferencias entre colores en el espacio RGB no se corresponden con las diferencias entre colores percibidas por el ser humano [12]. Adicionalmente, las dimensiones RGB están altamente correladas. Por el contrario, L*u*v y L*a*b son espacios de color perceptualmente uniformes, mientras que HSV (*Hue, Saturation, Value*) es aproximadamente uniforme (Figura 7(a)). Sin embargo, estos espacios de color son sensibles al ruido. En resumen, aún no se ha dicho la última palabra sobre qué espacio de color es más eficiente, por lo que se usan una gran variedad de ellos en *tracking*.

2.2.2. Bordes

Los límites de un objeto usualmente generan cambios bruscos en la intensidad de la imagen. La detección de bordes es usada para identificar estos cambios (Figura 7(b)). Una propiedad importante de los bordes es que son menos sensibles a los cambios de iluminación en comparación con las características de color. Los algoritmos que hacen *tracking* tomando como referencia el límite de los objetos normalmente usan bordes como característica representativa. Gracias a su simplicidad y precisión, la aproximación a la detección de bordes más popular es el detector de bordes de Canny [13]. Una evaluación de los algoritmos de detección de bordes puede ser consultada en [14].



(a) (b)

Figura 7. Distintas formas de seleccionar características de una imagen. (a) Imagen en el espacio de color HSV. (b) Bordes extraídos de la imagen mediante el detector de bordes de Sobel.

2.2.3. Textura

La textura es la medida de la variación de intensidad de una superficie, y cuantifica propiedades tales como suavidad o regularidad. Comparada con el color, la textura requiere un paso de procesamiento para generar los descriptores. Hay varios descriptores de textura: Matrices de Concurrencia de Niveles de Gris (*Gray Level Co-ocurrence Matrix* o GLCM's [15], un histograma 2D que muestra las concurrencias de intensidades en una dirección y distancia específicas), las medidas de textura de Law (25 filtros 2D generados a partir de 5 filtros 1D correspondientes a nivel, borde, punto, ola y rizo [16]), *wavelets* (bancos de filtros ortogonales [17])

y pirámides dirigibles [18]. Similares a las características de borde, las de textura son menos sensibles a los cambios de iluminación comparadas con las de color.

Muchas de las características son elegidas manualmente por el usuario dependiendo del dominio de la aplicación. Sin embargo, el problema de selección automática de características ha recibido una atención significativa dentro de la comunidad de reconocimiento de patrones. Los métodos de selección automática de características pueden ser divididos dentro de métodos de filtro y métodos de envoltorio. Los métodos de filtro intentan seleccionar características basándose en un criterio general, por ejemplo, que las características sean incorreladas. Los métodos de envoltorio seleccionan una característica basándose en la utilidad de esa característica en el dominio de aplicación del problema específico, por ejemplo, el rendimiento de la clasificación usando un subconjunto de características. PCA es un ejemplo de método de filtro para reducción de características. PCA implica transformación de un número de (posibles) variables correladas en un conjunto (reducido) de variables incorreladas llamadas componentes principales. Un método de envoltorio para seleccionar las características discriminantes para *tracking* de una clase particular de objetos es el algoritmo de *Adaboost*. Se trata de un método que permite encontrar un clasificador potente basado en una combinación de clasificadores débiles o moderadamente imprecisos. Dado un conjunto grande de características, un clasificador puede ser entrenado para cada característica. *Adaboost* descubrirá una combinación ponderada de clasificadores (representando características) que maximiza el rendimiento de la clasificación. Cuanto más grande sea el peso asignado a una característica, más discriminante es. Así, a partir de un número de características suficientemente grande, se podrían seleccionar las n características de mayor peso para el *tracking*.

2.3. DETECCIÓN DE OBJETOS

Todo algoritmo de *tracking* precisa de un mecanismo de detección del objeto que permita posteriormente realizar un seguimiento del mismo (ver tabla 1). Una aproximación común a la detección del objeto consiste en utilizar información de un solo plano o *frame* del vídeo. Sin embargo, algunos métodos de detección de objetos hacen uso de la información temporal calculada a partir de una secuencia de *frames* para reducir el número de falsas detecciones. Esta información temporal suele darse en la forma de diferencia entre *frames*, destacando las regiones cambiantes en planos consecutivos. Dada la región del objeto en la imagen, es

tarea del *tracker* realizar la correspondencia del objeto de un *frame* al siguiente para generar la trayectoria.

Categorías	Trabajo representativo
Detectores de puntos	Detector de Moravec Detector de Harris SIFT KLT
Segmentación	*Mean-shift* Grafo de cortes Contornos activos
Modelado del fondo	Mezcla de gaussianas Eigenbackground Wall flower Fondo de textura dinámica
Clasificación supervisada	Máquinas de vectores soporte (SVM) Redes neuronales *Adaptive Boosting*

Tabla 1. Resumen de las distintas técnicas de detección de objetos agrupadas por categorías.

2.3.1. Detectores de puntos

Los detectores de puntos se usan para encontrar puntos de interés en imágenes, definiendo estos puntos de interés como aquellos que tienen una textura expresiva o singular en sus respectivas localizaciones. Estos puntos han sido ampliamente usados en los contextos de movimiento, visión estéreo y problemas de *tracking*. Una propiedad deseable de un punto de interés es su invarianza frente a cambios de iluminación o del punto de vista de la cámara. En la literatura se incluyen ejemplos comúnmente utilizados como el operador de interés de Moravec, el detector de puntos de interés de Harris [19], el detector KLT [20] y el detector SIFT [21] (Figura 8). Para una evaluación comparativa de estos detectores, el lector interesado puede consultar [22].

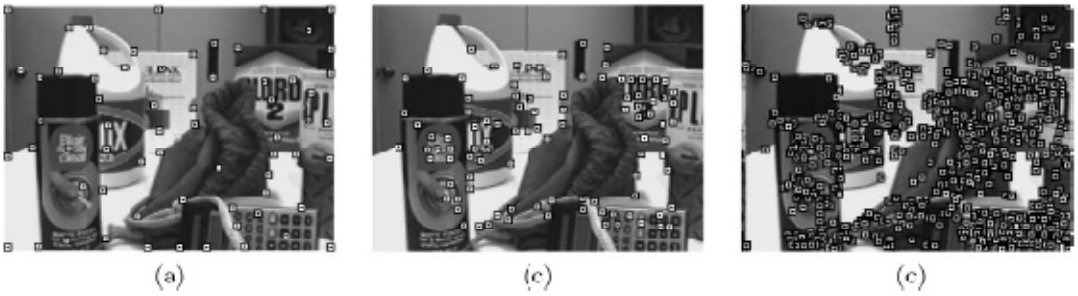


Figura 8. Puntos de interés obtenidos al aplicar las técnicas (a) Harris, (b) KLT, y (c) SIFT (Figura extraída de [9]).

2.3.2. Sustracción de fondo

La detección de objetos puede llevarse a cabo construyendo una representación de la escena llamada modelo de fondo y encontrando cualquier desviación del modelo en cada *frame* entrante al sistema [23]. Cualquier cambio significativo en una región de la imagen frente al modelo de fondo conlleva la aparición de un objeto en movimiento. Los píxeles que constituyen la región cambiada son marcados para el procesamiento posterior. Usualmente, un algoritmo de componentes conectados se aplica para obtener la región conectada correspondiente al objeto. Este proceso es denominado sustracción de fondo y puede hacer uso de modelos probabilísticos que aumenten la robustez del sistema frente a cambios de iluminación debidos a factores externos o la aparición de elementos móviles del fondo (por ejemplo, árboles que se mueven a causa del viento). En la Figura 9 se puede ver un ejemplo de la utilización de un modelo de mezcla de gaussianas para la detección de objetos en una secuencia de vídeo.

Figura 9: Modelo de mezcla de gaussianas para sustracción de fondo. (a) Imagen de una secuencia en la que una persona camina por la escena. (b) La media de las gaussianas de mayor peso en la posición de cada píxel. Estas medias representan los colores más persistentes temporalmente en cada píxel y deberían representar el fondo estacionario. (c) Las medias de la gaussiana con el segundo mayor peso. Estas medias representan colores observados menos frecuentemente. (d) Resultado de la sustracción del

fondo. El primer plano consiste en los píxeles del cuadro actual que coinciden con una gaussiana con pesos bajos, (Figura extraída de [9]).

2.3.3. Segmentación

El objetivo de los algoritmos de segmentación de imágenes es particionar la imagen en regiones perceptualmente similares. Todos los algoritmos se enfocan en dos problemas, el criterio para una buena partición y el método para lograr una partición eficiente. Algunas técnicas de segmentación relevantes se presentan a continuación.

a) *Mean-Shift Clustering*

Para el problema de segmentación, Comaniciu y Meer [10] propusieron una aproximación por *mean-shift* para encontrar *clusters* en la unión del espacio de color y de localización, [l,u,v,x,y], donde [l,u,v] representan las componentes del espacio de color Luv y [x,y] es la localización espacial de cada píxel de la imagen. Dada una imagen, el algoritmo se inicializa con un gran número de centroides elegidos aleatoriamente a partir de los datos. Después, cada centroide es movido a la media de los datos que hayan sido asignados a dicho centroide (asignación que se realiza mediante una medida de distancia). El vector definido por los centroides del viejo y el nuevo *cluster* se llama vector *mean-shift*. Este vector se calcula iterativamente hasta que los centroides no cambian de posición (Figura 10(a),(b)).
Este tipo de segmentación requiere un ajuste fino de algunos parámetros para obtener mejores resultados, como por ejemplo las bandas de color o el umbral para el mínimo tamaño de región que resulta de la segmentación.

b) Contornos activos

La segmentación del objeto se resuelve evolucionando un contorno cerrado hasta los bordes del objeto, de manera que el contorno encierra la región del objeto. La evolución del contorno está gobernada por una función de energía que define cuánto de buena es la adaptación del contorno a la región hipotética del objeto.

La tarea más importante en los métodos basados en contorno, además de la selección de la función de energía [24] y la forma adecuada del contorno, es la inicialización del mismo. En aproximaciones basadas en el gradiente de la imagen, el contorno se sitúa típicamente fuera de la región del objeto y se comprime hasta

que el borde del objeto es encontrado. En métodos basados en regiones, esta restricción se relaja, pudiendo inicializarse tanto dentro como fuera del objeto, de manera que el contorno puede expandirse o contraerse, respectivamente, para ajustarse a los límites del objeto.

c) *Graph-cuts*

La segmentación de imágenes también puede ser formulada como un problema de partición de un grafo, donde los vértices (*pixels*) de un grafo (imagen) son particionados en N subgrafos disjuntos (regiones), a través de la poda de bordes ponderados del grafo (Figura 10(c)). El peso total de los bordes podados entre dos subgrafos se denomina corte. El peso es calculado típicamente a partir de la similitud de color, brillo o textura entre los nodos.

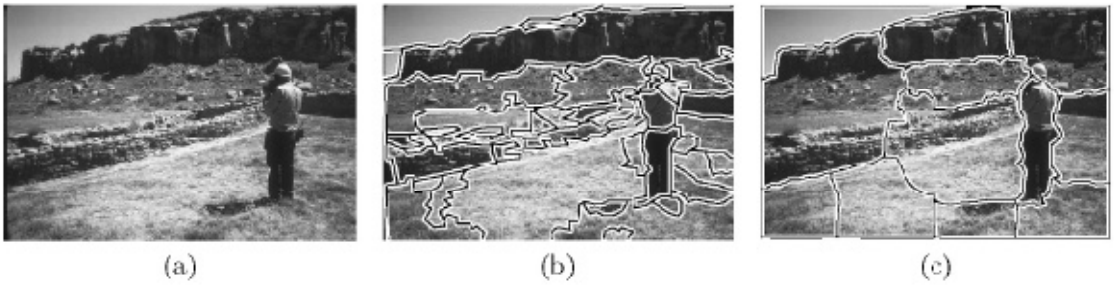


Figura 10. Ejemplos de métodos de segmentación. (a) imagen original, (b) segmentación obtenida mediante el algoritmo *mean-shift* y (c) segmentación obtenida mediante cortes normalizados [25]. (Figura extraída de [9]).

2.3.4. Aprendizaje supervisado

La detección de objetos puede ser resuelta mediante el aprendizaje automático de diferentes vistas del objeto extraídas de un conjunto de ejemplos. Es lo que se llama mecanismo de aprendizaje supervisado: dado un conjunto de ejemplos de aprendizaje, estas técnicas generan una función que mapea o relaciona las entradas con las salidas deseadas.

En el contexto de la detección de objetos, los ejemplos de aprendizaje se componen de diferentes clases de objetos y sus características asociadas. La correcta selección de las características o descriptores de los objetos juega un papel importante en el resultado de la clasificación. Por tanto, es fundamental elegir un

conjunto de descriptores que permitan discriminar entre una clase y otra, facilitando la labor del algoritmo.

Un ejemplo clásico de aplicación de estos mecanismos en detección de objetos es la detección de caras humanas (ver Figura 11). El conjunto de aprendizaje se compone, por un lado, de caras humanas de todo tipo, y por otro, de objetos y texturas que no representan caras. De esta forma el algoritmo puede establecer un criterio adecuado acerca de lo que es objeto y lo que no.

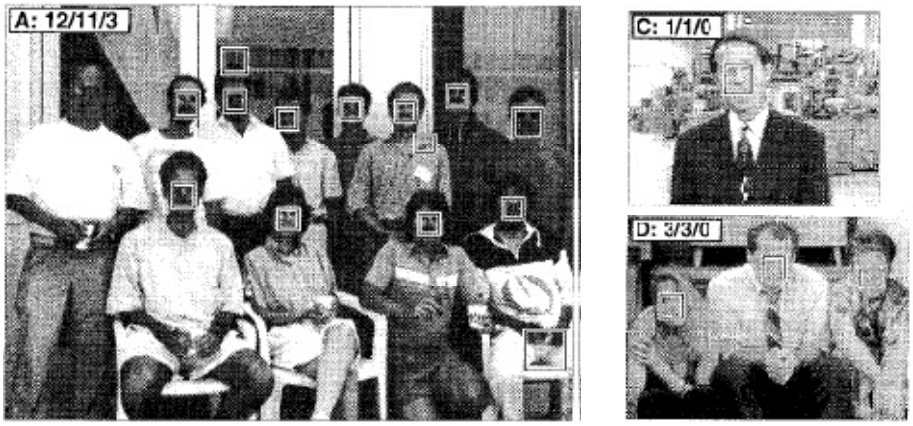


Figura 11. Ejemplo de aplicación de los métodos de aprendizaje supervisado al *tracking*: detección de caras humanas.(Figura extraída de [26]).

El principal problema de estos métodos es que requieren una extensa colección de muestras de cada clase de objeto para el aprendizaje. Además, esta colección debe ser etiquetada manualmente. Una posible aproximación es combinar el aprendizaje supervisado con co-entrenamiento [27]: se entrenan dos clasificadores usando un pequeño conjunto de muestras etiquetadas, donde las características de objeto que utiliza cada uno son independientes. Una vez realizado el entrenamiento, cada clasificador asigna los datos no etiquetados al conjunto de entrenamiento del otro clasificador.

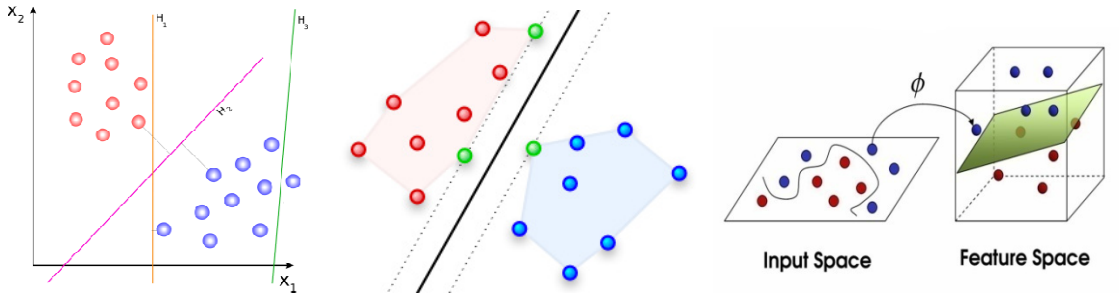
A continuación se describen algunas de las técnicas más comúnmente utilizadas en el aprendizaje de patrones.

a) SVM (Máquinas de vectores soporte/*Support Vector Machines*)

Tomando los datos de entrada como dos conjuntos de vectores en un espacio n-dimensional, una máquina de vectores soporte construye un hiperplano separador

en ese espacio. El objetivo es maximizar el margen del hiperplano, definido como la distancia entre el propio hiperplano y los vectores de datos más cercanos (Figura 12(a) y (b)). Estos vectores de datos que se utilizan para colocar el hiperplano se denominan vectores soporte [28]. En el contexto de detección de objetos, las clases de datos corresponden al objeto (muestras positivas) y al resto de la imagen (muestras negativas).

A pesar de ser un clasificador lineal, los SVM también pueden ser usados como clasificadores no lineales aplicando el truco del kernel al vector de características de entrada extraído. Este truco transforma los datos a un espacio de mayor dimensión, en el cual resulte más fácil la separación (Figura 11(c)). Los kernel usados para tal fin suelen ser funciones polinómicas o radiales, como por ejemplo el kernel Gaussiano o una función sigmoide.



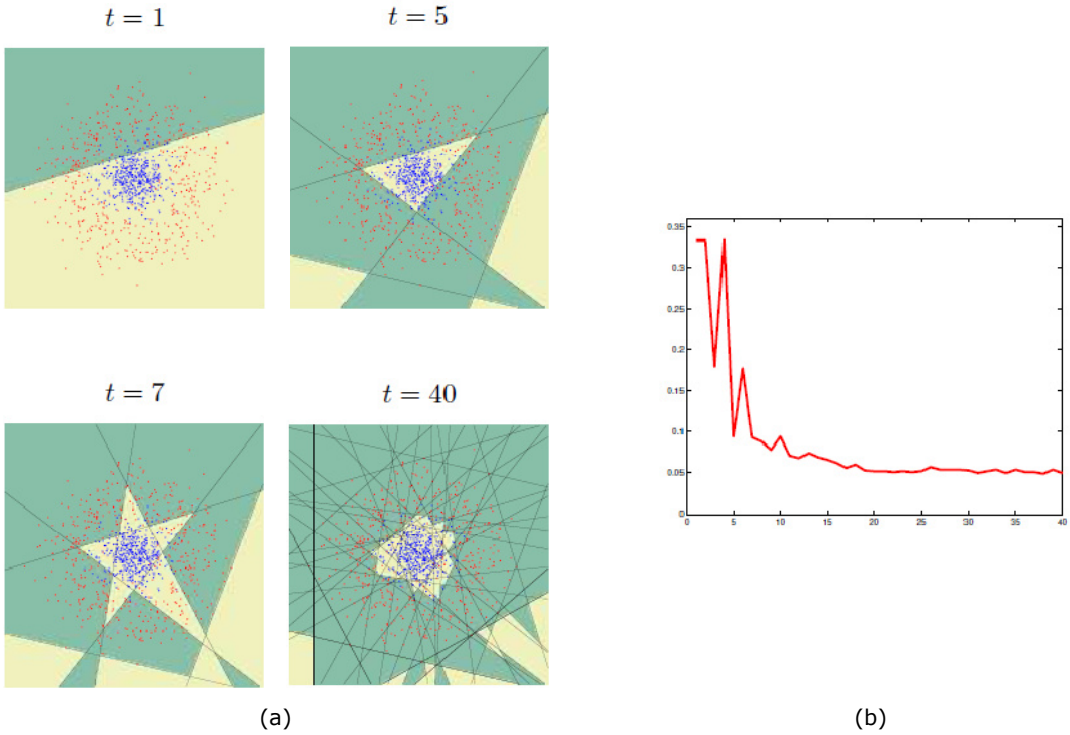
(a) (b) (c)

Figura 12. Funcionamiento de SVM. (a) Distintos hiperplanos de separación en un espacio bidimensional. H3 no separa las dos clases, H1 sí pero con un margen pequeño, y H2 con el máximo margen. (b) Hiperplano de separación de dos conjuntos de datos, margen obtenido y vectores soporte empleados para el cálculo. (c) Aplicación del truco del kernel para transformar los datos a un espacio de mayor dimensión en el cual sean más fácilmente separables.

b) Mejora adaptativa (*Adaboost*)

Este método iterativo obtiene clasificadores muy precisos mediante la combinación de muchos clasificadores base, siendo cada uno de ellos moderadamente preciso [29]. En la fase de entrenamiento del algoritmo, el primer paso es construir una distribución inicial de pesos sobre el conjunto de entrenamiento, que asignan más o menos importancia a los datos incluidos en dicho conjunto. El mecanismo de mejora selecciona entonces un clasificador base que proporciona el mínimo error, donde el error es proporcional a los pesos de los datos mal clasificados. A

continuación los pesos de los datos mal clasificados se incrementan y se entrena otro clasificador en la siguiente iteración, el cual hace especial énfasis en el aprendizaje de aquellos datos que habían sido mal clasificados (Figura 13).



(a) (b)

Figura 13. Funcionamiento de *Adaboost*. (a) Varias iteraciones del algoritmo para dos conjuntos de datos no separables linealmente. (b) Gráfico del error a lo largo de las iteraciones. (Figuras extraídas de [30]).

c) Redes neuronales

Las redes neuronales consisten en una simulación de las propiedades observadas en los sistemas neuronales biológicos a través de modelos matemáticos. Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida (Figura 14(a),(b)). Esta salida viene dada por tres funciones:

1. Una función de propagación (también conocida como función de excitación), que por lo general consiste en el sumatorio de cada entrada multiplicada por el peso de su interconexión (valor neto). Si el peso es positivo, la conexión se denomina *excitatoria*; si es negativo, se denomina *inhibitoria*.

2. Una función de activación, que modifica a la anterior. Puede no existir, siendo en este caso la salida la misma función de propagación.

3. Una función de transferencia, que se aplica al valor devuelto por la función de activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que queramos darle a dichas salidas. Algunas de las más utilizadas son la función sigmoidea (para obtener valores en el intervalo [0,1]) y la tangente hiperbólica (para obtener valores en el intervalo [-1,1]).

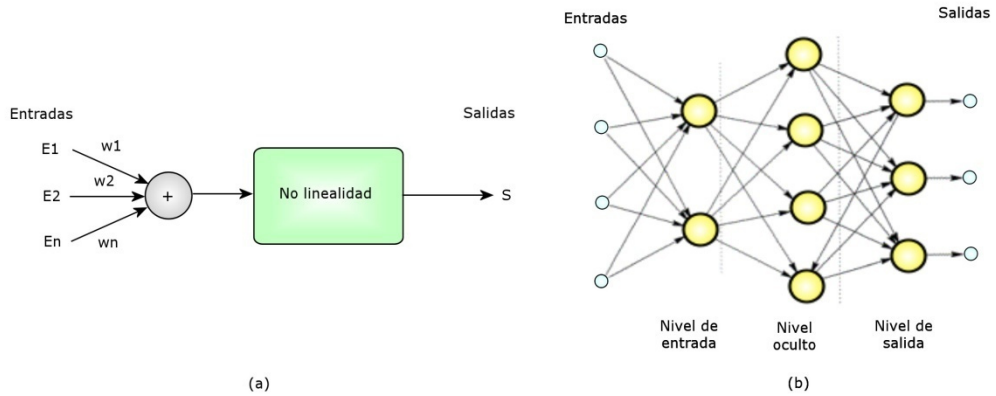


Figura 14. Esquema de una red neuronal. (a) Neurona simple. (b) Red neuronal de tres capas.

2.4. METODOS DE *TRACKING* O SEGUIMIENTO DE UN OBJETIVO

La finalidad de un *tracker* es generar la trayectoria de un objetivo a lo largo de un período de tiempo, localizando su posición en cada plano del vídeo. El *tracker* podría incluso proporcionar la región completa de la imagen que es ocupada por el objeto en cada instante. Las tareas de detectar el objeto y establecer una correspondencia (localizar dicho objeto en otra imagen) a lo largo de los *frames* pueden realizarse de forma separada o conjunta. En el primer caso, las regiones que componen el objeto en cada plano se obtienen por medio de un algoritmo de detección, siendo la función del *tracker* en este caso hacer corresponder los objetos a lo largo de los diferentes *frames*. En el segundo caso, la región del objeto y la correspondencia se estiman conjuntamente actualizando iterativamente la localización del objeto y la información de la región obtenidas de *frames* anteriores.

En cualquier caso, los objetos son representados usando una de las formas o apariencias descritas anteriormente. El modelo elegido para representar el objeto limita el tipo de movimiento o deformación a los que podrá adaptarse. En la Figura

15 se muestran algunos de los métodos de *tracking* que se describirán a continuación. Estos métodos se resumen en el esquema de la Figura 16.

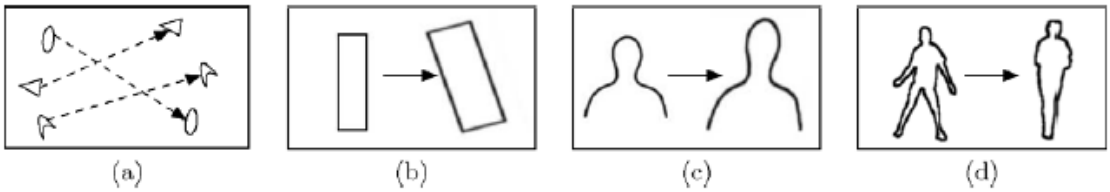


Figura 15. Diferentes aproximaciones al seguimiento de objetos. (a) Correspondencia multipunto, (b) Transformación paramétrica de un patrón rectangular, (c, d) Dos ejemplos de evolución del contorno. (Figura extraída de [9]).

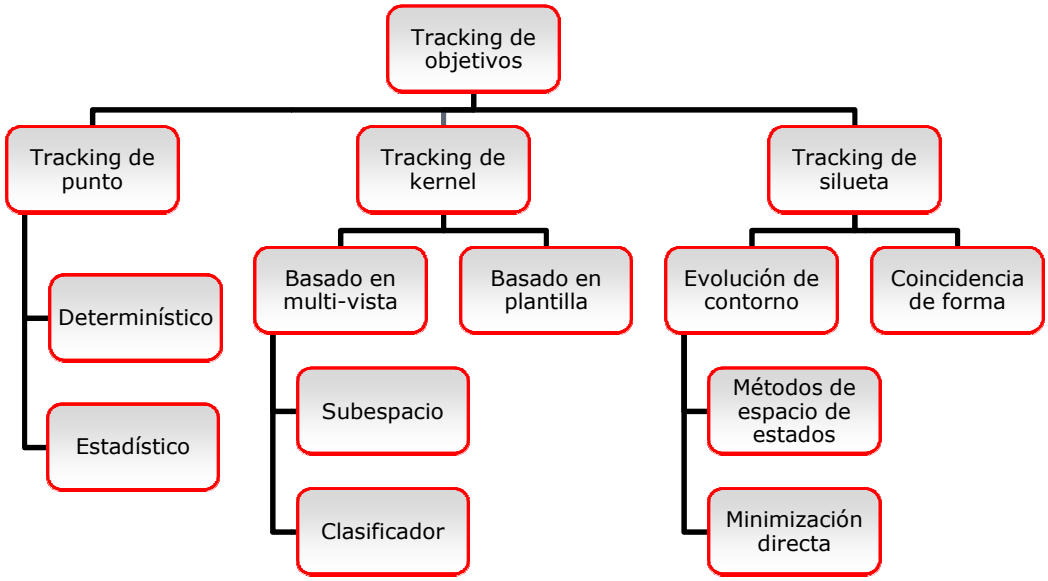


Figura 16. Esquema general de las diferentes técnicas de *tracking* revisadas en la sección.

2.4.1. *Tracking* de punto

Los objetos detectados en *frames* consecutivos son representados mediante puntos, y la asociación de los puntos se basa en el estado previo del objeto (que puede incluir su posición y su movimiento). Esta aproximación requiere de un mecanismo externo que detecte los objetos en cada *frame*. Un ejemplo de la correspondencia entre objetos se muestra en la Figura 17.

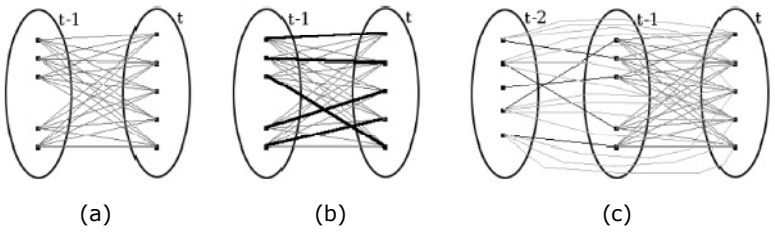


Figura 17. Correspondencia entre puntos. (a) todas las posibles asociaciones de un punto (objeto) del cuadro en t-1 con puntos (objetos) del cuadro en t, (b) conjunto único de asociaciones resaltadas con líneas en negrita, (c) correspondencias multicuadro. (Figura extraída de [9]).

Realizar esta correspondencia es un problema complicado – especialmente en presencia de oclusiones, errores de detección, o entradas y salidas de objetos. Los métodos de correspondencia de puntos pueden ser divididos en dos amplias categorías, los métodos determinísticos y los métodos probabilísticos. Los métodos determinísticos utilizan heurísticas de movimiento cualitativas para solventar el problema de correspondencia. Por otro lado, los métodos probabilísticos toman explícitamente la medida del objeto y tienen en cuenta la incertidumbre para establecer la correspondencia.

a) Métodos determinísticos

Estos métodos definen un coste a la asociación de cada objeto en el plano *t-1* con un objeto en el plano *t* usando un conjunto de restricciones de movimiento. La minimización del coste de correspondencia es formulada como un problema de optimización combinatoria. Una solución, que consiste en una correspondencia uno a uno entre todas las posibles asociaciones, puede ser obtenida por métodos de asignación óptima, como por ejemplo el algoritmo Húngaro [31] o métodos de búsqueda por gradiente (Figura 18). El coste de correspondencia es definido normalmente usando una combinación de las siguientes restricciones:

- Proximidad: asume que la localización del objeto no cambia notablemente de un *frame* al siguiente. (Figura 19(a)).

- Velocidad máxima: define un límite superior para la velocidad del objeto y limita las correspondencias posibles al vecindario circular alrededor del objeto. (Figura 19(b)).

- Cambio pequeño de velocidad (movimiento suave) asume que la dirección y velocidad del objeto no cambia drásticamente. (Figura 19(c)).

- Movimiento común: obliga a que la velocidad de los objetos en un pequeño vecindario sea similar. Esta restricción es adecuada para objetos representados por múltiples puntos. (Figura 19(d)).

- Rigidez: asume que los objetos del mundo 3D son rígidos, luego las distancias entre dos puntos cualesquiera del objeto se mantendrán sin cambios. (Figura 19(e)).

- Uniformidad próxima: es una combinación de las restricciones de proximidad y de cambio suave de velocidad.

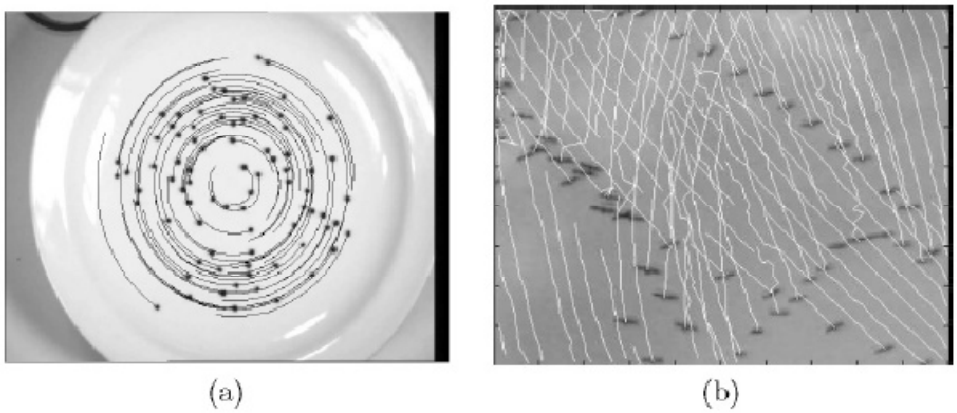


Figura 18. Resultados de dos algoritmos de correspondencia de puntos. (a) Seguimiento usando el algoritmo propuesto por [32] para detectar puntos negros en un plato blanco (© 2001 IEEE). (b) Seguimiento de pájaros usando el algoritmo propuesto por [33] (© 2003 IEEE). (Figura extraída de [9]).

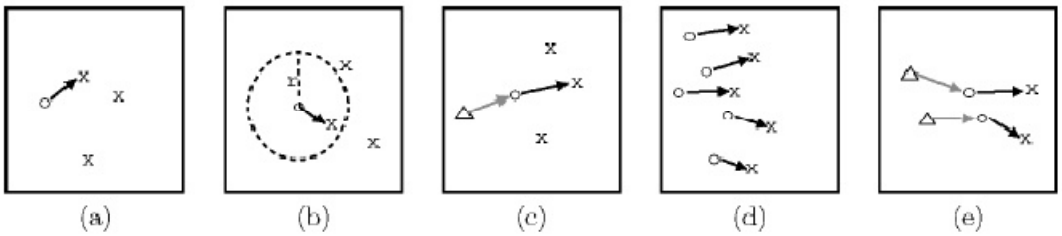


Figura 19. Diferentes restricciones de movimiento. (a) proximidad, (b) velocidad máxima (r denota al radio), (c) pequeños cambios de velocidad, (d) movimiento común, (e) restricciones de rigidez. △

denota la posición del objeto en el *frame t-2*, ○ denota la posición del objeto en el *frame t-1*, y x denota la posición del objeto en el *frame t*. (Figura extraída de [9]).

b) Métodos estadísticos

Las medidas obtenidas de sensores de vídeo irremediablemente contienen ruido. Además, los movimientos de los objetos pueden sufrir perturbaciones aleatorias. Los métodos de correspondencia estadística solucionan estos problemas teniendo en cuenta la incertidumbre de las medidas y del modelo durante la estimación del estado del objeto. Estos métodos emplean una aproximación del espacio de estados para modelar propiedades del objeto tales como posición, velocidad y aceleración. Las medidas usualmente consisten en la posición del objeto en la imagen, obtenidas mediante un mecanismo de detección (Figura 18).

Considérese un objeto en movimiento en la escena. La información que representa al objeto, por ejemplo la localización, está definida por una secuencia de estados $X^t : t = 1, 2,....$ El cambio de estado a lo largo del tiempo está gobernado por la ecuación dinámica,

$$X^t = f^t(X^{t-1}) + W^t \quad (1)$$

Donde $W^t : t=1,2,\ldots$ es ruido blanco. La relación entre la medida y el estado está especificada por la ecuación de medida $Z^t = h^t(X^t, N^t)$, donde N^t es ruido blanco y es independiente de W^t. El objetivo del *tracking* es estimar el estado X^t dadas todas las medidas hasta el momento o, equivalentemente, construir la función de densidad de probabilidad $p(X^t|Z^{1,\ldots,t})$. Una solución teóricamente óptima es la proporcionada por un filtro recursivo Bayesiano que soluciona el problema en dos pasos. El paso de predicción usa una ecuación dinámica y la f.d.p. ya calculada para el estado en el instante *t-1* para derivar la f.d.p. a priori del estado actual, esto es, $p(X^t|Z^{1,\ldots,t-1})$. Entonces, el paso de corrección emplea la función de verosimilitud $p(Z^t|X^t)$ de la medida actual para calcular la f.d.p. a posteriori $p(X^t|Z^{1,\ldots,t})$. En caso de que la medida se deba únicamente a la presencia de un objeto en la escena, el estado puede ser estimado fácilmente con los dos pasos definidos anteriormente. Por otra parte, si hay múltiples objetos, las medidas necesitan ser asociadas con los estados de los correspondientes objetos.

- Estimación del estado de un sólo objeto

Para este caso, si f^t y h^t son funciones lineales y tanto el estado inicial X^1 como el ruido tienen una distribución Gaussiana, entonces la estimación óptima del estado viene dada por el Filtro de Kalman, el cual modela la probabilidad a posteriori del estado como una gaussiana. En el caso general, esto es, cuando se asume que la función de densidad de probabilidad a posteriori del estado del objeto no tiene por qué ser una gaussiana, la estimación del estado puede ser resuelta empleando el Filtro de Partículas [34] (ver capítulo 3 para más información). En caso de que las funciones f^t y h^t sean no lineales, pueden ser linealizadas empleando la expansión en series de Taylor para obtener el Filtro de Kalman Extendido (EKF) [35]. Al igual que el Filtro de Kalman, el Filtro de Kalman Extendido asume que el estado se distribuye como una Gaussiana.

- Estimación del estado y asociación de datos de múltiples objetos

Se puede definir la trayectoria del objeto como una secuencia de medidas que se asume que son originadas por el mismo objeto. Cuando se hace *tracking* de múltiples objetos, la clave está en asociar las medidas observadas a los objetos individuales adecuados.

Si se usa el filtro de Kalman o el filtro de partículas, esta asociación se debe realizar de manera determinística, esto es, el problema de correspondencia necesita ser solventado antes de poder aplicar estos filtros. El método más sencillo para resolver la correspondencia es usar la aproximación al vecino más cercano. Sin embargo, si los objetos están cerca de otros, entonces siempre existe la posibilidad de que la correspondencia sea incorrecta. Una medida asociada incorrectamente puede hacer que el filtro falle y no converja. Existen varias técnicas estadísticas de asociación de datos para abordar este problema. Un análisis detallado de estas técnicas puede encontrarse en [35]. Dos de las técnicas más usadas en asociación de datos son JPDAF (*Joint Probability Data Association Filtering*) y MHT (*Multiple Hipothesis Tracking*). A continuación se ofrece una breve descripción de estas técnicas.

JPDAF

Para un determinado instante *t*, un algoritmo de *tracking* de múltiples objetos contará con *m* medidas y tendrá *N* trayectorias, una para cada uno de los *N* objetos en escena. El problema es asignar las medidas disponibles a las trayectorias existentes.

JPDAF es un estimador en el que se combina: la formulación probabilística de JPDA, para obtener el conjunto de valores de probabilidad de asociación conjunta de las distintas hipótesis de seguimiento, a partir de todas las posibles soluciones de asociación; con la formulación óptima de la regla de Bayes, para estimar el estado de estas hipótesis en función de las asociaciones.

JPDAF se define, por tanto, como la unión de un Filtro de Kalman y un JPDA para resolver de forma probabilística el problema de *tracking* de múltiples objetivos, bajo dos condiciones:

- los modelos de observación y actuación de los objetivos implicados han de ser lineales y estar afectados por ruidos gaussianos, blancos, de media nula e incorrelados, de modo que se pueda aplicar la definición básica del filtro de Kalman

- el número de objetivos o hipótesis de estimación ha de ser constante. Además, en cada solución de asociación solo puede haber una medida asociada con cada objetivo, y cada medida solo puede además asociarse con un único objetivo

MHT

Si la correspondencia del movimiento se establece empleando únicamente dos cuadros, siempre existe la posibilidad de generar una correspondencia incorrecta. Se pueden obtener mejores resultados en el *tracking* si la decisión de correspondencia es retrasada hasta que se han examinado varios cuadros. El algoritmo MHT mantiene varias hipótesis de correspondencia para cada objeto a lo largo de los cuadros. La trayectoria final del objeto es el conjunto de correspondencias más probable a lo largo del tiempo de observación.

El algoritmo tiene la habilidad de crear nuevas trayectorias para objetos entrantes en la escena, y de terminarlas cuando los objetos salen de ella. También es capaz de manejar oclusiones, es decir, continuar la trayectoria aunque falten algunas de las medidas.

2.4.2. *Tracking* de *kernel*

El *kernel* (núcleo) del objeto se refiere a la forma del objeto y su apariencia. Por ejemplo, el *kernel* puede ser un patrón rectangular o una forma elíptica con un histograma asociado. El seguimiento de un objeto se realiza mediante una evaluación del movimiento del *kernel* en *frames* sucesivos. Dividiremos estos métodos de *tracking* en dos categorías, atendiendo a la representación de apariencia usada.

a) Modelos de apariencia basados en plantillas y densidad

Estos modelos han sido ampliamente usados debido a su relativa simplicidad y bajo coste computacional. La aproximación más común es la denominada coincidencia de plantilla. Se trata de un método de fuerza bruta que busca en toda la imagen una región similar a la plantilla del objeto. Se suele utilizar la correlación cruzada para medir la similitud, y características de color o intensidad para formar la plantilla. También se utiliza el gradiente de la imagen dada la sensibilidad de la intensidad a los cambios de iluminación. La principal limitación de este método es el coste computacional derivado de la búsqueda por fuerza bruta. Para reducirlo, los investigadores normalmente limitan la búsqueda a la vecindad de la posición previa (ver Figura 20).

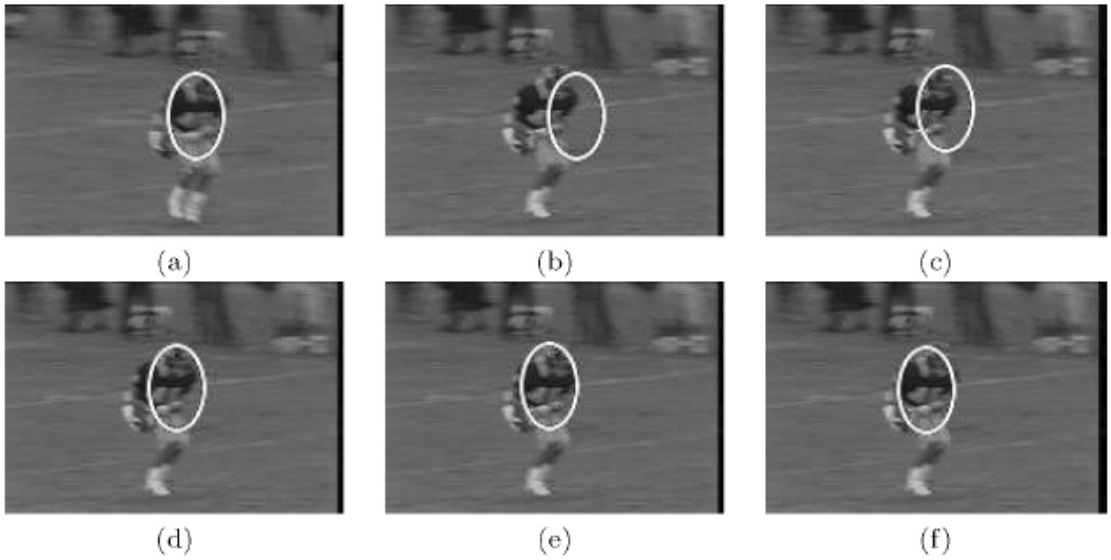


Figura 20. Iteraciones de *tracking* utilizando *mean-shift*. Mejora el rendimiento al buscar sólo en la vecindad de la posición en el instante anterior. (a) Localización estimada del objeto en el instante *t-1*. (b) cuadro en el instante *t* con posición inicial estimada utilizando la posición previa del objeto, (c), (d), (e) actualización de posición utilizando *mean-shift*, (f) posición final del objeto en el instante *t*. (Figura extraída de [9]).

b) Modelos de apariencia multivista

En los métodos previos, los modelos de apariencia (histogramas, plantillas…) son generados *online*. Por ello estos modelos representan la información obtenida del objeto en las observaciones más recientes. Los objetos pueden parecer diferentes desde diferentes vistas, y si la vista cambia drásticamente durante el seguimiento, el modelo de apariencia deja de ser válido y el seguimiento puede ser incorrecto. Para superar esta limitación, se pueden aprender *offline* diferentes vistas del objeto y usarlas para el seguimiento.

Para el aprendizaje se pueden utilizar diversas técnicas, como el análisis por componentes principales (PCA [36]) o las máquinas de vectores soporte (SVM [37]).

2.4.3. *Tracking* de silueta

El *tracking* se realiza estimando la región del objeto en cada *frame*. Los métodos basados en la silueta utilizan información codificada dentro de la región del objeto. Esta información puede encontrarse en forma de modelos de densidad de

aparición o forma que se utilizan para generar mapas de bordes. Dados unos modelos de un objeto, se realiza un seguimiento de las siluetas mediante la búsqueda de coincidencias o bien siguiendo la evolución del contorno (Figura 16(c), (d)). Ambos métodos pueden considerarse en términos generales como una segmentación del objeto aplicada en el dominio temporal usando la información generada en los *frames* previos.

Los objetos pueden tener formas complejas, por ejemplo, manos, cabeza y hombros que no pueden ser bien descritas por formas geométricas simples. Los métodos basados en silueta proporcionan una descripción de la forma precisa para estos objetos. El objetivo de un algoritmo de *tracking* de este tipo es encontrar en cada cuadro la región del objeto a través de un modelo de objeto generado usando los cuadros anteriores. Este modelo puede ser un histograma de color, los bordes del objeto o el contorno. Dividiremos estos métodos en dos categorías, denominadas coincidencia de forma y evolución de contorno.

a) Coincidencia de forma

Trata de buscar la silueta del objeto en el cuadro actual. Se puede llevar a cabo de manera similar a como se hacía en el *tracking* basado en coincidencia de plantilla (sección 2.4.2), donde la silueta del objeto y su modelo asociado es buscado en el *frame* actual. La búsqueda se realiza calculando la similitud del objeto con el modelo generado a partir de la hipotética silueta del objeto obtenida en el cuadro anterior. En esta aproximación, se asume que la silueta sólo se traslada del cuadro actual al siguiente, por lo que objetos no rígidos no pueden ser manejados explícitamente. El modelo de objeto, que suele ser un mapa de bordes, es reinicializado para ajustarse a los cambios de apariencia en cada cuadro una vez el objeto es localizado. Esta actualización es requerida para solucionar los problemas en el *tracking* debidos a cambios en el punto de vista o condiciones de iluminación, así como movimiento de objetos no rígidos [38].

b) Evolución de contorno

En esta aproximación se evoluciona iterativamente un contorno inicial del cuadro previo hasta su nueva posición en el cuadro actual (Figura 21). Esta evolución del contorno requiere que parte del objeto en el cuadro actual solape con la región del objeto del cuadro previo. Este método de *tracking* se puede llevar a cabo a través de dos aproximaciones. La primera emplea modelos de espacio de estados para

modelar la forma del contorno y el movimiento, mientras que la segunda evoluciona el contorno a través de la minimización directa de alguna función de energía [39], mediante técnicas como el descenso por gradiente.

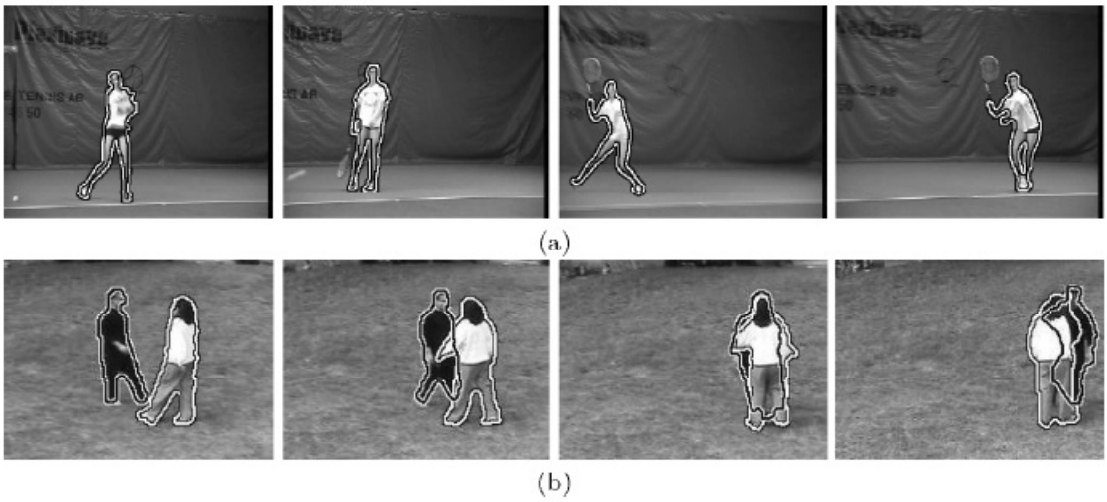


Figura 21. Resultados del *tracking* de contorno. (a) *Tracking* de una tenista, (b), *tracking* en presencia de oclusión, usando el método propuesto por [40]. (Figura extraída de [9]).

3. EL FILTRO DE PARTICULAS

3.1. INTRODUCCION

Muchos problemas físicos y científicos requieren estimar el estado de un sistema que cambia a lo largo del tiempo usando una secuencia de medidas ruidosas realizadas sobre el sistema. Una posible solución al problema la representan las aproximaciones de espacio de estados (*state-space models*). Esta aproximación centra su atención en el vector de estado del sistema, el cual contiene toda la información relevante necesaria para describir el sistema bajo investigación. Por ejemplo, en problemas de seguimiento, esta información podría estar relacionada con las características cinemáticas del objetivo, o con características extraídas de la imagen (ver Sección 2.2). Alternativamente, en problemas de econometría, podrían estar relacionadas con el flujo monetario, los ratios de interés, la inflación… El vector de medidas representa observaciones (ruidosas) que están relacionadas con el vector de estados.

Para analizar y hacer inferencia acerca de un sistema dinámico, se requieren al menos dos modelos: primero, un modelo que describa la evolución del estado con el tiempo (ecuación de estado o ecuación de movimiento) y, segundo, un modelo que relacione las medidas ruidosas con el estado (ecuación de observación o verosimilitud). La formulación probabilística del espacio de estados y el requerimiento de actualización de la información con la recepción de nuevas medidas son idealmente llevados a cabo por la aproximación Bayesiana. Esta aproximación proporciona una solución general para problemas de estimación de estados dinámicos.

3.2. FILTRADO BAYESIANO RECURSIVO

En la aproximación Bayesiana a la estimación de estados dinámicos, el objetivo es construir la f.d.p. a posteriori del estado basándose en toda la información disponible, incluyendo el conjunto de medidas recibidas. Dado que esta f.d.p. agrupa toda la información estadística disponible, se trata de una solución completa al problema de estimación, y se puede obtener una estimación óptima del estado (respecto a cualquier criterio).

Para muchos problemas, se requiere una estimación cada vez que se recibe una nueva medida. En este caso, un filtro recursivo es una solución conveniente. Una aproximación mediante filtrado recursivo permite que los datos recibidos sean procesados secuencialmente en vez de en bloque, por lo que no es necesario almacenar todo el conjunto de datos o reprocesar datos existentes cuando una nueva medida está disponible. Un filtro de este tipo consta de dos etapas básicas: predicción y actualización. La etapa de predicción utiliza el modelo de movimiento para predecir el estado de la f.d.p. de un instante al siguiente. Dado que el estado está normalmente sujeto a perturbaciones desconocidas (modeladas como ruido Gaussiano), la predicción generalmente traslada, deforma y expande la f.d.p.. La etapa de actualización utiliza la última medida para modificar la predicción de la f.d.p..

Para definir el problema del seguimiento, considérese un sistema de ecuaciones movimiento y verosimilitud dependientes del tiempo

$$x_t = f_t(x_{t-1}, v_{t-1}) \quad (2)$$

$$z_t = h_t(x_t, n_t) \quad (3)$$

Donde x_t denota el estado del sistema en el instante *t*, y z_t denota la medida como una función de un estado del sistema desconocido en el instante *t*. Nótese que implícitamente asumimos que el estado del proceso admite una propiedad de Markov de orden 1, lo que significa que x_t depende únicamente del estado en el anterior instante de tiempo *t-1*.

La ecuación 2 es la ecuación de actualización del sistema o ecuación de estado, y representa la evolución del estado del sistema del instante *t-1* al instante *t*. Depende de los estados previos del sistema x_{t-1}, y de un error estocástico v_{t-1} que representa la incertidumbre en la actualización del estado. Es decir, implícitamente define la f.d.p. a priori $p(x_t|x_{t-1})$. La ecuación 3 se conoce como ecuación de observación, y define cómo la medida z_t depende del valor actual del estado x_t y de un término de error n que representa la incertidumbre en la medida del estado. Dado que n_t es una variable estocástica, esta ecuación también define una verosimilitud que llamaremos f.d.p. de medida, $p(z_t|x_t)$.

El estimador Bayesiano para el estado desconocido x_t en el tiempo *t* se deriva de la ecuación de actualización de estado (2), la ecuación de medida (3), y la estadística de los parámetros de ruido v_{t-1} y n_t, que se asume conocida:

$$p(x_t|z_t) = \frac{p(z_t|x_t)p(x_t|z_{t-1})}{p(z_t|z_{t-1})} \quad (4)$$

Donde $p(x_t|z_{t-1})$ se deriva de la asunción de Harkov y de la ecuación de Chapman-Kolmogorov:

$$p(x_t|z_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{t-1})dx_{t-1} \quad (5)$$

Las relaciones de recurrencia (4) y (5) constituyen la base de la solución Bayesiana óptima. La propagación recursiva de la densidad a posteriori es sólo una solución conceptual que, en general, no puede ser determinada analíticamente. Existen soluciones en un conjunto restrictivo de casos, entre las que se incluye el Filtro de Kalman, descrito en la siguiente sección. También describiremos como, cuando no existe solución analítica, el filtro de Kalman extendido (EKF) y el filtro de partículas aproximan la solución Bayesiana óptima.

3.3. EL FILTRO DE KALMAN

3.3.1. Introducción al filtro de Kalman

El filtro de Kalman consiste en un conjunto de ecuaciones matemáticas que proveen una solución recursiva óptima al problema de filtrado lineal de datos discretos, mediante el método de mínimos cuadrados. Se trata de una solución óptima en el sentido de que minimiza la covarianza estimada del error. La meta de esta solución consiste en calcular un estimador lineal, insesgado y óptimo, del estado de un sistema en *t* con base en la información disponible en *t-1*, y actualizar, con la información adicional disponible en t, dichas estimaciones [41]. El filtro se desempeña suponiendo que el sistema puede ser descrito a través de un modelo estocástico lineal, en donde el estado tiene una distribución Gaussiana.

El filtro de Kalman es el principal algoritmo para estimar sistemas dinámicos representados en la forma de espacio de estados. En esta representación el sistema es descrito por un conjunto de variables denominadas de estado. El estado contiene toda la información relativa al sistema en un cierto instante de tiempo. Esta información debe permitir la inferencia del comportamiento pasado del sistema, con el objetivo de predecir su comportamiento futuro [42].

Lo que hace al filtro tan interesante es precisamente su habilidad para predecir el estado de un sistema en el pasado, presente y futuro, aún cuando la naturaleza precisa del sistema modelado es desconocida. En la práctica, las variables de estado individuales de un sistema dinámico no pueden ser exactamente determinadas por una medición directa. Dado lo anterior, su medición se realiza por medio de procesos estocásticos que involucran algún grado de incertidumbre en la medición.

El filtro de Kalman tiene como objetivo resolver el problema general de estimar el estado $X \in \Re^n$ de un proceso controlado en tiempo discreto, el cual es dominado por una ecuación lineal en diferencia estocástica de la siguiente forma:

$$X_t = AX_{t-1} + w_{t-1} \tag{6}$$

con una medida $Z \in \Re^n$, que es

$$Z_t = HX_t + v_t \tag{7}$$

La matriz *A* se asume de una dimensión *n x n* y relaciona el estado en el instante previo *t-1* con el estado en el momento *t*. La matriz *H* de dimensión *m x n* relaciona el estado con la medición Z_t. Estas matrices pueden cambiar en el tiempo, pero en general se asumen como constantes.

Las variables aleatorias w_t y v_t representan el ruido o error del proceso y de la medida respectivamente. Se asume que son independientes entre ellas, de naturaleza blanca y con distribución de probabilidad normal:

$$p(w) = \mathcal{N}(0, Q) \tag{8}$$
$$p(v) = \mathcal{N}(0, R) \tag{9}$$

En la práctica las matrices de covarianza de la perturbación del proceso, Q, y de la perturbación de la medida, R, podrían cambiar en el tiempo aunque, por simplicidad, en general se asumen que son constantes.

3.3.2. Algoritmo

El filtro de Kalman estima el estado utilizando un control de retroalimentación: estima el estado en un instante de tiempo determinado y entonces obtiene la retroalimentación por medio de los datos observados.

Desde este punto de vista las ecuaciones que se utilizan para derivar el filtro de Kalman se pueden dividir en dos grupos: las que actualizan el estado en el tiempo o **ecuaciones de predicción** y las que actualizan el estado a partir de los datos observados o **ecuaciones de actualización**. Las del primer grupo son responsables de proyectar la estimación del estado y de la covarianza del error al momento *t* tomando como referencia el estado en el momento *t-1*. El segundo grupo de ecuaciones son responsables de la retroalimentación, es decir, incorporan nueva información dentro de la estimación a priori anterior con lo cual se llega a una estimación (a posteriori) mejorada del estado.

Las ecuaciones que actualizan el tiempo pueden ser vistas como ecuaciones de predicción, mientras que las ecuaciones que incorporan nueva información pueden considerarse como ecuaciones de corrección. Efectivamente, el algoritmo de estimación final puede definirse como un algoritmo de predicción-corrección: se pronostica el nuevo estado y su incertidumbre a partir de la información disponible, y se corrige la predicción con la nueva medida, minimizando estadísticamente el error. Este ciclo se muestra en la Figura 22.

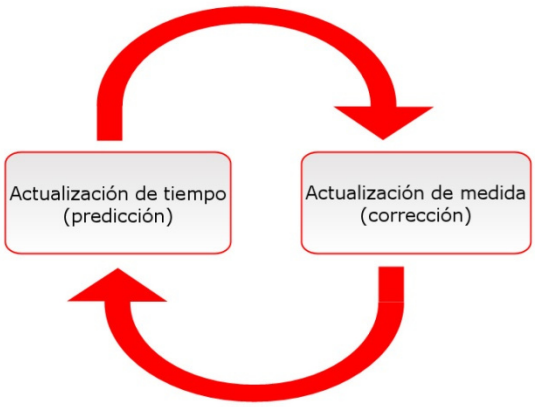


Figura 22. Ciclo de funcionamiento del filtro de Kalman. La predicción proyecta la estimación del estado adelante en el tiempo, mientras que la corrección ajusta la estimación proyectada con una medida del instante actual.

Las ecuaciones específicas para la predicción y la corrección del estado son detalladas en el Algoritmo 1.

Ecuaciones de predicción

$$\hat{X}_t^* = A\hat{X}_{t-1} \quad (10)$$

$$P_t^* = AP_{t-1}A^T + Q \quad (11)$$

Ecuaciones de corrección

$$K_t = P_t^* H^T (HP_t^* H^T + R)^{-1} \quad (12)$$

$$\hat{X}_t = \hat{X}_t^* + K_t(Z_t - H\hat{X}_t^*) \quad (13)$$

Algoritmo 1. Filtro de Kalman discreto

Nótese cómo las ecuaciones de predicción realizan estimaciones del estado X_t^* y la covarianza del error P_t^* hacia delante desde *t-1* a *t*. La matriz *A* relaciona el estado en el momento previo *t-1* con el estado en el momento actual *t* (ecuación (6)). *Q* representa la covarianza de la perturbación aleatoria del proceso que trata de estimar el estado (ecuación (8)).

La primera tarea durante la corrección de la proyección del estado es el cálculo de la ganancia de Kalman, K_t, (ecuación (12)). Este factor de ponderación o ganancia es seleccionado de tal forma que minimice la covarianza del error de la nueva estimación del estado. El siguiente paso es medir el proceso para obtener Z_t y entonces generar una nueva estimación del estado $\hat{X}_t$ que incorpora la nueva observación (ecuación (13)). El paso final consiste en obtener una estimación mejorada de la covarianza del error P_t mediante la ecuación (14).

Después de cada par de actualizaciones, tanto del tiempo como de la medida, el proceso es repetido tomando como punto de partida las nuevas estimaciones del estado y de la covarianza del error. Esta naturaleza recursiva es una de las características más llamativas del filtro de Kalman.

La Figura 23 ofrece un cuadro completo de la operación del filtro, combinando la Figura 22 con las ecuaciones del Algoritmo 1.

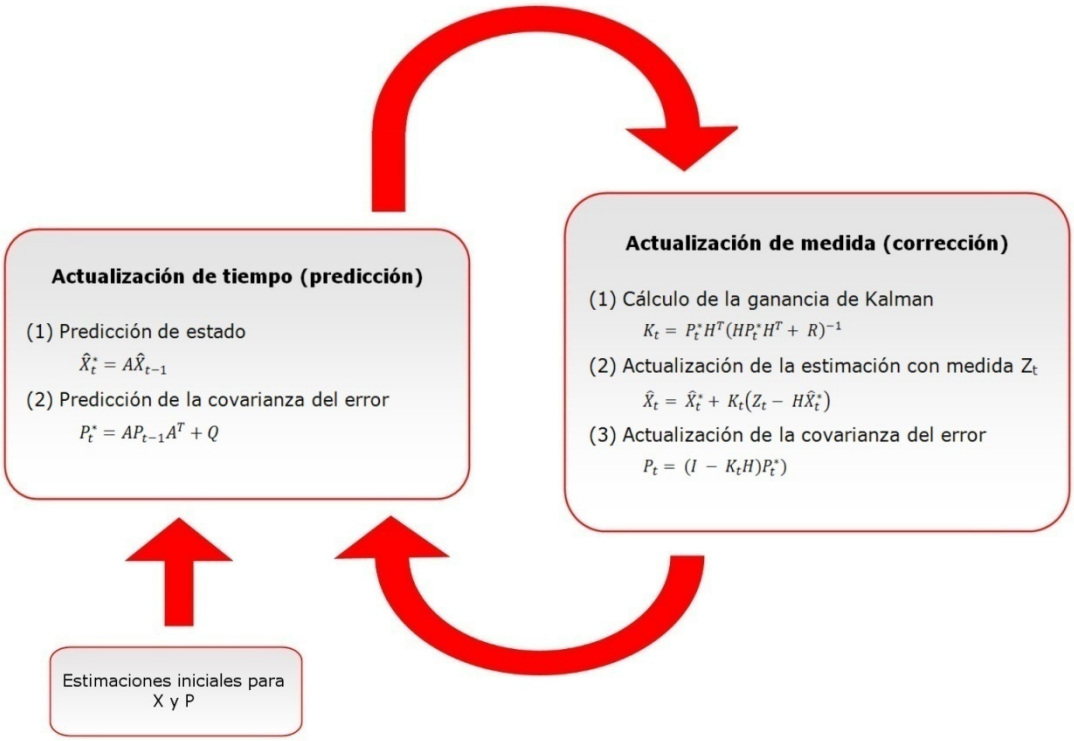


Figura 23. Diagrama completo de operación del filtro de Kalman, combinando el diagrama de alto nivel de la Figura 22 con las ecuaciones del algoritmo 1.

3.3.3. El filtro de Kalman Extendido (EKF)

Si las ecuaciones (2) y (3) no pueden ser reescritas en la forma de (6) y (7), porque las funciones son no lineales, entonces una linealización local de las ecuaciones puede ser una descripción suficiente de la no linealidad.

El filtro de Kalman Extendido está basado en la siguiente aproximación: $p(x_t|z_t)$ es aproximado por una Gaussiana que depende de funciones no lineales, y linealizaciones locales de tales funciones (obtenidas mediante derivación).

EKF utiliza el primer término de la expansión en series de Taylor de la función no lineal. Existe un EKF de mayor orden que retiene más términos de la expansión en series de Taylor, pero la complejidad adicional que introduce lo hacen inviable para un uso generalizado.

Hace tiempo que surgió una mejora del EKF denominada *Unscented Kalman Filter* (UKF). Este filtro considera un conjunto de puntos que son seleccionados determinísticamente de la aproximación Gaussiana de $p(x_t|z_t)$. Estos puntos son propagados sobre la no linealidad, y los parámetros de la aproximación Gaussiana son reestimados. Para algunos problemas, este filtro muestra un mejor rendimiento que EKF ya que aproxima mejor la no linealidad.

Sin embargo, EKF siempre aproxima $p(x_t|z_t)$ a una Gaussiana. Si la verdadera f.d.p. no es Gaussiana (por ejemplo, si es bimodal o fuertemente sesgada), entonces una Gaussiana nunca la describirá correctamente. En estos casos, la aproximación mediante filtro de partículas proporciona una mejor solución en términos de rendimiento.

3.4. EL METODO MONTE CARLO

El método Monte Carlo es un método numérico que permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatorias. El método Monte Carlo fue bautizado así por su clara analogía con los juegos de ruleta de los casinos, el más célebre de los cuales es el de Monte Carlo.

La importancia actual del método Monte Carlo se basa en la existencia de problemas que tienen difícil solución por métodos exclusivamente analíticos o numéricos, pero que dependen de factores aleatorios o se pueden asociar a un modelo probabilístico artificial (resolución de integrales de muchas variables, minimización de funciones, etc.).

El Método de Monte Carlo da solución a estos problemas posibilitando la realización de experimentos con muestreos estadísticos en una computadora. El método es aplicable a cualquier tipo de problema, ya sea estocástico o determinístico. A diferencia de los métodos numéricos que se basan en evaluaciones en *N* puntos en un espacio *M*-dimensional para producir una solución aproximada, el método de Monte Carlo tiene un error absoluto en la estimación que decrece en $1/\sqrt{N}$ en virtud del Teorema Central de Límite.

3.5. FILTRO DE PARTÍCULAS

3.5.1. Introducción

El filtro de partículas es un método empleado para estimar el estado de un sistema que cambia a lo largo del tiempo. Fue propuesto en 1993 por N. Gordon, D. Salmond y A. Smith como filtro *bootstrap* para implementar filtros bayesianos recursivos mediante el método de Monte Carlo. Este algoritmo es también conocido como SIS (*Sequential Importance Sampling*) o algoritmo *Condensation* [43].

Básicamente, el filtro de partículas se compone de un conjunto de muestras (las partículas) y unos valores, o pesos, asociados a cada una de esas muestras. Las partículas representan muestras del espacio de estados (estados posibles) del proceso, y los pesos representan muestras de la f.d.p. a posteriori del estado, dadas las observaciones.

Posee cuatro etapas principales:

- Inicialización
  - Para realizar el seguimiento (por ejemplo de un objeto sobre una secuencia de imágenes), el filtro de partículas "lanza" al azar un conjunto de puntos (sobre el plano de imagen en este caso). En esta etapa de inicialización, el conjunto de partículas se puede crear con un estado aleatorio, o se puede emplear algún tipo de información a priori (tamaño del objeto, posición aproximada…).

- Actualización
  - En función de la similitud del estado de cada partícula respecto al estado de referencia se le asignará un peso a cada uno de ellas.

- Estimación
  - A partir de estos valores, se creará un nuevo conjunto de partículas que constituirá la estimación a priori del estado en el siguiente instante de tiempo. Para ello suelen emplearse métodos de remuestreo probabilísticos que consideran la probabilidad a posteriori de cada una de las partículas, de forma que aquellas que mejor se ajusten a las medidas disponibles darán lugar a nuevas partículas con mayor probabilidad.

- **Predicción**
    - Una vez que se crea el conjunto de partículas para el nuevo instante temporal, se realiza una leve modificación al estado de cada uno de ellos introduciendo algún tipo de ruido aditivo que aporte variabilidad al sistema, con el fin de estimar el estado del objeto en el instante siguiente.

Al terminar la etapa de predicción, se obtiene un nuevo conjunto de partículas al que se le vuelve a aplicar la etapa de actualización, repitiéndose este bucle hasta que termine la secuencia de datos, caso en el cual se volvería a la etapa de inicialización.

3.5.2. Diferencias con el filtro de Kalman

Una limitación del filtro de Kalman es la asunción de que las variables de estado se distribuyen como una Gaussiana. Por ello, el filtro de Kalman proporciona pobres estimaciones para variables de estado que no sigan esta distribución. Esta limitación se puede superar usando el filtro de partículas. En el filtrado de partículas, la densidad de estado condicional en el instante *t* es representada por un conjunto de muestras (partículas) con pesos (probabilidad de muestreo). Los pesos definen la importancia de la muestra, esto es, la frecuencia de observación [43]. Para reducir la complejidad computacional, para cada tupla, se almacena un peso acumulativo c_n, donde $c_N=1$. Las nuevas muestras en el instante *t* son dibujadas desde el paso en el instante previo *t-1* basándose en diferentes esquemas de muestreo [44]. El esquema de muestreo más común es el muestreo por importancia (*importance sampling*), definido en el siguiente apartado.

3.5.3. Algoritmo

La idea del filtro de partículas consiste en representar la f.d.p. a posteriori requerida mediante un conjunto de partículas aleatorias con pesos asociados y calcular la estimación final del estado del sistema en base a estas muestras y pesos [45]. A medida que el número de muestras se hace grande $(N \rightarrow \infty)$, por el método de Monte Carlo, la aproximación a la f.d.p. a posteriori es exacta y la describe completamente, permitiendo al filtro de partículas conseguir una estimación Bayesiana óptima.

Para desarrollar los detalles del algoritmo, denotaremos como $\{x_{0:t}^i\}_{i=1}^{Ns}$ al conjunto de puntos muestreados del espacio de estados, que caracterizan la f.d.p. a posteriori del estado, $p(x_{0:t}|z_{1:t})$, y como $\{w_t^i\}_{i=1}^{Ns}$ a sus pesos asignados. Estos pesos están normalizados de manera que $\sum_i w_t^i = 1$. De esta manera, la f.d.p. a posteriori en el instante t, $p(x_t|z_t)$, puede ser aproximada por

$$p(x_t|z_t) \approx \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i) \tag{15}$$

Así obtenemos una aproximación ponderada discreta de la f.d.p. a posteriori, $p(x_{0:t}|z_{1:t})$. Los pesos se eligen utilizando el principio de muestreo por importancia [46]. Este principio se basa en lo siguiente: supongamos que $p(x) \propto \pi(x)$ es una f.d.p. de la cual es difícil dibujar muestras pero para la cual $\pi(x)$ puede ser evaluada. Adicionalmente, sean $x^i \sim q(x), i = 1, \ldots, N_s$ muestras que son fácilmente generadas de una función propuesta $q(\cdot)$ llamada función de importancia. Entonces, una aproximación ponderada a la f.d.p. $p(\cdot)$ viene dada por

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i) \tag{16}$$

donde

$$w^i \propto \frac{\pi(x^i)}{q(x^i)} \tag{17}$$

es el peso normalizado de la partícula i-ésima.

Aplicando el teorema de Bayes, la fórmula para obtener recursivamente la f.d.p. a posteriori, $p(x_{0:t}|z_{1:t})$ a partir de $p(x_{0:t-1}|z_{1:t-1})$ se deriva como sigue

$$p(x_{0:t}|z_{1:t}) = \frac{p(z_t|x_{0:t}|z_{1:t-1})p(x_{0:t-1}|z_{1:t-1})}{p(z_t|z_{1:t-1})}$$

$$= \frac{p(z_t|x_{0:t},z_{1:t-1})p(x_t|x_{0:t},z_{1:t-1})\, p(x_{0:t-1}|z_{1:t-1})}{p(z_t|z_{1:t-1})}$$

$$= \frac{p(z_t|x_t)p(x_t|x_{t-1})}{p(z_t|z_{1:t-1})} p(x_{0:t-1}|z_{1:t-1}) \tag{18}$$

donde $p(z_t|z_{1:t-1})$ es una constante de proporcionalidad, de manera que

$$p(x_{0:t}|z_{1:t}) \propto p(z_t|x_t)p(x_t|x_{t-1})\, p(x_{0:t-1}|z_{1:t-1}) \tag{19}$$

Volviendo al caso secuencial, en cada iteración, se obtienen muestras que constituyen una aproximación a $p(x_{0:t-1}|z_{1:t-1})$. El objetivo es aproximar la f.d.p. a posteriori $p(x_{0:t}|z_{1:t})$. Para ello, el algoritmo genera nuevas partículas x_t^i y las añade a las ya existentes, $x_{0:t-1}^i$ para formar $x_{0:t}^i$ y actualizar los pesos w_t^i, de tal manera que la nueva medida aleatoria aproxima la f.d.p. a posteriori pretendida.

Si la función de importancia es elegida para poder ser factorizada de manera que

$$q(x_{0:t}|z_{1:t}) = q(x_t|x_{0:t-1}, z_{1:t})q(x_{0:t-1}|z_{1:t-1}) \tag{20}$$

entonces el segundo factor obtenido se corresponde con la función de importancia hasta el instante temporal previo y el primer factor es la función en el instante actual. Así, se pueden obtener muestras $x_{0:t}^i \sim q(x_{0;t}|z_{1:t})$ agregando cada una de las muestras existentes $x_{0:t-1}^i \sim q(x_{0:t-1}|z_{1:t-1})$ con el nuevo estado

$$x_t^i \sim q(x_t|x_{0:t-1}, z_{1:t}) \tag{21}$$

A la hora de actualizar los pesos w_t^i, nos fijamos en la expresión (17) que se corresponde con

$$w_t^i \propto \frac{p(x_{0:t}^i|z_{1:t})}{q(x_{0:t}^i|z_{1:t})} \tag{22}$$

Como el factor de normalización de la f.d.p. a posteriori es desconocido, los pesos sólo se pueden determinar de forma proporcional. Si sustituimos las ecuaciones y en la obtención de los pesos, nos queda la siguiente relación

$$w_t^i \propto \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i)p(x_{0:t-1}^i|z_{1:t-1})}{q(x_t^i|x_{0:t-1}^i, z_{1:t})q(x_{0:t-1}^i|z_{1:t-1})}$$

$$= w_{t-1}^i \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, z_t)} \tag{23}$$

En la estimación recursiva Bayesiana estamos interesados en calcular la f.d.p. a posteriori $p(x_t^i|z_{1:t})$ en cada instante. Si consideramos la función de importancia

como $q(x_t|x_{0:t-1}, z_{1:t}) = q(x_t|x_{t-1}, z_t)$, entonces ésta depende únicamente del estado en el instante anterior y la observación disponible en ese instante. Esto es particularmente útil en el caso común en el que sólo se requiere una estimación filtrada de $p(x_t|z_{1:t})$ en cada instante de tiempo. De aquí en adelante asumiremos este caso, pues al ser el filtro un modelo de Markov de primer orden, no se pierde información. En este escenario, sólo x_t^i necesita ser almacenado; de esta manera, se puede descartar el conjunto anterior $x_{0:t-1}^i$ y la historia de las observaciones $z_{1:t-1}$. Así, las partículas se generan a partir de la función de importancia modificada

$$x_t^i \sim q(x_t|x_{t-1}, z_t) \tag{24}$$

Y los pesos se obtienen según

$$w_t^i \propto w_{t-1}^i \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, z_t)} \tag{25}$$

Con las partículas generadas según (24) y sus correspondientes pesos (25), la f.d.p. a posteriori puede ser aproximada por

$$p(x_t|z_{1:t}) \approx \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i) = p_N(x_t|z_{1:t}) \tag{26}$$

Puede demostrarse que a medida que $N \to \infty$ la ecuación (26) converge a la verdadera f.d.p. a posteriori, $p(x_t|z_{1:t})$.

3.5.4. Degeneración de los pesos

Después de un cierto número de pasos del algoritmo SIS surge un problema: sólo un pequeño número de partículas tienen pesos no despreciables [46]. Este fenómeno se conoce como problema de degeneración de los pesos. El algoritmo SIS presentado no procura ninguna solución pero este problema puede ser cuantificado mediante el tamaño efectivo de la muestra N_{eff} que se puede estimar como

$$N_{eff} \approx \hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2} \tag{27}$$

Donde w_t^i son los pesos normalizados obtenidos según (23). Si todas las partículas tienen el mismo peso, i.e., $w_t^i = \frac{1}{N}$ para $i = 1, \dots, N$, entonces la efectividad de las partículas es máxima $N_{eff} = N$. Pero si todas las partículas menos una tienen peso cero, la efectividad es mínima, $N_{eff} = 1$ [47]. Esto significa que un bajo tamaño muestral efectivo indica una degeneración del algoritmo. Una posible pero poco práctica forma de sobreponerse a la degeneración de los pesos consiste en emplear un número de partículas N muy elevado, esto supondría una carga computacional excesiva. Otra posibilidad es el uso de una buena función de importancia para la elección de las partículas. De todas maneras, estas posibilidades sólo consiguen retardar la aparición del problema, no lo solucionan.

3.5.5. Selección de la función de importancia

La elección de la densidad de importancia $q(x_t^i|x_{t-1}^i, z_t)$ es una de las partes más críticas dentro del diseño de un filtro de partículas. A la hora de buscar una densidad óptima se tiene en cuenta que una manera de limitar la degeneración del algoritmo SIS es el empleo de una función de importancia que minimice la varianza de los pesos, de modo que N_{eff} sea maximizada. Doucet et al. [46] muestran que la elección óptima es $q\left(x_t^i|x_{t-1}^i, z_t\right) = p(x_t|x_{t-1}^i, z_t)$ ya que en este caso la varianza de los pesos es cero. Si la función de importancia óptima es escrita como

$$q\left(x_t^i|x_{t-1}^i, z_t\right) = p\left(x_t|x_{t-1}^i, z_t\right) = \frac{p(z_t|x_t, x_{t-1}^i)p(x_t|x_{t-1}^i)}{p(z_t|x_{t-1}^i)} \tag{28}$$

Entonces la sustitución en (23) muestra el cálculo de los pesos

$$w_t^i \propto w_{t-1}^i \, p\left(z_t|x_{t-1}^i\right) = w_{t-1}^i \int p(z_t|x_t) \, p\left(x_t|x_{t-1}^i\right) dx_t \tag{29}$$

Se aprecia cómo se pueden obtener los pesos en el instante t antes de propagar las partículas. En general, no somos capaces de muestrear la f.d.p. $p\left(x_t|x_{t-1}^i, z_t\right)$ y la integral (29) no dispone de forma analítica. Sin embargo, para algunos supuestos, como puede ser un modelo de estado Gaussiano, es posible una evaluación analítica.

Por otro lado, una elección subóptima pero sencilla es emplear como función de importancia la función a priori,

$$q(x_t|x_{t-1}^i, z_t) = p(x_t|x_{t-1}^i) \tag{30}$$

Si sustituimos (30) en la relación (23) los pesos se actualizan cada instante de tiempo a través de la verosimilitud de la nueva observación,

$$w_t^i \propto w_{t-1}^i \, p(z_t|x_t^i) \tag{31}$$

Ahora no es posible obtener los nuevos pesos sin haber propagado previamente las partículas.

3.5.6. Remuestreo

El empleo del remuestreo en determinados instantes del algoritmo permite solucionar el problema de la degeneración de los pesos. El remuestreo elimina partículas con pesos muy pequeños y replica aquellas con un mayor peso cuando se observa que el tamaño efectivo de la muestra, N_{eff}, cae por debajo de un umbral predefinido, N_T.

Dentro de las diversas técnicas de remuestreo, aquí se propone una de las formas más sencillas, conocida generalmente como remuestreo multinomial.

La medida aleatoria obtenida en el instante t, $\{x_t^i, w_t^i\}$ se transforma durante el remuestreo en el nuevo conjunto $\{x_t^{*\,i}, \frac{1}{N}\}$, donde todas las partículas tienen el mismo peso. Cada nueva partícula $\{x_t^{*\,i}\}$ se genera remuestreando la función a posteriori aproximada

$$p(x_t|z_t) \approx \sum_{i=1}^{N} w_t^i \delta(x_t - x_t^i) \tag{32}$$

Con una probabilidad correspondiente a su peso

$$\Pr(x_t^{*\,i} = x_t^j) = w_t^i \tag{33}$$

El resultado es una muestra i.i.d. de la f.d.p. discreta (33).

Sirva como ejemplo la situación de la Figura 24. Se tiene un sistema con $N = 7$ partículas, representadas por círculos de tamaño proporcional a su peso. De

acuerdo con (18) se muestra un posible resultado del proceso de remuestreo, donde se pierden tres de las partículas con menor peso a cambio de replicar cuatro veces la que mayor peso tiene. Las demás partículas permanecen en el sistema.

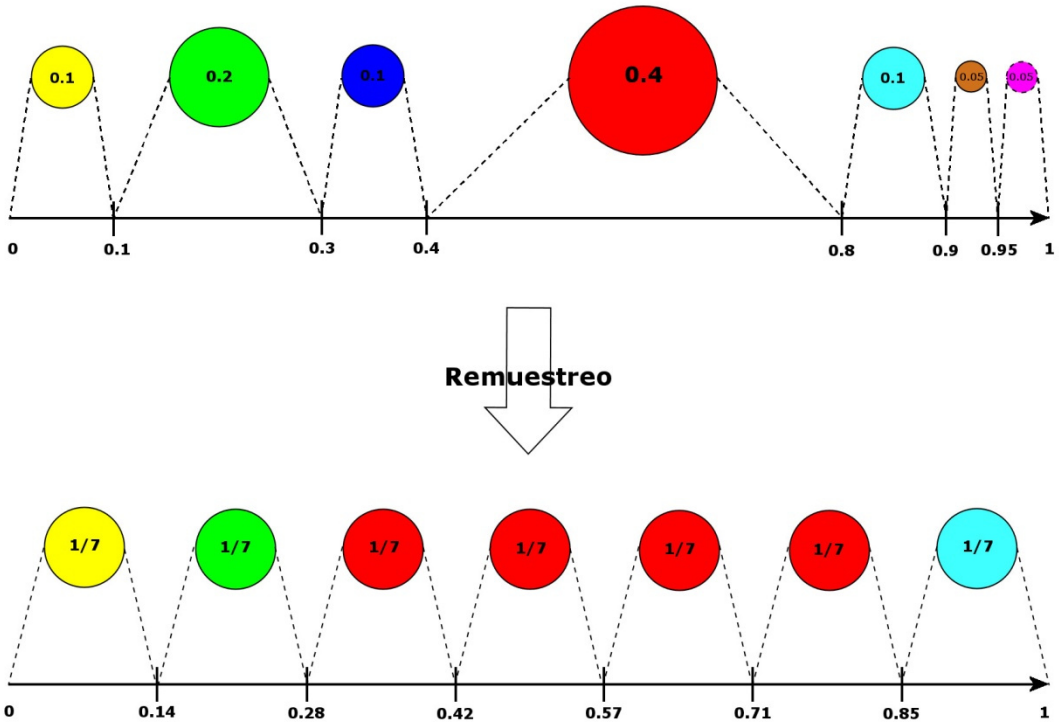


Figura 24. Ilustración del proceso de Remuestreo.

El remuestreo es un método introducido con el fin de reducir el problema de la degeneración, pero, por otro lado, la selección de partículas con pesos muy elevados hace que la muestra después de cada paso de remuestreo contenga muestras repetidas muchas veces. Esto lleva a una pérdida de diversidad que se suele denominar empobrecimiento muestral.

4. SOLUCION PROPUESTA

4.1. INTRODUCCIÓN A LA SOLUCIÓN

Una vez descrito el filtro de partículas básico, se puede presentar la solución que se utilizará en este estudio, así como las mejoras propuestas con el fin de comparar y evaluar los resultados obtenidos.

La solución implementada está basada en el algoritmo SIR (*Sequential Importance Resampling*) propuesta en [2]. Este algoritmo mejora el rendimiento del filtro de partículas básico a través de una etapa de remuestreo que previene la degeneración de los pesos (ver Sección 3.5).

La solución empleada en el problema particular del seguimiento de objetivos utiliza:

- Un vector de estados que incluye la posición del objetivo, sus dimensiones (representando al mismo como un rectángulo), su velocidad y la variación temporal de sus dimensiones.

- Descriptores de objeto basados en histograma RGB.

- La similitud de Bhattacharyya para calcular la f.d.p. de las observaciones.

- Una f.d.p. del estado (función de importancia) independiente de las observaciones.

- Un remuestreo mediante la técnica "Stochastic Universal Sampling".

- Un ruido gaussiano con desviación típica prefijada.

El filtro de partículas cuenta con multitud de parámetros que deben ser definidos adecuadamente para que el rendimiento sea óptimo, tanto en términos de calidad de la estimación como de carga computacional necesaria. En este capítulo se describirán los más importantes y la influencia que tienen en el *tracking.* También expondremos diversas modificaciones que se aplicarán al sistema básico con el objetivo de mejorar su rendimiento:

1. Probar diversas técnicas de remuestreo de los pesos, con el fin de restar importancia a partículas débiles.

2. Emplear otro espacio de color, HSV, en la creación del histograma.

3. Analizar si otras medidas de similitud pueden mejorar los resultados: comparación de histogramas frente a similitudes entre matrices de covarianza.

4. Implementar adaptatividad en la desviación típica del ruido gaussiano sobre el estado.

5. Comprobar si otros vectores de estados pueden dar lugar a una solución mejor.

4.2. SOLUCIÓN BÁSICA

Como se ha comentado anteriormente, la solución básica del filtro de partículas que se implementará para realizar el estudio es el algoritmo SIR [2], también denominado filtro *bootstrap*. Básicamente se trata del algoritmo descrito en la Sección 3.5.3, incorporando un paso de remuestreo que previene la degeneración de los pesos (ver Sección 3.5.6).

El espacio de estados para este algoritmo de *tracking* está definido a partir de vectores de la forma:

$$x = [\, x \;\; y \;\; w \;\; \rho \;\; \dot{x} \;\; \dot{y} \;\; \dot{w} \;\; \dot{\rho} \,]^T = [\, s \;\; d \,]^T \tag{34}$$

Como se muestra en la ecuación (34), el vector de estado está dividido en dos componentes. La parte estática, $s = [x\, y\, w\, p]$, especifica la posición y el tamaño del objeto seguido. La componente dinámica, $d = [\dot{x} \;\; \dot{y} \;\; \dot{w} \;\; \dot{\rho}]$, especifica las velocidades de los elementos estáticos en s.

La función de importancia empleada en este filtro es la f.d.p. a priori del estado $q(x_t|x_{t-1}^i, z_t) = p(x_t|x_{t-1}^i)$, y el remuestreo se lleva a cabo en cada instante de tiempo t.

Como consecuencia, sustituyendo la función de importancia empleada en este filtro dentro de la relación especificada en la ecuación (25), nos queda una ecuación de actualización de los pesos bastante sencilla a través de la verosimilitud de las observaciones

$$w_t^i \propto p\left(z_t|x_t^i\right) \tag{35}$$

donde no se incluyen los pesos del paso anterior, ya que los pesos después de cada iteración toman siempre el mismo valor después del remuestreo, $w_{t-1}^i = \frac{1}{N}$ para $i = 1, \dots, N$.

La ecuación de actualización de estado empleada es

$$x_t = Ax_{t-1} + v_{t-1}, \qquad A = \begin{bmatrix} I_4 & I_4\Delta t \\ 0 & I_4 \end{bmatrix} \tag{36}$$

donde A es una matriz 8 x 8, I_4 es la matriz identidad 4 x 4, Δt es el paso de tiempo (relacionado con el *frame rate* de trabajo) y v_{t-1} es un término de incertidumbre en forma de ruido aditivo Gaussiano. Esta incertidumbre está parametrizada en función de su desviación típica sobre cada componente del vector de estado:

$$\left(\sigma_x, \sigma_y, \sigma_w, \sigma_\rho, \sigma_{\dot{x}}, \sigma_{\dot{y}}, \sigma_{\dot{w}}, \sigma_{\dot{\rho}}\right) \tag{37}$$

Se empleará una descripción del objeto basada en formas rectangulares, definidas a partir de su esquina superior izquierda *(x,y)* y su ancho y alto *(w,p)* (ver Sección 2.1.1).

Por otra parte, se utilizará la similitud entre histogramas como modelo de medida. Así cada partícula modelará una potencial región de la imagen correspondiente al objeto a seguir, de forma rectangular, que será usada para extraer el histograma local correspondiente. Éste será comparado con el histograma de la estimación de referencia con el que el *tracker* fue inicializado (cuando el objetivo fue detectado y analizado). Para comparar ambos histogramas, la distancia de Bhattacharyya se suele utilizar como medida de similitud entre densidades de probabilidad

$$\sum_{i=1}^{N} \sqrt{p_i q_i} \tag{38}$$

donde N es el número de elementos de los histogramas p y q. Esta medida es 1 sólo si los histogramas corresponden completamente, y 0 si son totalmente distintos.

Finalmente, la salida del *tracker* se calcula ponderando cada partícula por su peso asociado:

$$x_t^f = \sum_{i=1}^{N} w_t^i * x_t^i \tag{39}$$

El pseudocódigo para una única iteración del filtro de partículas se presenta en el Algoritmo 2. Los pasos 3-5 y 8 representan una fase de remuestreo en la cual las partículas son remuestreadas de acuerdo a su peso. Esto previene un empobrecimiento del conjunto de partículas en el cual sólo unas pocas partículas de importancia sobreviven [46] (ver Figura 24). Los pasos 10-12 normalizan los pesos para que sumen 1.

Entrada: $\{x_{t-1}^i, w_{t-1}^i, c_{t-1}^i\}_{i=1}^{N_s}, z_t$

Salida: $\{x_t^i, w_t^i, c_t^i\}_{i=1}^{N_s}$

1 $c_t^0 = 0;$

2 ***for*** $i \in [1; N_s]$ *do*

3 $\quad draw\ r \sim \mathcal{U}[0,1];$

4 $\quad j = \min\ \{l \in \{1 \dots N_s\} | c_{t-1}^l \geq r\};$

5 $\quad \tilde{x}_{t-1}^i \coloneqq \tilde{x}_{t-1}^j;$

6 $\quad \tilde{x}_t^i \coloneqq f_t(\tilde{x}_{t-1}^i, v_{t-1});$ $\quad (de\ ecuación\ 36)$

7 $\quad w_t^i \coloneqq p(z_t | x_t^i);$ $\quad (de\ ecuación\ 35)$

8 $\quad c_t^i \coloneqq c_t^{i-1} + w_t^i;$

9 ***endfor***

10 ***for*** $i \in [1; N_s]$ *do*

11 $\quad w_t^i \coloneqq \frac{w_t^i}{c_t^N}; \quad c_t^i \coloneqq \frac{c_t^i}{c_t^N};$

12 ***endfor***

Algoritmo 2: Filtro de partículas

El filtro de partículas SIR es muy sencillo de implementar pero tiene dos debilidades básicas. La primera se refiere a la aplicación de un paso de remuestreo cada instante de tiempo, lo cual nos lleva a un empobrecimiento muestral rápido como

ya se comentó en la Sección 3.5.6. La segunda aparece en general en filtros de partículas que emplean la f.d.p. a priori $p(x_t|x_{t-1})$ como función de importancia. Si la función a priori es mucho más dispersa que la verosimilitud $p(z_t|x_t)$ el filtro puede desembocar en un problema de degeneración ya que únicamente unas pocas partículas disponen de grandes pesos.

4.3. AJUSTE DE PARÁMETROS

La solución del filtro SIR implementada requiere que algunos parámetros sean fijados manualmente, pues dependen de multitud de factores referidos tanto al vídeo (tipo de objetivo, forma, tamaño, trayectoria) como al entorno de trabajo (capacidad de cómputo, tipo de aplicación, necesidad de seguimiento en tiempo real). En esta sección se abordará este problema, describiendo los parámetros a fijar y estudiando su influencia en el algoritmo. En el siguiente capítulo se determinarán los valores reales para las secuencias de test disponibles, extrayendo diversas conclusiones que complementan a las que se ofrecen a continuación.

4.3.1. Numero de partículas

Es un elemento decisivo a la hora de implementar el algoritmo, pues establece las posibilidades de capturar al objetivo. Un número bajo dará, por lo general, peor resultado, ya que no permite muestrear la función con suficiente precisión. Sin embargo, un número muy alto de partículas no tiene por qué ofrecer los mejores resultados (ver Figura 25). De hecho, los estudios realizados demuestran que a partir de cierto umbral, la calidad del *tracking* no mejora aunque, si bien es cierto, los resultados son más estables.

(a) (b) (c)

Figura 25. Ejemplos de *tracking* con diferente número de partículas. (a) Ns = 100, (b) Ns = 500, (c) Ns = 2000. Se puede apreciar la mejora que supone emplear 500 partículas respecto a emplear 100 (el

rectángulo está centrado en el objetivo), así como el resultado similar que se consigue con 2000 partículas, a pesar de ser un valor 4 veces mayor.

4.3.2. Modo de generación de la salida del tracker

El funcionamiento del filtro de partículas se basa en crear una estimación final ponderando todas las partículas de acuerdo a sus pesos asociados (39). Sin embargo, también se podría resolver el problema escogiendo como estimación la partícula que tenga un mayor peso (40). De esta manera se podrían evitar errores por la influencia de partículas débiles y proporcionar un *tracking* más exacto. Como se verá a tenor de los resultados, este tipo de estimaciones son menos robustas, ya que dependen de una única muestra del espacio de estados.

$$x_t^f = \sum_{i=1}^{N} w_t^i * x_t^i \tag{39}$$

$$x_t^f = \max_i\left(w_t^i\right) * x_t^i \tag{40}$$

4.3.3. Desviación típica del ruido

Éste parámetro muestra una mayor dependencia con el video escogido en las pruebas: movimiento, tamaño del objetivo, trayectoria simple o compleja… En general, para vídeos con objetivos pequeños o alejados de la cámara (cambio de posición y tamaño relativos pequeño), o que se mueven despacio, se puede utilizar ruido más bajo, mientras que para objetivos cerca de la cámara (cambios más bruscos de posición y tamaño) o que se muevan rápido los valores óptimos han de ser más elevados. Si bien se ha empleado el mismo valor en la desviación típica de todas las componentes del vector de ruido, tanto de la parte estática como de la dinámica (y para posición y tamaño), también se podrían ajustar las desviaciones típicas de cada componente por separado, en función del objetivo a seguir (objeto rápido que no cambia de tamaño, objetos lentos que se acercan y alejan de la cámara…).

Lo ideal sería un ajuste dinámico de estos valores en función de parámetros como el tamaño del objeto o los cambios de velocidad en *frames* anteriores (ver Figura 26).

[68 88 36 20] [66 87 34 19]

(a)

[226 79 30 127] [230 80 41 124]

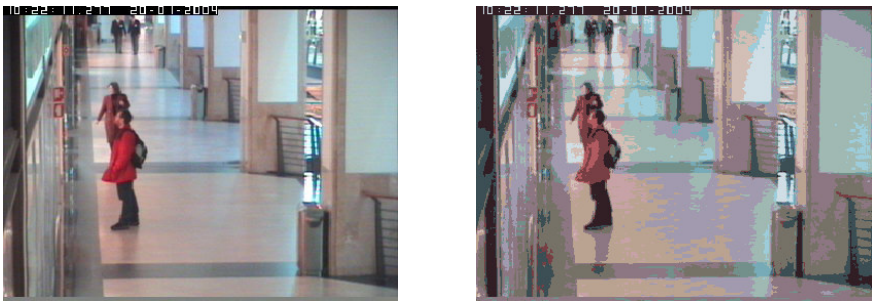
(b)

Figura 26. Diferencias en el cambio en el vector de estados de un *frame* al siguiente. (a) El cambio es [-2 -1 -2 -1]. (b) El cambio es [4 1 11 -3]. Se puede apreciar como en el segundo vídeo los cambios de un instante a otro son más bruscos, debido en este caso a que el objetivo está más cerca de la cámara. Sería recomendable ajustar estos parámetros dinámicamente para que el sistema se adaptase mejor a los cambios.

4.3.4. Número de *bins* del histograma

Este parámetro decide la resolución del histograma que se va a comparar. Si se utilizan muy pocos *bins*, toda la gama de colores se reduce a unos pocos valores, con lo que se reducen las diferencias en la imagen y es más posible que el *tracking* sea erróneo (ver Figura 27). Este parámetro ofrecerá mejores resultados cuanto más alto sea, pues permitirá diferenciar con mayor exactitud el objeto a seguir del resto de elementos de la escena, por muy parecidos que sean los colores. Sin embargo, hay que tener en cuenta la carga computacional que introduce, ya que los histogramas a calcular son de dimensión K^3.

(a)



(b)

Figura 27. Influencia de la resolución del histograma en el *tracking*. Imágenes en alta resolución (8 bits y 256 valores por canal) y en baja resolución (4 bits y 16 valores por canal). (a) Se puede apreciar la pérdida de calidad y la mayor facilidad para confundir el objetivo (hombre de blanco) con el fondo. (b) Errores de *tracking*, al confundir a las dos personas con colores similares.

4.4. EXTENSIONES

Una vez ajustados los parámetros libres a valores que den unos resultados aceptables, se extenderá la versión básica con diversas modificaciones que mejoren el resultado global del algoritmo. Las modificaciones están orientadas a optimizar diversas etapas del algoritmo, como la medida de similitud, la fijación de parámetros o el remuestreo.

4.4.1. Extensión 1: corrección de los pesos

En ciertas ocasiones las partículas se expanden por la escena más de lo necesario, y las partículas alejadas, aunque tengan pesos pequeños, pueden resultar decisivas a la hora de generar la estimación (no olvidemos que se realiza una ponderación de todas las partículas). En estos casos dichas partículas pueden desviar la atención del algoritmo y provocar errores (ver Figura 28).

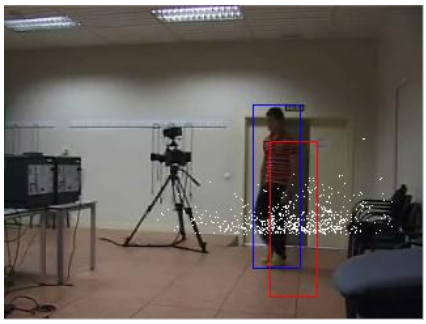


Figura 28. Importancia de los outliers. Las partículas situadas a la derecha del objetivo, aunque con un peso inferior, provocan un error en la fijación del objetivo.

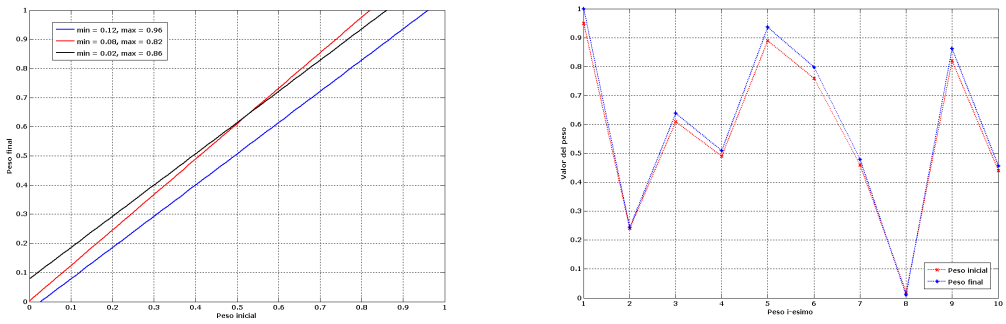
Para mejorar el rendimiento del remuestreo, se pueden emplear diversas funciones de expansión que reduzcan la importancia de las partículas con pesos pequeños, disminuyendo su probabilidad de ser elegidas en la siguiente iteración del algoritmo de muestreo por importancia y aumentando así la calidad de las muestras.

A) Función lineal

Una posibilidad es emplear una función lineal que expanda el margen dinámico, asignando los valores mínimo y máximo a 0 y 1, respectivamente

$$w' = 0.9 * \frac{w - \min(w)}{\max(w) - \min(w)} + 0.1 \qquad (41)$$

Sirva como ejemplo un conjunto de $N = 10$ partículas con valor aleatorio. Como se puede observar en la Figura 29, el cambio no es muy notable. De hecho, la expansión depende del rango dinámico de los valores: a más distancia entre los valores mínimo y máximo, menor expansión.



(a) (b)

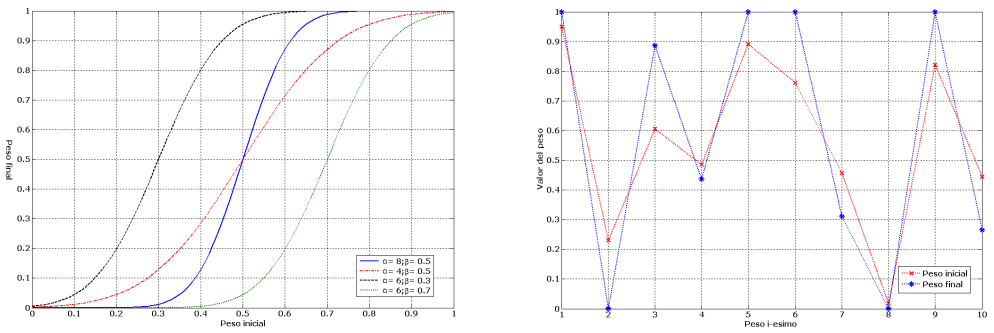
Figura 29. Función lineal aplicada al remuestreo. (a) Función lineal para varios rangos dinámicos. (b) Pesos modificados por función lineal.

B) Función sigmoide

Con el fin de mejorar la etapa de remuestreo se puede utilizar también una función no lineal, que permite un ajuste más fino que una función lineal. Emplearemos en nuestra modificación una función sigmoide (42), que mejora los pesos progresivamente a partir de un cierto umbral β, y los reduce en la misma proporción por debajo de cierto umbral. El factor de aumento/reducción viene dado por el parámetro α (ver Figura 30(a)).

$$w' = \frac{\mathrm{erf}(\alpha((w)-\beta))+1}{2} \tag{42}$$

La función sigmoide escogida tiene parámetros $\alpha = 8$, $\beta = 0.5$. Utilizaremos para mostrar un ejemplo las mismas partículas del apartado anterior (ver Figura 30(b)). Se puede observar en este caso un cambio más apreciable en el valor de las partículas. El paso a través de la función optimiza los 4 pesos de mayor valor (0.95, 0.89, 0.76, 0.82), ya que la función eleva a 1 los pesos por encima de 0.75. El peso de valor 0.61 se transforma en 0.89, mientras que al resto de pesos, de valor menor a 0.5, se les resta importancia, anulando 2 de ellos que tienen valor inferior a 0.25.



(a) (b)

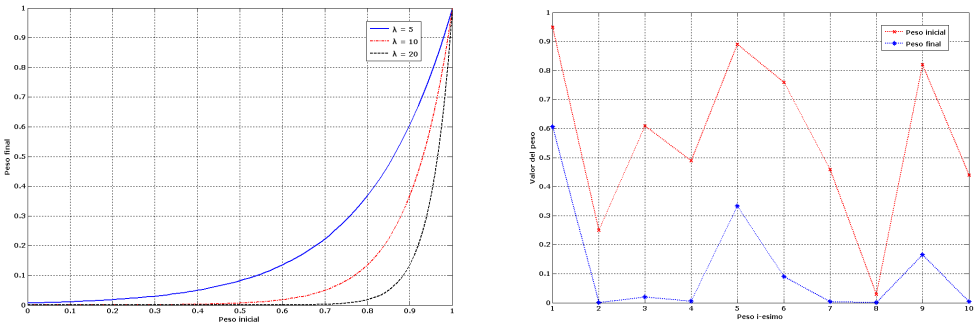
Figura 30. Función sigmoide aplicada al remuestreo. (a) Función sigmoide para varios valores de α y β. (b) Pesos modificados por función sigmoide.

C) Función exponencial

Por último, también se puede probar con una función exponencial, del tipo

$$w' = e^{-\lambda(1-w)} = e^{-\lambda}e^{\lambda w} = ke^{\lambda w} \tag{43}$$

Donde λ determina la pendiente de la función (Figura 31(a)). Como se puede observar en la Figura 31(b), todos los pesos se reducen notablemente, siendo muchos de ellos aproximadamente 0. Después de normalizar los valores, la diferencia entre los pesos buenos y los malos es mucho más grande, y por tanto, la probabilidad de repetición de las partículas débiles menor.



(a) (b)

Figura 31. Función exponencial aplicada al remuestreo. (a) Función exponencial para varios valores de λ. (b) Pesos modificados por función exponencial para $\lambda = 10$.

Con las 3 funciones descritas se mejora la calidad de las partículas, aunque conviene recordar que esta modificación introduce el problema del empobrecimiento muestral (ver Sección 3.5.6). Por tanto, hay que tener cuidado a la hora de escoger los parámetros de las funciones de corrección, para que haya diversidad suficiente a lo largo del proceso de *tracking*.

4.4.2. Extensión 2: ruido adaptativo

En los *trackers* basados en filtro de partículas, algunos de los parámetros más importantes a la hora de controlar el rendimiento son las varianzas de ruido en la ecuación (37). Considérese el caso donde las varianzas estáticas $\sigma_s = (\sigma_x, \sigma_y, \sigma_w, \sigma_\rho)$ se establecen a valores muy altos. En este caso las muestras del filtro se colocan sobre un área extensa para maximizar la posibilidad de capturar al objeto en caso de cambios erráticos de dirección o velocidad. El inconveniente de esta estrategia es que también se incrementa la probabilidad de que el filtro de partículas sea distraído por zonas espurias similares que pertenecen al fondo.

Considérese también la incertidumbre impuesta en la parte dinámica d_t del vector de estados x_t. De la ecuación (36), la ecuación de actualización para propagar una partícula del instante *t-1* al instante t es, en la componente x únicamente y después de sustituir la expresión por x_{t-1}:

$$x_t = x_{t-1} + \dot{x}_{t-2} + v_{\dot{x},t-2} + v_{x,t-1} \tag{44}$$

Se puede observar que la incertidumbre de la componente dinámica es propagada a la componente estática, amplificando el ruido de posición y tamaño. Aún peor, esta incertidumbre estimada está basada en información de la iteración en *t-2*, y no sólo de la iteración previa. En cualquier caso, el ruido en los elementos estático y dinámico no debería ser establecido a valores altos simultáneamente.

Para obtener una solución de compromiso entre la incertidumbre de las componentes estáticas y dinámicas, usamos la similitud de la estimación actual con el histograma de referencia (el peso de la partícula actual) como un indicador de la calidad del *tracking*. Denotando por w_t a la probabilidad a posteriori de la salida del *tracking* en la iteración actual t, obtenemos un valor de adaptación ζ introduciéndolo en una sigmoide:

$$\zeta(\psi_t) = \frac{\mathrm{erf}(\alpha((1-w_t)-\beta))+1}{2} \tag{45}$$

La Figura 32 muestra algunos ejemplos de la función de adaptación para diferentes valores de los parámetros α y β. El parámetro α controla la pendiente de la transición, y β ajusta la posición a la cual tiene lugar dicha transición.

En cada iteración del filtro de partículas, ajustamos las varianzas de ruido de acuerdo al valor de adaptación actual, estimada en la iteración previa:

$$\sigma_s^t = \zeta(w_t) \cdot \sigma_s, \;\; \sigma_d^t = (1 - \zeta(w_t)) \cdot \sigma_d \tag{46}$$

Esta técnica adaptativa ajusta las varianzas de forma que el ruido en observaciones de componentes estáticas nunca es amplificado debido al ruido en las componentes dinámicas.

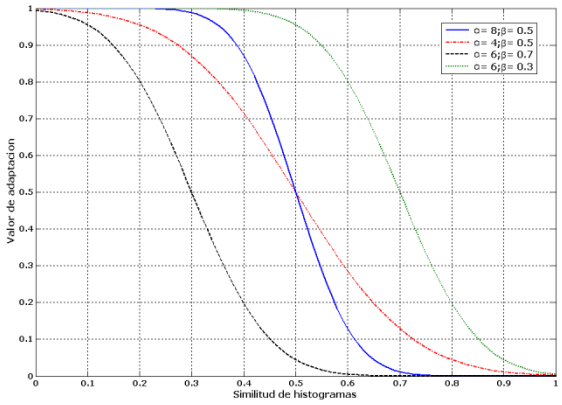


Figura 32. Función sigmoide para varios valores de α y β

4.4.3. Extensión 3: histograma HSV

El espacio de color HSV es un espacio cilíndrico, pero normalmente asociado a un cono o cono hexagonal, debido a que es un subconjunto visible del espacio original con valores válidos de RGB (ver Figura 33).

Este espacio representa mejor la forma de percibir los colores del ser humano que RGB, pudiendo definir cualquier valor de color en términos del tono, la saturación y la intensidad del mismo:

- Tonalidad (*Hue*): Se refiere a la frecuencia dominante del color dentro del espectro visible. Es la percepción de un tipo de color, es decir, es la sensación humana de acuerdo a la cual un área parece similar a otra o cuando existe un tipo de longitud de onda dominante. Se representa como un grado de ángulo cuyos valores posibles van de 0º a 360º (aunque para algunas aplicaciones se normaliza de 0 a 100%). Su valor se incrementa mientras nos movemos de forma antihoraria en el cono, con el rojo en el ángulo 0.

- Saturación (*Saturation*): Se refiere a la cantidad del color o a la "pureza" de éste. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará. Se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100%. También se puede considerar como la mezcla de un color con blanco o gris.

- Valor (*Value*): Es la intensidad de luz de un color. Dicho de otra manera, es la cantidad de blanco o de negro que posee un color. Representa la altura en dicho eje blanco-negro, teniendo unos valores posibles del 0 al 100%.

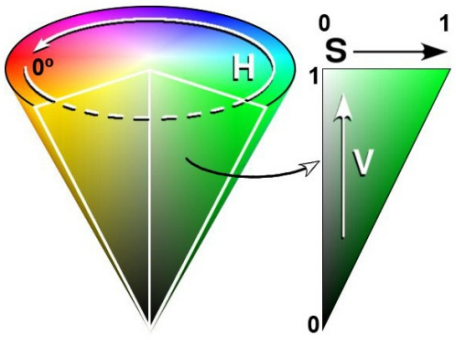


Figura 33. Representación del espacio de color HSV.

Empleando el espacio de color HSV (*Hue-Saturation-Value*) se consigue desacoplar la información cromática de los efectos de iluminación o sombras. La información de color, sin embargo, sólo es de utilidad cuando los niveles de saturación e intensidad (*S* y *V*) no son demasiado pequeños. Por eso, empleamos un histograma con las componentes *H* y *S*, que además permiten reducir la carga computacional.

4.4.4. Extensión 4: histograma múltiple

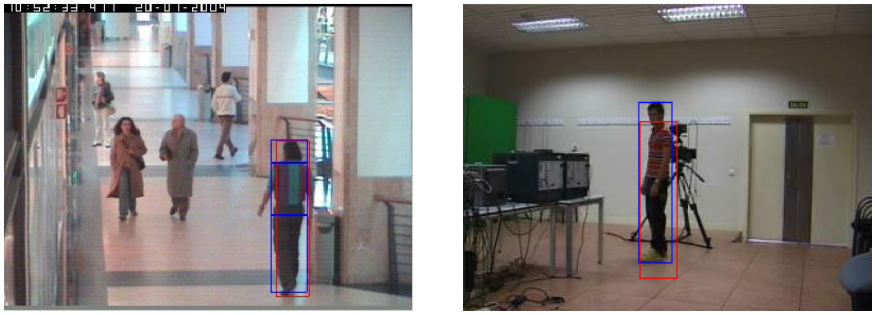
Si la región de interés contiene zonas de distintos colores, por ejemplo, la cara y la ropa de una persona, el modelo basado en histograma lo capturará. Sin embargo, toda la información relativa a la distribución espacial de estas zonas dentro de la región de interés se perderá. Mantener un seguimiento de esta información espacial podría ser beneficioso para mejorar el rendimiento del *tracker*.

Este objetivo puede ser llevado a cabo fácilmente en nuestro modelo dividiendo la región de interés en sub-regiones con modelos de color individuales. Estas regiones están rígidamente unidas de manera que el espacio de estados se mantiene sin cambios, con una única localización y tamaño por objetivo (ver Figura 34(a)). El peso final será una ponderación de los pesos asociados a las distintas regiones. En este punto se pueden establecer prioridades en función de la importancia de cada región, de su tamaño... En nuestro caso seleccionaremos una

función de ponderación equitativa, de manera que el peso final sea el producto de los pesos individuales:

$$w(i) = w_1(i) * w_2(i) * w_3(i) \quad (47)$$

Capturando esta información espacial de los colores, la localización podría ser más precisa, posicionándose mejor en el objetivo y capturando mejor su tamaño (ver Figura 34(b)). Además podría corregir errores derivados de oclusiones parciales, ya que las zonas no ocultas elevarían el peso final y evitarían posibles pérdidas del objetivo.



(a) (b)

Figura 34. Ejemplos de *tracking* empleando la técnica de histograma múltiple. (a) División del objetivo en sub-regiones independientes con histograma propio. (b) Error de *tracking* común, producido por la apariencia similar del fondo o de partes del objetivo que dan lugar a un histograma parecido, perdiendo la referencia espacial.

4.4.5. Extensión 5: matriz de covarianzas

Uno de los mayores problemas en *tracking* es la falta de un criterio de similitud potente que capture propiedades tanto estadísticas como espaciales, ya que la mayoría de aproximaciones dependen solo de la distribución del color o de modelos estructurales. También se han utilizado diferentes representaciones basadas en agregar propiedades estadísticas o de apariencia a los modelos estructurales. Como ya se ha visto, los histogramas de color son representaciones muy populares de densidad (ver Sección 2.1.2). Sin embargo, no tienen en cuenta la distribución espacial de los valores. Además, no pueden escalar a mayores dimensiones debido al crecimiento exponencial que supone. Los modelos de apariencia mapean las características de la imagen en un tamaño de ventana fijo. Como la dimensionalidad crece polinómicamente en función del número de

características y el tamaño de la ventana, existe una limitación clara en el número de características que pueden ser utilizadas.

Para superar las limitaciones de las aproximaciones existentes se ha propuesto en la sección anterior emplear múltiples histogramas que capturen subregiones distintas del objetivo de interés. De esta manera se tiene una información espacial básica, que describe la posición de cada subregión dentro de la región completa. Una solución más aproximada consiste en utilizar la representación basada en matrices de covarianza para describir los objetos [48].

Denotaremos la imagen observada como I, la cual puede ser una imagen de intensidad unidimensional, una imagen a color tridimensional, una imagen tetradimensional que combine color e infrarrojo, etc. Sea F de dimensiones $W\,x\,H\,x\,d$ (ancho x alto x nº características) la imagen de características extraída de I, la cual puede incluir color, gradientes I_x, I_{xx}, …, orientación de los bordes, valores de textura… Para una ventana rectangular $R \subset F$, sea $\{f_k\}_{k=1\ldots n}$ los vectores de características de dimensión d dentro de R. Construimos el vector de características f_k usando dos tipos de información: atributos espaciales obtenidos de las coordenadas de los píxeles, y atributos de apariencia como color, gradiente, infrarrojo, etc. Estas características se pueden asociar directamente a las coordenadas del píxel

$$f_k = [\ x\ \ y\ \ I(x,y)\ \ I_x(x,y)\ \ \ldots\ ] \tag{48}$$

Alternativamente, pueden ser enlazados a través de la relación radialmente simétrica

$$f_k^r = [\ ||(x',y')||\ \ I(x,y)\ \ I_x(x,y)\ \ \ldots\ ] \tag{49}$$

donde

$$\left|\left|x',y'\right|\right| = \sqrt{(x'^2+y'^2)},\quad (x',y') = (x-x_0, y-y_0) \tag{50}$$

son las coordenadas relativas, y (x_0,y_0) las coordenadas del centro de la ventana. Este último vector de características f_k^r proporciona invarianza frente a rotaciones del objeto.

Representamos una región rectangular R de dimensiones $M\ x\ N$ con una matriz de covarianza C_R de dimensiones $d\ x\ d$

$$C_R = \frac{1}{MN}\sum_{k=1}^{MN}(f_k - \mu_R)(f_k - \mu_R)^T \quad (51)$$

Donde μ_R es el vector de medias de las características correspondientes para los puntos de la región R. La matriz de covarianza es una matriz simétrica donde los valores de la diagonal representan la varianza de cada característica y los valores fuera de la diagonal representan sus correlaciones respectivas.

Utilizar matrices de covarianza como descriptores de región tiene bastantes ventajas. Estas matrices proponen un método natural de fundir múltiples características sin la necesidad de normalizarlas o utilizar pesos ponderados. Además, la matriz agrupa la información que se puede extraer de los histogramas así como la información derivada de los modelos de apariencia. En general, una única matriz de covarianza extraída de una región es suficiente para encontrar una región desde diferentes vistas y en diferentes posiciones. El ruido que suele afectar a muestras individuales es mayoritariamente filtrado con un filtro de media durante el proceso de cálculo. Las matrices de covarianza de cualquier región tienen el mismo tamaño, lo que permite comparar regiones sin estar restringido a un tamaño de ventana constante, proporcionando así la propiedad de invarianza frente a cambios de tamaño.

Como se ha visto anteriormente, las matrices de covarianza pueden ser invariantes frente a rotaciones. Sin embargo, si se incluye información que tenga en cuenta la orientación de los puntos en el vector de características, es posible detectar discrepancias rotacionales. También hay que destacar la invarianza frente a cambios medios, como cambios idénticos en distintos valores de color. Esta propiedad es muy ventajosa cuando los objetos tienen que ser seguidos bajo condiciones de iluminación variable.

Para obtener la región más similar al objeto dado, es necesario calcular distancias entre las matrices de covarianza correspondientes a la ventana del objeto y las regiones candidatas. Suponiendo que no haya características en vector de características exactamente idénticas, lo que significaría que las matrices de covarianza fueran definidas positivas, es posible aplicar la medida de distancia propuesta por Förstner [49]. Esta distancia utiliza la suma de los logaritmos al

cuadrado de los autovalores para calcular la diferencia entre las matrices de covarianza como

$$\rho(C_i, C_j) = \sqrt{\sum_{k=1}^{d} ln^2 \lambda_k(C_i, C_j)} \quad (52)$$

Donde $\{\lambda_k(C_i, C_j)\}$ son los autovalores generalizados de C_i y C_j . La distancia ρ satisface los axiomas métricos, positividad, simetría y desigualdad triangular, para matrices simétricas definidas positivas.

4.5. CONCLUSIONES

En este capítulo se ha presentado el modelo del sistema escogido para estudio en el presente proyecto, así como su ámbito de aplicación. En particular, se ha descrito brevemente el algoritmo del filtro de partículas implementado. También se han analizado los parámetros que se ajustarán a partir de diversas pruebas sobre un conjunto de secuencias de entrenamiento, así como las distintas técnicas utilizadas en las extensiones propuestas. Estas técnicas se han escogido en base a los avances expuestos anteriormente (ver Sección 2) y tienen como objetivo mejorar el rendimiento final del algoritmo.

5. PRUEBAS Y RESULTADOS

En el presente capítulo se describirán exhaustivamente las pruebas realizadas para evaluar, analizar y comparar los algoritmos implementados. En particular, se describirá el entorno de trabajo, condicionante de los resultados obtenidos, así como las medidas utilizadas para evaluar el rendimiento. Se analizará la influencia de los parámetros libres y se comparará la versión básica con las distintas mejoras desarrolladas, mostrando los resultados más relevantes. Finalmente, se extraerán conclusiones de los resultados obtenidos, y se detallarán las limitaciones de la aplicación. Posibles soluciones y ampliaciones del trabajo realizado se describirán en el capítulo siguiente.

5.1. ENTORNO DE TRABAJO

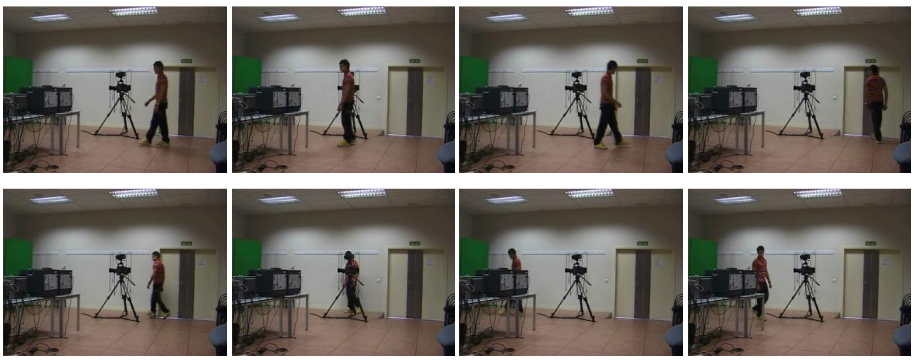
El ordenador utilizado para las pruebas es un Intel Core2 Duo E6550 @ 2.33 GHz, con 1.98 GB de memoria RAM. El sistema operativo es Microsoft Windows XP Professional en su versión SP3. La aplicación se ha diseñado íntegramente en Matlab, y el ordenador de pruebas cuenta con la versión R2007b (7.5.0.342).

Se han empleado diversos vídeos de prueba para probar la aplicación. La tabla 2 muestra un resumen de las 7 secuencias usadas en los experimentos. Han sido divididas en dos grupos: las 3 primeras secuencias son relativamente cortas y simples, y han sido utilizadas para calibrar el algoritmo y estimar los parámetros base. El segundo grupo de vídeos son más largos y contienen ejemplos de situaciones que se podrían clasificar como dificultosas para un algoritmo de *tracking*, como cambios de iluminación, oclusiones parciales, movimiento no lineal, fondos confusos y objetivos múltiples (ver Figura 35).

Las secuencias 2, 3, 6 y 7 han sido extraídas de la base de datos *Caviar*, especialmente diseñada para algoritmos de seguimiento de objetos [50]. El resto han sido creadas en la ejecución del proyecto.

Secuencia	Nombre	Duración	Multiobjetivo	Cámara	Oclusiones	Movimiento
1	Estudio 1	220	No	Estática	No	Lateral, parada y cambio dirección
2	WalkByShop1cor	130	Sí	Estática	No	Aproximación
3	WalkByShop1cor	80	Sí	Estática	No	Alejamiento
4	Estudio 1	665	No	Estática	Parciales	Complejo
5	Estudio 2	655	No	Dinámica	Parciales	Complejo
6	WalkByShop1cor	170	Sí	Estática	Parciales	Complejo
7	WalkByShop1cor	170	Sí	Estática	Parciales	Complejo*

Tabla 2. Secuencias de test empleadas en los experimentos. * Se diferencia de la secuencia 6 en que sólo se realiza el *tracking* de la camiseta del objetivo, y no del objetivo entero.



(a)

(b)

(c)

Figura 35. Secuencias utilizadas en las pruebas. (a) Estudio 1. (b) Estudio 2. (c) WalkByShop1cor.

5.2. MEDIDAS DE CALIDAD UTILIZADAS EN LA EVALUACIÓN

Las medidas utilizadas en la evaluación de los distintos algoritmos tienen por objetivo poder compararlos a todos los niveles, no sólo en términos de calidad del seguimiento, sino también de fiabilidad, estabilidad o tiempo de cálculo. Como se comentó al principio del estudio, las aplicaciones del *tracking* son muy diversas, y no siempre es mejor el algoritmo que haga un seguimiento más exacto. En la mayoría de las situaciones se hace necesario buscar una solución de compromiso entre las variables mencionadas.

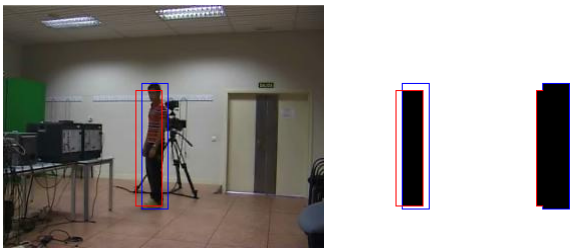
Dada la naturaleza estocástica del filtro de partículas, todos los resultados mostrados en el estudio han sido promediados sobre 10 repeticiones del experimento en las mismas condiciones. Además, con el fin de obtener valores más generales, se ejecuta el algoritmo para las 4 secuencias descritas en el apartado anterior. El valor final es una media de las 40 pruebas.

5.2.1. Calidad de la estimación

Para evaluar la precisión del algoritmo de *tracking* se utilizan dos medidas de calidad. Por un lado, se emplea la medida propuesta por Phillips y Chhabra para la evaluación de sistemas de reconocimiento de imágenes [51]. Denotando como E_g^t y E_a^t el rectángulo de *ground truth* y el estimado por el algoritmo para el *frame* t, respectivamente, la medida de rendimiento viene dada por:

$$Q\left(E_g^t, E_a^t\right) = \frac{\left|E_g^t \cap E_a^t\right|}{\left|E_g^t \cup E_a^t\right|} \tag{53}$$

Esta medida es 0 sólo en los *frames* donde los rectángulos del *tracker* y de *ground truth* son completamente disjuntos, y 1 solo cuando coinciden exactamente (ver Figura 36). Una media de la medida de rendimiento, $\bar{Q}$, es obtenida promediando $Q\left(E_g^t, E_a^t\right)$ a lo largo de todos los *frames* de la secuencia.



(a)

Q=0.9 Q=0.76 Q=0.45 Q=0.16

(b)

Figura 36. Medida de calidad *Q1* propuesta por Phillips y Chhabra [51]. (a) Regiones de la imagen que intervienen en la medida: área compartida por la estimación y el *ground truth* (numerador), y área conjunta (denominador). (b) Estimaciones y medidas asociadas para varios *frames* de una secuencia.

Por otro lado, se utiliza una modificación del *ratio de detección* propuesto en [48]. Este ratio evalúa el número de *frames* en los que la localización del objeto es estimada de forma precisa respecto al número total de *frames* de la secuencia. Consideramos una localización estimada como precisa si la posición final se sitúa dentro de un cuadrado centrado en el rectángulo de *ground truth* y cuyo lado es el 80% del lado menor de éste (ver Figura 37). La medida se promedia para toda la secuencia, siendo 1 sólo si el objeto es estimado de manera precisa en todos los *frames*.

Figura 37. Medida de calidad *Q2*, ejemplos de localización estimada precisa. El rectángulo azul representa el *ground truth* del objeto a seguir, mientras que el cuadrado rojo representa la región en la cual la salida del *tracker* es clasificada como precisa y obtiene un valor 1 en el ratio de detección; fuera de esta área el valor obtenido es 0.

En adelante denominaremos a estas medidas de calidad *Q1* o simplemente calidad, y *Q2* o precisión, respectivamente.

5.2.2. Desviación típica media de la secuencia (σ_Q)

Conviene destacar que los filtros de partículas se basan en la simulación de un proceso estocástico, y por tanto los resultados de la simulación pueden variar. En consecuencia, incluso cuando el rendimiento medio puede parecer razonablemente bueno, es posible que ocasionalmente el rendimiento esté siendo extremadamente malo (así como extremadamente bueno). La desviación típica mide la variabilidad de los resultados entre distintas medidas. A menor desviación típica, los resultados de distintas iteraciones serán más parecidos entre sí, y por tanto el algoritmo será más estable (ver Figura 38). Mediremos este parámetro a partir de la calidad *Q1* en cada *frame*, promediando para todos los *frames*. Se podría decir que es una *desviación típica temporal*, ya que mide cuánto varía el resultado entre las distintas repeticiones para un mismo *frame*.

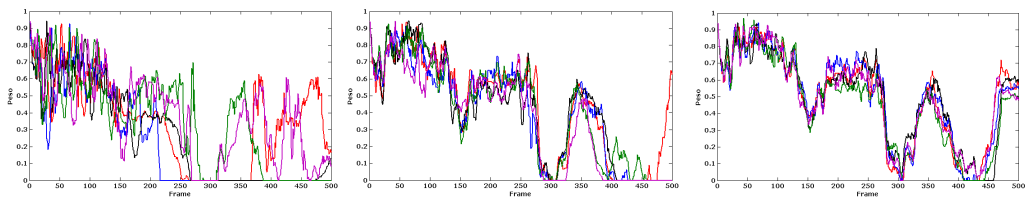


Figura 38. 5 iteraciones del algoritmo para distintos valores de N_s: 100, 500, 2000. Se puede observar que conforme aumenta el número de partículas los experimentos son más estables. σ_Q mide cuánto varía el resultado entre las distintas repeticiones para un mismo *frame*.

5.2.3. Tiempo medio de cálculo por *frame* de la secuencia

Esta medida calcula el tiempo que tarda el algoritmo en ejecutar todas las operaciones requeridas para el cálculo de la estimación de un *frame* de la secuencia. De esta manera podemos comparar los algoritmos en términos de carga computacional, seleccionando aquellos que, para un mismo rendimiento, tarden menos en su ejecución. La medida se da en *segundos/frame*.

5.3. PRUEBAS REALIZADAS

5.3.1. Establecimiento de los parámetros óptimos del algoritmo básico

En primer lugar, se buscarán unos valores adecuados para los parámetros libres del algoritmo. Los parámetros a ajustar son:

- Cálculo del estado final a partir de la información del conjunto de partículas
- Número de partículas
- Número de *bins* del histograma
- Desviación típica del ruido

Para establecer los valores óptimos de los distintos parámetros, seleccionamos unos valores de referencia para usar en las simulaciones, variando el correspondiente a la prueba únicamente. Los parámetros básicos son:

- N_s = 750 partículas
- $K = 16\ bins$
- Desviación típica del ruido: σ = [0.5 0.5 0.125 0.125 0.25 0.25 0.0002 0.0002]
- Modo de generación de la salida del *tracker*: ponderación

a) Partícula media vs Partícula máxima

Comparamos el caso general del filtro de partículas, en el que la salida del *tracker* se forma a partir de las partículas individuales ponderadas, con el caso especial en el que se elige la mejor partícula. Los resultados obtenidos son similares, si bien la ponderación que aportan las partículas permite recuperar el objetivo en situaciones en las que el algoritmo pierde la pista (ver Figura 39).

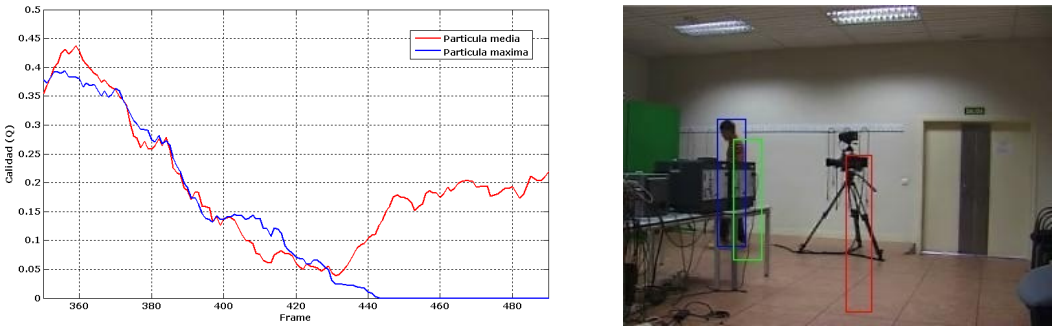


Figura 39. Comparación partícula media y partícula máxima. (a) Gráfica de calidad $\overline{Q1}$. (b) *Frame* ilustrativo de la pérdida del objetivo por parte del método de partícula máxima (rectángulo rojo) (secuencia 1, *frame* 480).

b) Número óptimo de partículas.

Seleccionamos distintos valores de N_s: 100, 250, 500, 750, 1000 y 2000 partículas.

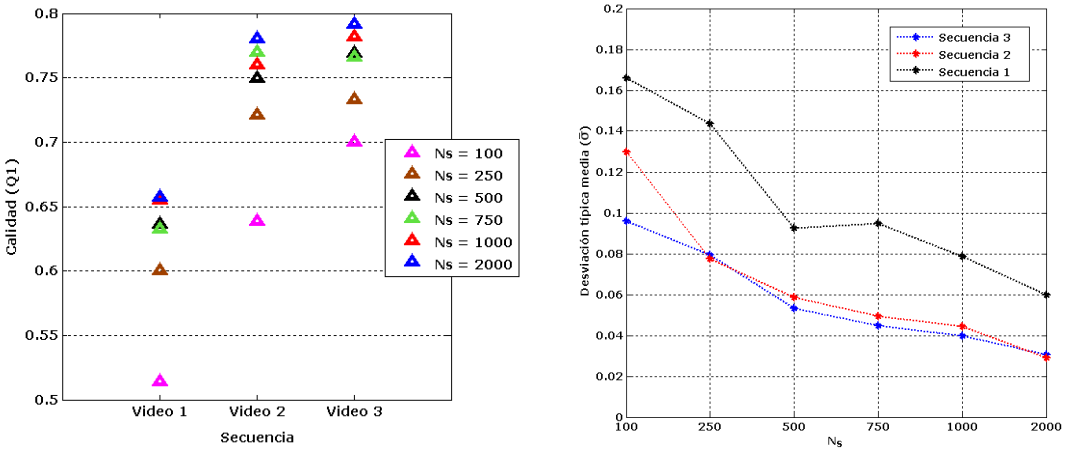
Los resultados demuestran que a partir de un cierto umbral, 750, un incremento en el número de partículas no produce un incremento significativo de la calidad. Sin embargo, disminuye la desviación típica de los experimentos (ver Figura 40), lo que incrementa la estabilidad del algoritmo (ver Sección 5.2.2.). También se observa una gran mejoría en el valor de $\overline{Q2}$, lo que implica una mayor precisión en el seguimiento.

Para 1000 y 2000 partículas el tiempo de cálculo se incrementa considerablemente respecto al caso de 750, si bien la desviación típica no disminuye en la misma proporción. Por tanto los valores óptimos son 500 y 750, en función del compromiso entre calidad/estabilidad y tiempo de cálculo.

$\overline{Q1}$	Sec. 1	Sec. 2	Sec. 3
Ns=100	0.51	0.61	0.7
Ns=250	0.60	0.72	0.73
Ns=500	0.64	0.75	0.77
Ns=750	0.63	0.77	0.77
Ns=1000	0.66	0.76	0.78
Ns=2000	0.66	0.78	0.79

$\overline{Q2}$	Sec. 1	Sec. 2	Sec. 3
Ns=100	0.425	0.64	0.7
Ns=250	0.63	0.86	0.89
Ns=500	0.78	0.95	0.97
Ns=750	0.77	0.98	0.95
Ns=1000	0.88	0.98	0.94
Ns=2000	0.92	1	0.99

Tabla 3. Tablas comparativas de $\overline{Q1}$ y $\overline{Q2}$ para distintos valores de N_s.



(a) (b)

Figura 40. Comparativa de N_s para las distintas secuencias. (a) Gráfica comparativa de $\overline{Q1}$ (b) Evolución de la desviación típica en función de N_s.

C) Número óptimo de *bins* del histograma

Se realizará una barrido de valores de K: 8, 16, 32 y 64. Los resultados obtenidos reflejan que conforme aumenta el grado de resolución aumenta también la calidad y disminuye la varianza de los experimentos. Sin embargo, la mejora no es significativa en relación al incremento en el tiempo de cálculo. De nuevo es necesario buscar un compromiso, aunque en este caso es más evidente que la solución óptima es K = 16. Hay que destacar que para 64 *bins* el histograma es una matriz 64 x 64 x 64, aumentando el tiempo de cómputo de manera excesiva, y haciendo inviable este valor de cara al establecimiento de los parámetros óptimos.

El resto de resultados se muestran en la Tabla 4.

Nº Sec	Sec.1				Sec. 2				Sec. 3			
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
K = 8	0.63	0.69	0.09	1.28	0.74	0.96	0.07	0.95	0.77	0.9	0.05	1.36
K = 16	0.63	0.77	0.09	1.34	0.77	0.98	0.05	1.07	0.77	0.95	0.04	1.43
K = 32	0.71	0.98	0.07	2.33	0.77	1	0.04	1.97	0.78	0.95	0.04	2.38

Tabla 4. Comparativa de las medidas obtenidas para distintos valores de K.

D) Desviación típica del ruido

Escogemos el doble y la mitad del valor de referencia, σ = [1 1 0.25 0.25 0.5 0.5 0.0005 0.0005] y σ = [0.25 0.25 0.062 0.062 0.125 0.125 0.0001 0.0001], respectivamente.

Un ruido alto permite al algoritmo abarcar más región en la imagen, posibilitando la detección de objetos con movimientos bruscos o muy rápidos. Sin embargo, si el movimiento es suave y regular, esta cualidad es innecesaria, y al introducir más variabilidad a los resultados se corre el riesgo de perder estabilidad y mermar el rendimiento. Además, en esta prueba hemos aumentado por igual la desviación típica de la posición y del tamaño, por lo que ahora la región de interés es más vulnerable a los cambios (ver Figura 41).

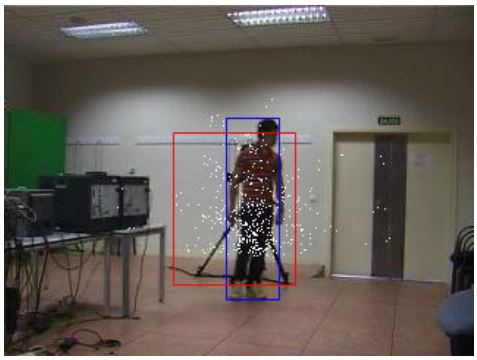


Figura 41. Consecuencias de introducir ruido elevado. Las partículas se expanden excesivamente por la escena, situándose sólo unas pocas de ellas en la posición del objetivo, mientras que el tamaño se ve alterado y a veces puede crecer (o decrecer) de manera progresiva e incontrolada.

Un ruido muy bajo puede ser adecuado para secuencias en las que el objeto es pequeño o se mueve despacio, como personas alejadas de la cámara. Sin

embargo, para el caso general, presenta un problema de adaptación: el algoritmo no puede seguir al objetivo porque los cambios, ya sea de posición o tamaño, son demasiado bruscos, empeorando el rendimiento por el período de tiempo que transcurre hasta que consigue adaptar los valores (ver Figura 42).

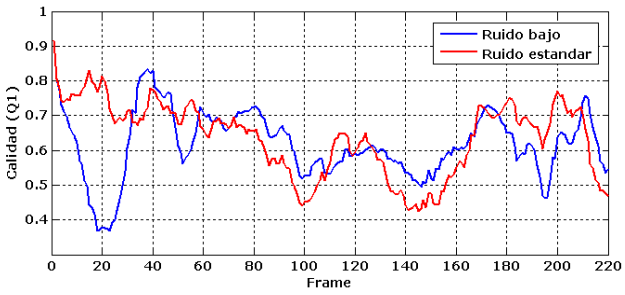


Figura 42. Resultados del algoritmo para valores de ruido bajos.. La variabilidad es tan pequeña que no permite al algoritmo adaptarse con rapidez a los cambios en la escena (*frames* iniciales y finales).

Con los resultados anteriores se tiene una ligera idea de la influencia de cada parámetro en los estadísticos medidos. Se ha podido apreciar como a medida que aumenta el número de partículas y la resolución del histograma, aumenta la calidad y el tiempo de cálculo, y disminuye la desviación típica. Por otra parte, ajustar los parámetros de ruido es una tarea compleja, ya que las consecuencias de establecer un valor u otro no son tan evidentes. Para las varianzas de posición queda claro que un mayor ruido permite abarcar cambios más bruscos, a pesar de ser más sensible para movimientos suaves, pero para las varianzas de tamaño es más complicado encontrar unos valores que permitan al *bounding box* ajustarse rápidamente a los cambios.

Finalmente seleccionamos como valores óptimos los propuestos como estándar, N_s=750, K=16 y σ = [0.5 0.5 0.125 0.125 0.25 0.25 0.0002 0.0002], pues alcanzan un compromiso aceptable entre calidad, estabilidad y tiempo de cálculo. Estos valores se emplearán para el resto de pruebas.

5.3.2. Algoritmo básico

Una vez definidos los parámetros libres del algoritmo, se puede evaluar su rendimiento en secuencias más complejas. Emplearemos de aquí en adelante las secuencias 4,5, 6 y 7.

Las pruebas arrojan unos valores muy positivos para las secuencias 4,6 y 7, con una calidad media ($\overline{Q1}$) en torno al 61 % y una precisión ($\overline{Q2}$) del 77 %. Profundizando un poco más, se pueden destacar varias apreciaciones:

- La evolución de los parámetros es irregular en función de cuál sea analizado. En general, los parámetros de posición *(x,y)* así como su vector velocidad asociado *(x,y)* son estimados correctamente, con pequeñas desviaciones puntuales. Sin embargo, los parámetros de tamaño *(w,ρ)* y su vector velocidad *(w,ρ)* son pobremente estimados, produciéndose cambios muy leves. Esto es debido a que el ruido en estos parámetros es muy pequeño, ya que un ruido más alto desestabiliza al algoritmo y el rendimiento empeora (ver apartado anterior). Sin embargo, no se trata de una pérdida grave ya que, en general, los cambios de tamaño en secuencias de vídeo no son muy bruscos, y además son menos importantes que los cambios de posición del objetivo. De esta manera nuestro algoritmo, de manera general, estimará bien el movimiento de los objetivos y su posición, pero le asignará un tamaño similar al inicial durante toda la secuencia. En la Figura 43 se muestran como ejemplo los resultados de la secuencia 6.

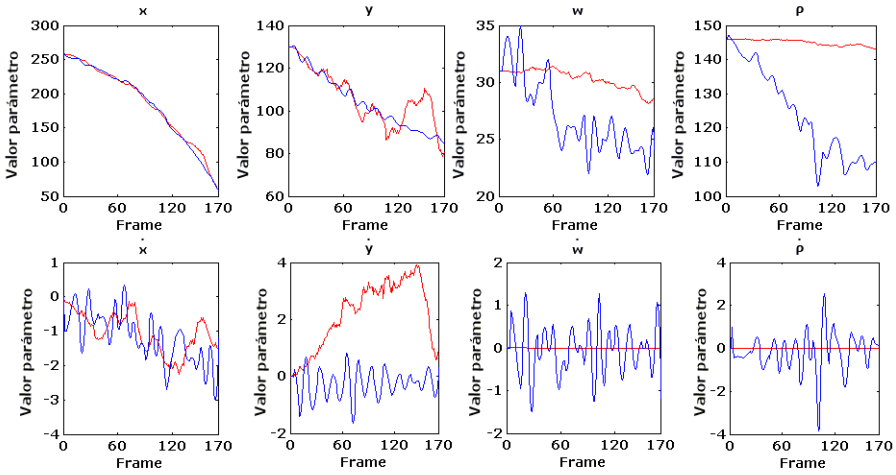


Figura 43. Gráfica de estimación de los parámetros del objetivo (ground-truth en azul y salida del algoritmo en rojo) para la Secuencia 6. Se aprecia una adaptación correcta en los parámetros de posición, no siendo así en los parámetros de tamaño. La tónica se repite en todas las secuencias analizadas.

- La mejora de la secuencia 6 (el objetivo es la persona completa) a la secuencia 7 (el objetivo es únicamente la camisa) es más que notable: $\overline{Q1}$ mejora un 7% y la precisión alcanza el 100%, lo que supone una mejora de

$\overline{Q2}$ del 37%. Esto es debido a dos razones. En primer lugar, al ser un histograma más limitado (menos componentes cromáticas), la similitud no es tan ambigua y distingue mejor al objetivo del fondo. Por otra parte, también influye el hecho de que los colores de la camisa sean llamativos y diferentes al resto de la escena. Finalmente, al ser de un tamaño más reducido y acoplarse mejor a la forma rectangular, se cuela menos componente del fondo y el algoritmo se pierde menos, mejorando en consecuencia la precisión (ver Figura 44). En este punto cabe destacar que el menor tamaño del histograma influye en el tiempo de cálculo, reduciéndose de 1.5 a 0.9 segundos.

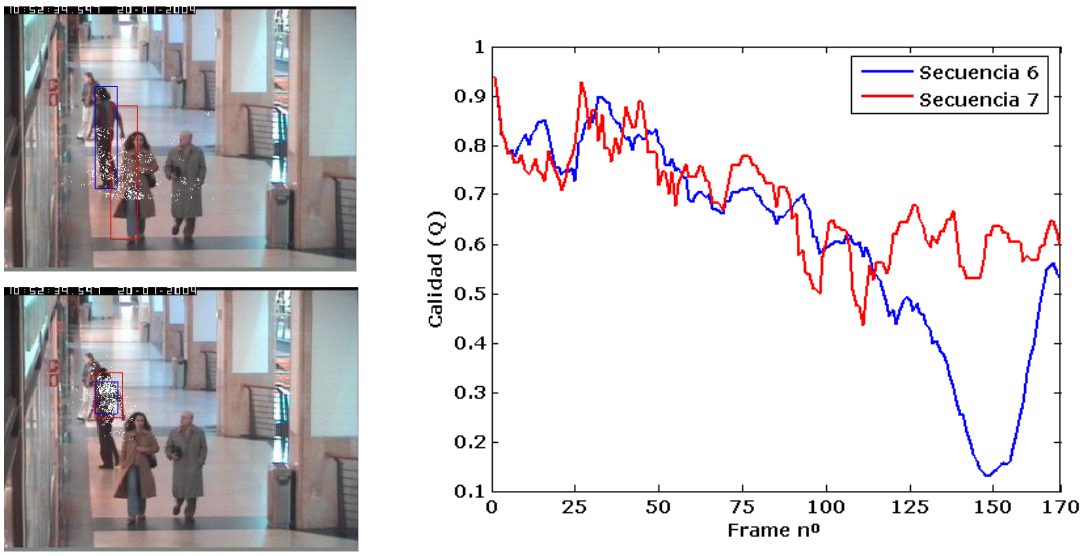


Figura 44. Diferencia entre utilizar objetivos completos o zonas características. Se reduce el nº de colores del histograma y el tiempo de cálculo. En este caso detectar únicamente la camiseta permite mejorar la estimación en la oclusión de los *frames* finales.

La secuencia 5 no comparte los buenos resultados mostrados anteriormente. De hecho, en las pruebas realizadas el rendimiento cae en picado y la calidad ($\overline{Q1}$) se sitúa en un 25%. El algoritmo no es capaz de resolver el movimiento de cámara durante una oclusión, lo que provoca una pérdida del objetivo (ver Figura 45). Como no consigue recuperarse a tiempo (las partículas siguen el rastro pero la velocidad no es suficiente) se pierde el objetivo para el resto de la secuencia.

Figura 45. Funcionamiento del algoritmo básico, secuencia 5. El algoritmo pierde la pista del objetivo para el resto de secuencia, debido a la oclusión en los *frames* 671-846 durante el movimiento de cámara.

A pesar de fallar en dicha secuencia, la versión básica en general ofrece buenos resultados y cubre las expectativas del proyecto, ya que la posición es estimada correctamente y resuelve con razonable éxito situaciones como cambios de dirección, aceleraciones y paradas, cambios de iluminación y oclusiones casi totales durante largos períodos (ver Tabla 5). Se verá en los siguientes apartados si las distintas extensiones planteadas consiguen mejorar el rendimiento y subsanar el error comentado.

Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Secuencia 4	0.54	0.69	0.09	1.57
Secuencia 5	0.26	0.24	0.14	1.37
Secuencia 6	0.61	0.63	0.06	1.48
Secuencia 7	**0.67**	**1**	**0.05**	**0.95**

Tabla 5. Resultados de la versión básica para las 4 secuencias de prueba. Se observan los mejores resultados en la secuencia 7, con un 100% de precisión.

5.3.3. Extensión 1: corrección de los pesos

La primera extensión busca optimizar la calidad de los pesos reduciendo la influencia de los *outliers*, para lo que se presentaron tres funciones de corrección a las que se sometían los pesos antes de generar la partícula final.

La consecuencia de aplicar estas funciones se observa en la Figura 46. No se aprecia una gran diferencia entre las estimaciones (rectángulos rojos), pero sí que es evidente la reducción de muestras alejadas y la concentración de las resultantes en torno al objetivo. Al disminuir el peso de estos *outliers* disminuye su probabilidad de propagarse en *frames* posteriores, mejorando la calidad de la

estimación en cuanto a precisión (y consecuentemente menor desviación en los resultados) y robustez.

Figura 46. Comparación entre la versión básica y la extensión 1. Imágenes extraídas de las secuencias 4 y 6. Se puede observar en ambas situaciones como dicha extensión reduce los *outliers* y concentra las partículas en el objetivo.

Los resultados demuestran que las 3 correcciones mejoran ligeramente el algoritmo básico. Con las modificaciones realizadas el algoritmo es más robusto frente a pérdidas (ver Figura 49), y la precisión ($\overline{Q2}$) se incrementa (ver Tabla 6). Además, la carga computacional no se resiente. A pesar de todo, no se consigue resolver el problema de la oclusión en la secuencia 5.

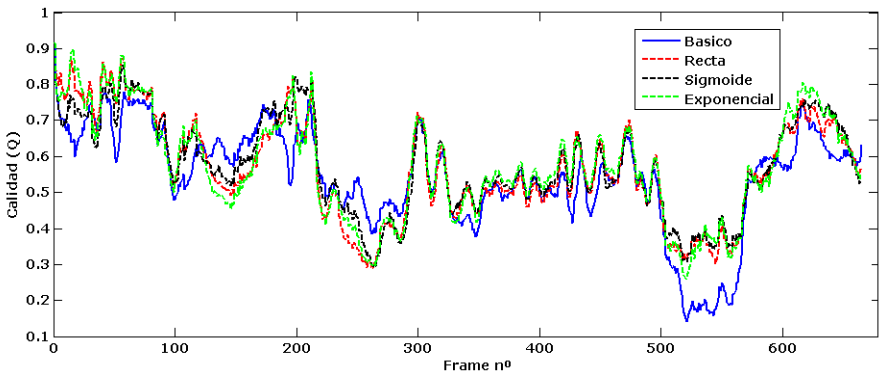


Figura 47. Gráfica comparativa de Q entre la versión básica y las distintas implementaciones de la extensión 1. Secuencia 4

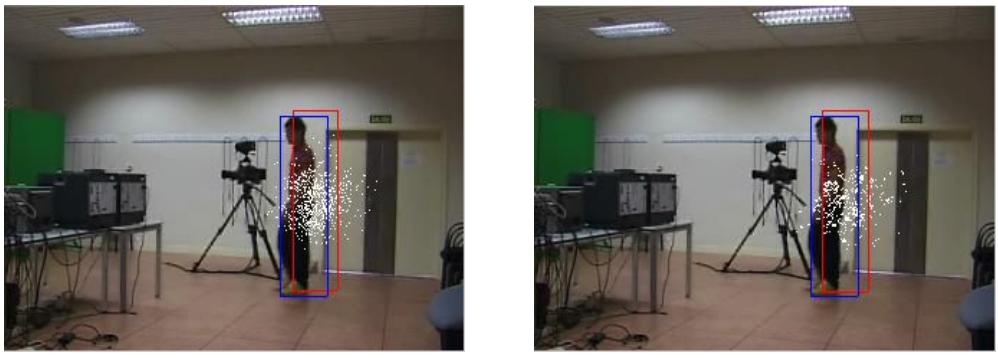
Nº Sec	Sec. 4				Sec. 6			
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Básico	0.54	0.69	0.09	1.57	0.61	0.63	0.06	1.48
Recta	0.57	0.75	**0.04**	1.49	0.65	0.65	0.04	1.59
Sigmoide	**0.58**	**0.76**	0.06	1.44	0.63	**0.67**	**0.04**	1.51
Exponencial	0.57	0.76	0.05	**1.40**	**0.65**	0.66	0.04	**1.46**

Tabla 6. Comparativa de las diferentes funciones utilizadas en la extensión 1. Secuencias 4 y 6.

5.3.4. Extensión 2: ruido adaptativo

Como se explicó en 4.2, el vector de estados tiene dos componentes, una estática y una dinámica. En ambas se introduce un ruido que permite aleatorizar la muestra. Sin embargo, el efecto en cada componente es muy distinto.

El ruido en la parte estática del vector de estados hace que las partículas se separen unas de otras y permite abarcar una región mayor. A las partículas supervivientes les introduce un desplazamiento aleatorio, generándose una nube de partículas (ver Figura 48 (a)). Por otra parte, el ruido dinámico tiene en cuenta únicamente la velocidad a la que cambian las componentes estáticas, por lo que traslada menos las partículas respecto al estado anterior. Las partículas que se generan a partir de las supervivientes se sitúan muy cerca de la posición de éstas en el instante anterior, y por ello aparecen agrupadas (ver Figura 48 (b)).



(a) (b)

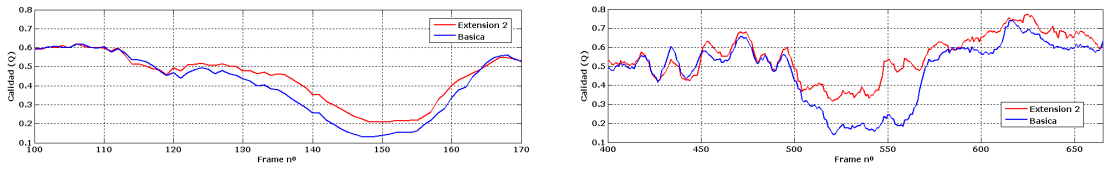
Figura 48. Influencia del ruido en las componentes (a) estática y (b) dinámica.

La mejora propuesta adapta el tipo de ruido introducido en función de los resultados que se van obteniendo. Es fácil observar que el ruido estático es de mayor utilidad cuando el objetivo se ha perdido o está a punto de perderse, pues abarca una mayor región y permite recuperarlo. El ruido dinámico será más útil cuando el objetivo esté fijado correctamente, pues las partículas se adaptan a su velocidad y no experimentan cambios tan bruscos.

En nuestro experimento empleamos la función sigmoide vista en 4.4.3 con parámetros $\alpha = 20$ y $\beta = 0.7$. De esta manera, mientras la similitud de histogramas no caiga por debajo del 30% sólo se empleará ruido dinámico. A partir de este umbral se emplea únicamente ruido estático para alcanzar el objetivo antes de perderlo definitivamente o ajustar mejor su estado.

Los resultados demuestran que esta técnica es eficaz (ver Tabla 7), mejorando la precisión cuando el objetivo se sigue correctamente, y recuperándolo rápidamente cuando la similitud de histogramas decae (ver Figura 49).

(a)



(b)

Figura 49. Resultados de la extensión 2. (a) Imágenes en las que se puede apreciar la mejora inducida por la adaptatividad del ruido, tanto en momentos buenos mejorando la situación de las partículas, como en los malos adaptándose rápidamente a los cambios. (b) Gráficas en las que se observa esta adaptación de los parámetros.

Nº Sec	Sec. 4				Sec. 6			
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Básico	0.54	**0.69**	0.09	1.57	0.61	0.63	0.06	**1.48**
Ext. 2	**0.57**	0.67	**0.09**	**1.39**	**0.61**	**0.73**	**0.05**	1.5

Tabla 7. Comparativa de la extensión 2 y la versión básica.

5.3.5. Extensión 3: histograma HSV

El objetivo de esta extensión es mejorar la detección del objetivo empleando un espacio de color que aporte más información que el espacio RGB. Para ello se optó por HSV, un espacio de color capaz de separar la información cromática de la de iluminación y sombras. Por otra parte, reducimos el espacio a las componentes H y S, ya que la intensidad (Value o V) sólo era importante para ciertos valores de las otras dos componentes.

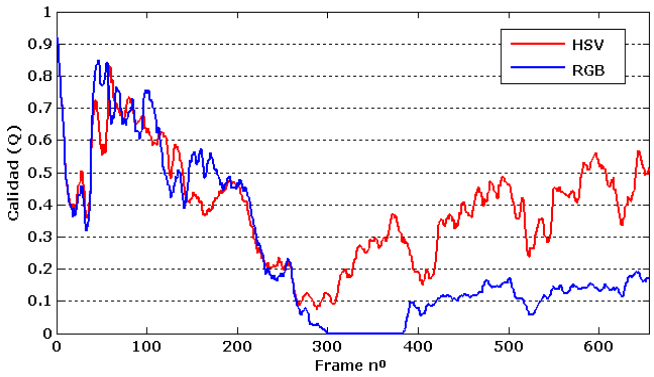
El rendimiento de la extensión es similar al de la versión básica, si bien podemos destacar varios hechos:

- Este espacio de color, a pesar de la teoría, no distingue bien al objetivo del fondo, lo que provoca una expansión de las partículas por la escena. Este alejamiento respecto al objetivo provoca inestabilidad y deriva en una mayor facilidad de pérdida que con histogramas RGB, como en la secuencia 6 (ver Figura 50). De hecho, la desviación típica de los valores de Q1 crece hasta un 5%.

Figura 50. Inestabilidad de la extensión 3: histograma HSV. En la primera imagen se puede observar la expansión de las partículas y las distracciones que acarrea en la secuencia 4. En las otras dos imágenes se representa la inestabilidad y la posterior pérdida del objetivo en la secuencia 6.

- Gracias a esta difusión de las partículas el algoritmo reacciona muy bien frente a cambios de sentido. De hecho, maneja de manera efectiva la oclusión de la secuencia 4 y, por primera vez, la problemática oclusión de la secuencia 5 (ver Figura 51).

(a)



(b)

Figura 51. Resolución de la oclusión en la secuencia 5. (a) Tras unos cuantos *frames* atascado en la zona izquierda de la escena, el algoritmo reacciona y consigue recuperar al objetivo. (b) Gráfica descriptiva de la situación anterior en comparación con la versión básica que emplea histograma RGB.

- Por último, la carga computacional que introduce transformar cada región de cada partícula al espacio HSV es excesiva, duplicándose el tiempo de procesamiento de cada *frame* (ver Tabla 8).

En resumen, se trata de un algoritmo que resuelve de manera aceptable el principal problema que planteaba la versión básica, pero que ofrece peores resultados en situaciones más sencillas (y más comunes en vídeos reales) y una inestabilidad poco apropiada para introducirlo en la versión final (ver Figura 55).

	Versión básica				Extensión 3: HSV			
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Secuencia 4	0.54	**0.69**	**0.09**	**1.57**	**0.58**	0.56	0.14	2.68
Secuencia 5	0.26	0.24	**0.14**	**1.37**	**0.39**	**0.28**	0.19	2.63
Secuencia 6	**0.61**	**0.63**	**0.06**	**1.48**	0.45	0.53	0.06	3.1
Secuencia 7	**0.67**	**1**	**0.05**	**0.95**	0.63	0.98	0.06	1.49

Tabla 8. Comparativa de la extensión 3 y la versión básica. Se puede observar la mejora en las 2 secuencias que introducen oclusiones severas, pero rendimiento inferior en el resto de secuencias, además de inestabilidad y un tiempo excesivo de procesamiento.

5.3.6. Extensión 4: histograma múltiple

Esta extensión es la primera que introduce información espacial con el fin de mejorar la localización del objetivo en situaciones con fondos confusos u oclusiones parciales. Para ello divide el histograma atendiendo a las regiones en que podemos dividir perceptualmente al objetivo, y el peso se calcula de forma ponderada como el producto de los sub-pesos asociados a cada región.

Las pruebas arrojan resultados dispares; mientras que en las secuencias 6 y 7 los resultados son aceptables, aunque sin resolver localizaciones erróneas, las secuencias 4 y 5 tienen unos resultados muy pobres. De nuevo aparecen problemas de inestabilidad, aunque en este caso no sólo en la situación de las partículas. Esta vez también se ven afectados los parámetros relacionados con el tamaño del objetivo, deformándose la región de detección de forma incontrolada (ver Figura 52). Tampoco resuelve la oclusión parcial de la secuencia 5, objeto de estudio desde el inicio, cuando los pesos individuales son una gran ayuda para distinguir el objetivo en tales situaciones.

Figura 52. Inestabilidad de la extensión 4. Deformación de la región de detección en la secuencia 4.

Por otra parte, la carga computacional que supone realizar 3 histogramas en vez de uno produce un incremento en t_f de hasta el 70%, una cifra nada despreciable. Un resumen de los resultados del algoritmo se presenta en la Tabla 9.

	Versión básica				Ext. 4: histograma múltiple			
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Secuencia 4	**0.54**	**0.69**	0.09	**1.57**	0.13	0.14	**0.01**	2
Secuencia 5	**0.26**	**0.24**	0.14	**1.37**	0.22	0.19	**0.12**	2.32
Secuencia 6	**0.61**	**0.63**	**0.06**	**1.48**	0.5	0.61	0.06	2.31

Tabla 9. Comparativa de la extensión 4 y la versión básica. Se puede observar un pobre rendimiento en las secuencias 4 y 5, tanto por la inestabilidad que deforma la región de detección como por la incapacidad para resolver oclusiones severas. También cabe destacar el incremento en t_f.

5.3.7. Extensión 5: matriz de covarianzas

En esta extensión se proponía otro mecanismo para introducir información espacial: asociar el valor RGB de cada píxel a su localización espacial y calcular la matriz de covarianza asociada a la región, para luego comparar dichas matrices, de menor tamaño que los histogramas, mediante la distancia de Forstner [49].

Los resultados extraídos se resumen en la Tabla 10.

	Versión básica				Extensión 5: covarianza			
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Secuencia 4	**0.54**	**0.69**	0.09	**1.57**	0.53	0.65	**0.04**	9.5
Secuencia 5	0.26	0.24	0.14	**1.37**	**0.5**	**0.63**	**0.04**	7.7
Secuencia 6	**0.61**	0.63	0.06	**1.48**	0.6	**0.63**	**0.04**	7.6
Secuencia 7	0.67	1	0.05	**0.95**	**0.74**	**1**	**0.03**	2.6

Tabla 10. Comparativa de la extensión 5 y la versión básica.

Como se puede observar, esta extensión ofrece buenos resultados de calidad y precisión en todos los vídeos. El rendimiento es cercano al de la versión básica, pero se pueden extraer varias conclusiones:

- A diferencia de los anteriores algoritmos, éste obtiene resultados similares en todas las pruebas, sin ningún valor alejado o extraño. Además, la desviación típica es mínima. Estos hechos demuestran una estabilidad superior, y una capacidad de generalización que no poseen el resto de extensiones. Atendiendo a la Figura 59, se puede observar que las partículas se mantienen agrupadas en todo momento y no se expanden por la escena.

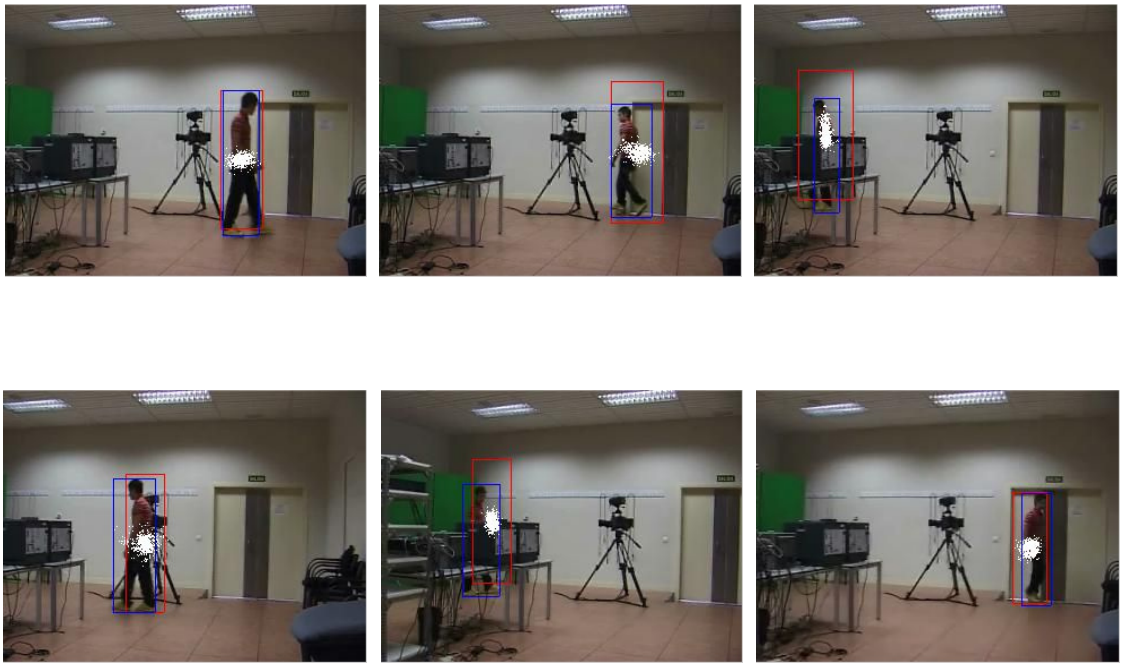


Figura 53. Estabilidad de la extensión 5. Las partículas se mantienen agrupadas a lo largo del vídeo, adaptándose a la velocidad del objetivo. Se muestran las secuencias 5 y 6 (las más largas) para demostrarlo.

- Como se puede ver en las secuencias anteriores, resuelve satisfactoriamente las oclusiones, destacando la de la secuencia 5 con el mejor resultado de todos los algoritmos propuestos.

- A pesar de haber llegado a resultados satisfactorios, que cumplen nuestras expectativas de resolver los problemas de la versión básica (generalización y resolución de la oclusión de la secuencia 5), el algoritmo tiene un pero: el tiempo de procesamiento es excesivo. Un valor 6 veces superior al de la versión básica lo hacen inviable de cara a cualquier aplicación real, ya que, por poner un ejemplo, para hacer *tracking* en una secuencia de 10 s a 25 *fps*, el algoritmo requeriría $25\,fps * 10\,s * 8\,s/frame = 2000\,s \approx 33\,min$ para llevarlo a cabo.

5.4. ALGORITMO FINAL

En base a los resultados obtenidos, tenemos 2 algoritmos que funcionan razonablemente bien: la extensión 1, basada en corrección de pesos, y la extensión 5, que emplea similitud de matrices de covarianza. Esta segunda extensión generaliza mejor y consigue resolver todos los problemas planteados, pero tiene un tiempo de cálculo excesivo. Si este factor no fuera crítico, se seleccionaría dicha extensión. Suponiendo que requerimos un compromiso entre todos los factores analizados, el algoritmo que mejor rendimiento presenta es la modificación del algoritmo SIR en el que los pesos son filtrados para mejorar la calidad de la estimación. En particular, la corrección mediante funciones exponencial y sigmoide arrojan los mejores resultados.

Sin embargo, puede resultar de gran utilidad en otras secuencias adaptar la desviación típica del ruido *on-line*, para situaciones en las que el objetivo se pierda o haya oclusiones severas. Por ello, se ha decidido utilizar como algoritmo final una versión mixta de ambas extensiones, que permita una mejor generalización.

El resto de extensiones no han aportado la mejora esperada, siendo en ambos casos, y al contrario de lo que cabría pensar inicialmente, un algoritmo peor que el de partida, en términos como estabilidad (extensión 3) y calidad de la estimación (extensión 4).

En la Tabla 11 se muestra una comparación general de todos los algoritmos, en la que se evalúan los parámetros de calidad y la desviación típica de los resultados.

Nº Sec	Sec. 4			Sec. 5			Sec. 6			Sec. 7		
Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$
Básico	0.54	0.69	0.09	0.26	0.24	0.14	0.61	0.63	0.06	0.67	1	0.05
Ext. 1 sigmoide	0.58	0.76	0.06	0.26	0.23	0.11	**0.63**	0.67	0.04	0.71	1	0.04
Ext. 2	0.57	0.67	0.57	0.31	0.31	0.19	0.61	**0.73**	0.05	0.61	0.73	0.05
Ext. 3	0.58	0.56	0.14	0.39	0.28	0.19	0.45	0.53	0.06	0.63	0.98	0.06
Ext. 4	0.13	0.14	0.01	0.22	0.19	0.12	0.5	0.61	0.06	0.58	0.81	0.06
Ext. 5	0.53	0.65	**0.04**	**0.5**	**0.63**	**0.04**	0.6	0.63	0.04	**0.74**	1	0.03
Versión final	**0.59**	**0.77**	0.05	0.27	0.22	0.11	0.6	0.6	**0.04**	0.65	**1**	**0.02**

Tabla 11. Comparación general de los algoritmos desarrollados. Los mejores resultados corresponden a la versión final y a la extensión 5, un algoritmo bueno pero excesivamente lento.

También se pueden evaluar los avances de las distintas extensiones en la gráfica de la Figura 54, que representa la calidad $\overline{Q1}$ medida a lo largo de la secuencia 4. Lo más notable es el hecho de que la versión final es la que mejor resuelve la oclusión en los frames finales. También cabe destacar que su comienzo es el peor junto al de la extensión 2, ya que el ruido comienza siendo totalmente dinámico (los pesos son altos), pero como la velocidad inicial es 0, se queda estancado hasta que se adapta a la velocidad del objetivo.

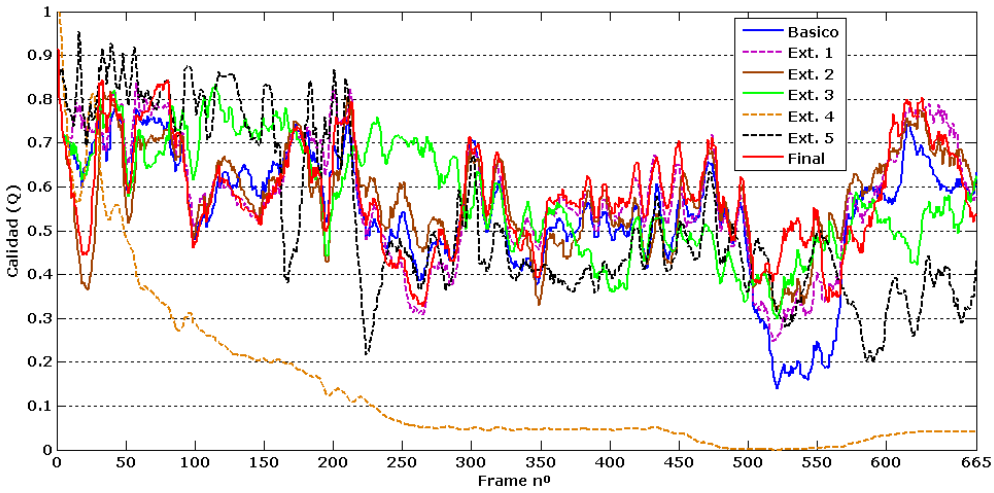
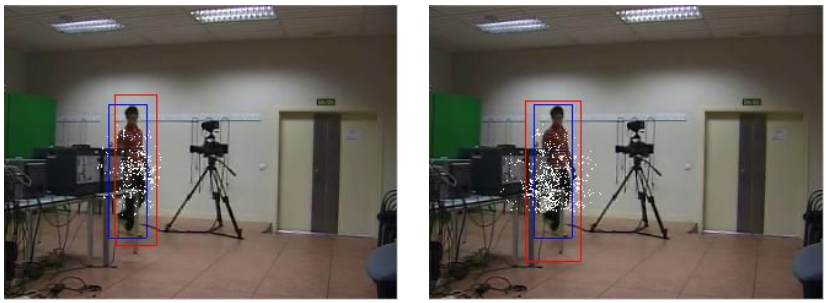


Figura 54. Gráfica comparativa de calidad Q de todas las implementaciones. Secuencia 4.

Como se ha podido observar en las figuras anteriores, los cambios que introduce la adaptación del ruido en la versión final respecto a la extensión 1 no son muy notables, e incluso el rendimiento empeora en las secuencias 6 y 7. Sin embargo, dicho cambio resulta interesante para secuencias largas o más complejas (como pueden ser las secuencias 4 y 5). En la figura 55 se pueden observar 2 situaciones en las que el algoritmo combinado mejora la detección: reduciendo la aparición de pesos inservibles y adaptándose rápidamente a pérdidas del objetivo.



(a)

(b)

Figura 55. Mejora resultante de la combinación de las extensiones 1 y 2. (a) Se reducen pesos molestos que no detectan correctamente el objetivo. (b) La reacción ante una pérdida es más rápida.

5.5. MULTITRACKING

Una vez obtenido el algoritmo final, se aplicará a una secuencia con múltiples objetivos para probar su rendimiento. Hay que destacar que estas secuencias son uno de los principales retos del *tracking*, ya que las oclusiones entre objetivos móviles son más difíciles de resolver por el problema que representa asociar correctamente los objetivos tras la oclusión (ver Figura 56)

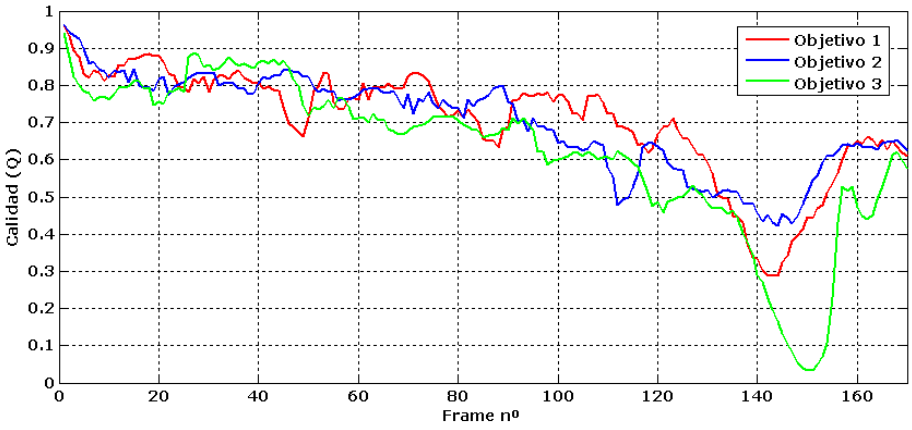
Figura 56. Asociación de los objetivos tras una oclusión en *multitracking*. Es importante que el algoritmo no los confunda y sea capaz de determinar quién es quién tras la "fusión" que se produce en el encuentro.

Utilizaremos para el experimento la secuencia 6, en la que aparecen 4 objetivos móviles. Realizaremos el seguimiento de 3 de ellos, dos personas que caminan juntas con un movimiento vertical, y el hombre del que realizamos el seguimiento en las secuencias 6 y 7, cuyo movimiento es más complejo e incluye oclusiones parciales con el resto de objetivos.

Los resultados son muy satisfactorios, ya que se resuelven con éxito las oclusiones entre los distintos objetivos y se alcanzan unos valores de calidad y precisión elevados (ver Tabla 12). El único problema se presenta al final de la secuencia, donde el objetivo verde se confunde con el rojo (ver Figura 57). Sin embargo, la recuperación es rápida y podemos extraer la conclusión de que el algoritmo es viable para el seguimiento de múltiples objetivos simultáneamente.

Medida	$\overline{Q1}$	$\overline{Q2}$	$\overline{\sigma_Q}$	t_f
Obj. rojo	0.7	0.64	0.05	3.7
Obj. azul	0.7	0.81	0.05	3.7
Obj. verde	0.62	0.64	0.03	3.7

Tabla 12. Resultados del algoritmo final aplicado a *multitracking*.



(a)

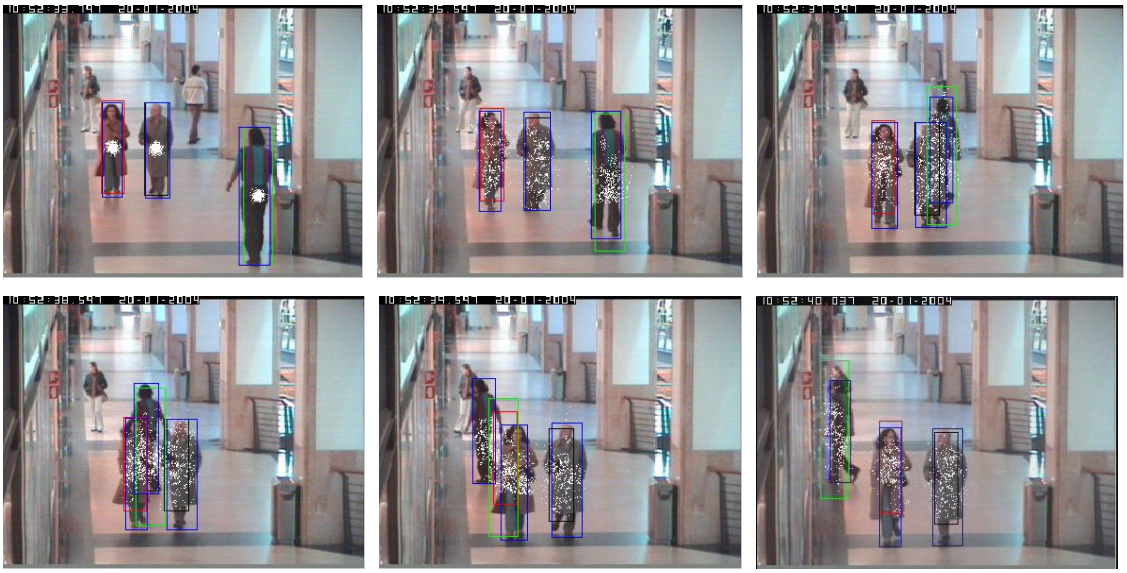


Figura 57. Algoritmo final aplicado al *multitracking*. Secuencia de imágenes.

6. CONCLUSIONES Y TRABAJO FUTURO

Como se ha podido observar en el apartado anterior, el algoritmo SIR implementado, incluso aplicando alguna de las mejoras propuestas, no obtiene resultados óptimos. En esta sección se resumen algunos de los motivos por los que los resultados se ven mermados, ofreciendo alternativas y líneas de investigación futuras que podrían mejorarlos.

- Representación rectangular

  El utilizar esta representación del objetivo limita la aplicación del algoritmo a Figuras con una forma determinada o ángulos de cámara determinados (ver Figura 58(a), (b)). Si la Figura es compleja o el ángulo de cámara inadecuado, se introduce parte del fondo en el histograma que empeora los resultados del *tracking*. Además, es poco flexible frente a cambios en el tamaño o la forma del objeto, introduciendo mucho ruido (ver Figura 58(c)).

  Se podría mejorar dicha solución con otra representación de objeto, como una elipse o combinaciones de estas, o mejor aún, empleando una representación basada en silueta.

Figura 58. Limitación impuesta por representación rectangular del objeto. (a),(b) Forma o ángulo no adecuado. (c) Cambio de forma no aproximable por rectángulo, introduce mucho ruido.

- Tiempo de procesamiento

  Como se comentó al principio del estudio, muchas de las aplicaciones reales del *tracking* requieren funcionamiento en tiempo real. El uso del entorno de programación Matlab es un factor determinante en el tiempo de cómputo. Los resultados reflejan tiempos de cálculo del orden de segundos para un sólo *frame*, cuando una cámara de vídeo convencional graba a una tasa de 25 imágenes por segundo. Este hecho impide que el

sistema desarrollado pueda ser utilizado para aplicaciones en tiempo real, como video vigilancia.

Para este propósito, se podría migrar la aplicación a otro entorno más eficiente en términos de tiempo de procesamiento, como por ejemplo C/C++.

- Mejora de la descripción espacial del objeto

  Algunos de los problemas del seguimiento se derivan de errores al confundir al objeto con el fondo o con otro objeto similar cercano. Esto es debido a que la característica seleccionada para el *tracking*, el histograma de color, no incorpora información espacial. Por tanto, regiones de la imagen con histogramas similares al de referencia pueden dar buenos resultados, haciendo que la partícula no fije correctamente el objetivo (véase Figura 59(a))

  Para corregir este defecto se utilizó la matriz de covarianzas como característica, que sí incorpora información espacial y que tiene un tamaño más reducido. También se utilizó un histograma dividido que permite identificar diferentes regiones en el objetivo. Sin embargo, ambas soluciones fracasaron en las pruebas.

  Otra solución es utilizar correlograma, una técnica que también incorpora información espacial y que permite, entre otras cosas, detectar variaciones de orientación del objeto (ver Figura 59(b)).

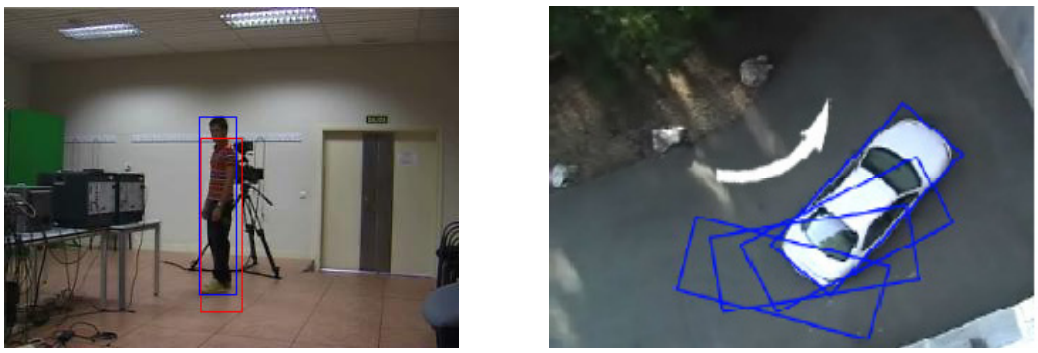


Figura 59. Importancia de la información espacial en *tracking*. (a) Error de detección debido a una región con histograma similar (sustituye la cara y el pelo por la cámara, negra, y las baldosas, de un color rosado). (b) Resultado de utilizar correlograma en el *tracking* de un vehículo.

- Rigidez frente a cambios de apariencia del objeto

  El algoritmo presenta malos resultados para cambios de apariencia en el objeto: cambios en el punto de vista, giros o cambios de iluminación (ver Figura 60). Esto es debido a que el histograma de color varía de manera significativa en estos casos, y al igual que durante una oclusión, el algoritmo pierde al objetivo.

  Existen otros descriptores de objeto que solucionan estos problemas, como los modelos de apariencia multivista, que almacenan y comparan varias vistas del objetivo en cada instante.

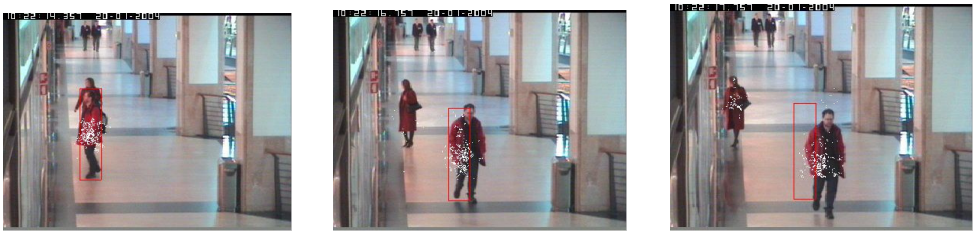


Figura 60. Error derivado de cambio de apariencia (giro del objetivo).

- Necesidad de inicialización manual.

  La posición y tamaño iniciales deben ser fijados manualmente al inicio de la simulación. De cara a una mejora futura, se podría emplear sustracción de fondo en el caso en que la cámara fuera fija, o utilizar técnicas de procesamiento de imagen (segmentación, técnicas morfológicas, modelos gaussianos) para detectar el objeto automáticamente al entrar en escena.

- Parámetros subóptimos

  A pesar de haber hecho diversas pruebas, los valores encontrados no son óptimos, y menos aún generales. No existe una fórmula que establezca valores apropiados en función de las características del vídeo o del objetivo, aunque se podría investigar qué parámetros resultan más adecuados en función del objetivo y la escena, y tener algunos valores preestablecidos (personas al aire libre, cámaras de seguridad, partidos de fútbol…).

7. BIBLIOGRAFIA

[1]
MPEG REQUIREMENTS GROUP. 2002. *ISO/MPEG N4676, MPEG-7 Applications*, v 11.0, N. Day, ed., MPEG Requirements Group.

[2]
BAGDANOV, A., DINI, F., DEL BIMBO, A. AND NUNZIATI, W. 2007. *Improving the robustness of particle-filter based visual trackers using online parameter adaptation*. IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS).

[3]
VEENMAN, C., REINDERS, M. AND BACKER, E. *Resolving motion correspondence for densely moving points*. IEEE Trans. Patt. Analy. Mach. Intell. vol. 23, nº 1, p. 54-72.

[4]
SERBY, D., KOLLER-MEIER, S. AND V. GOOL., L. 2004. *Probabilistic object tracking using multiple features.* IEEE International Conference of Pattern Recognition (ICPR), p. 184-187.

[5]
COMANICIU, D., RAMESH, V. AND MEER, P. 2003. *Kernel-based object tracking*. IEEE Trans. Patt. Analy. Mach. Intell. 25, p. 564-575.

[6]
YILMAZ, A., LI, X. AND SHAH, M. 2004. *Contour based object tracking with occlusion handling in video acquired using mobile cameras*. IEEE Trans. Patt. Analy. Mach. Intell. 26, 11, p. 1531–1536.

[7]
BALLARD, D., AND BROWN, C. 1982. *Computer Vision*, Cap. 8. Prentice-Hall.

[8]
ALI, A. AND AGGARWAL, J. 2001. *Segmentation and recognition of continuous human activity.* IEEE Workshop on Detection and Recognition of Events in Video, p. 28–35.

[9]
YILMAZ, A., JAVED, O. AND SHAH, M. 2006. *Object tracking: A survey.* ACM Computing Surveys (CSUR), v.38 n.4, p. 13-es.

[10]
FIEGUTH, P., AND TERZOPOULOS, D. 1997 *Color-based tracking of heads and other mobile objects at video frame rates.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 21–27.

[11]
COOTES, T. F., EDWARDS, G. J., AND TAYLOR, C. J. 1998. *Active appearance models*. Lecture notes in Computer Science, vol. 1407.

[12]
PASCHOS, G. 2001. *Perceptually uniform color spaces for color texture analysis: an empirical evaluation*. IEEE Transactions on Image Processing 10(6), p. 932-937.

[13]
CANNY, J. F. 1986. *A computational approach to edge detection*. IEEE Trans. Pattern Analysis and Machine Intelligence*, 8 (6), p. 679-698.*

[14]
SHIN, M. C., GOLDGOF, D. B., BOWYER, K. W., AND NIKIFOROU, S. 2001. *Comparison of edge detection algorithms using a structure from motion task*. IEEE Transactions on Systems, Man, and Cybernetics, Part B 31(4), p. 589-601.

[15]
HARALICK, R. M., SHANMUGAM, K., AND DINSTEIN, I. 1973. *Textural features for image classification.* IEEE Trans. Syst. Man, Cybernetics 3, p. 610-621.

[16]
LAWS, K. 1980. *Rapid texture identification*. SPIE Vol. 238 Image Processing for Missile Guidance, p. 376-380

[17]
MALLAT, S. 1996. *Wavelets for a Vision*. Proc. of the IEEE, Vol. 84, No. 4.

[18]
GREENSPAN, H., BELONGIE, S., AND PERONA, P. 1994. *Invariant Texture Recognition Using a Steerable Pyramid.* Proc. Int. Conf. on Pattern Recognition.

[19]
HARRIS, C., AND STEPHENS, M. 1998. *A combined corner and edge detector.* 4th Alvey Vision Conference, p. 147–151.

[20]
SHI, J., AND TOMASI, C. 1994. *Good features to track.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 593–600.

[21]
LOWE, D. 2004 *Distinctive image features from scale-invariant keypoints.* Int. J. Comput. Vision 60, 2, p. 91–110.

[22]
MIKOLAJCZYK, K. AND SCHMID, C. 2005. *A performance evaluation of local descriptors*. IEEE transactions on pattern analysis and machine intelligence, vol. 27, nº 10.

[23]
NAGEL, J. 1979. *On the analysis of accumulative difference pictures from image sequences of real world scenes*. IEEE Trans. on Pattern Analysis and Machine Intelligence.

[24]
PARAGIOS, N. AND DERICHE, R. 2000. *Geodesic active regions and level set methods for supervised texture segmentation*. Int. J. Comput. Vision 46, 3, p. 223-247.

[25]
SHI, J. AND MALIK, J. 2000. *Normalized cuts and image segmentation*. IEEEE Trans. Patt. Analy. Mach. Intelli. 22, 8, p. 888-905.

[26]
ROWLEY, H., BALUJA, S., AND KANADE, T. 1998. *Neural network-based face detection*. IEEE Trans. Patt. Analy. Mach. Intell. 20, 1, p. 23-38.

[27]
BLUM, A. AND MITCHELL, T. 1998. *Combining labelled and unlabelled data with co-training*. In 11th Annual Conference on Computational Learning Theory, p. 92-100.

[28]
BOSER, B., GUYON, I. M., AND VAPNIK, V. 1992. *A training algorithm for optimal margin classifiers*. In ACM Workshop on Conference on Computational Learning Theroy (COLT), p. 142-152.

[29]
VIOLA, P., JONES, M., AND SNOW, D. 2003. *Detecting pedestrians using patterns of motion and appearance*. In IEEE International Conference on Computer Vision (ICCV), p. 734–741.

[30]
MATAS, J. AND SOCHMAN, J. *AdaBoost*. Centre for Machine Perception. Czech Technical University.

[31]
KUHN, H. W. 1955. *The Hungarian method for the assignment problem.* Naval Research Logistics Quarterly, 2, p. 83–87.

[32]
VEENMAN, C., REINDERS, M., AND BACKER, E. 2001. *Resolving motion correspondence for densely moving points*. IEEE Trans. Patt. Analy. Mach. Intell. 23, 1, p. 54–72.

[33]
SHAFIQUE, K. AND SHAH, M. 2003. *A non-iterative greedy algorithm for multi-frame point correspondence*. In IEEE International Conference on Computer Vision (ICCV), p. 110–115.

[34]
TANIZAKI, H. 1987. *Non-gaussian state-space modeling of nonstationary time series*. J. Amer. Statist. Assoc. 82, p. 1032–1063.

[35]
BAR-SHALOM, Y., AND FORTMANN, T. E. 1988. *Tracking and data association*. Boston Academic Press.

[36]
BLACK, M., AND JEPSON, A. 1998. *Eigentracking: Robust matching and tracking of articulated objects using a view-based representation*. Int. J. Comput. Vision 26, 1, p. 63–84.

[37]
AVIDAN, S. 2001. *Support vector tracking*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 184–191.

[38]SATO, K., AND AGGARWAL, J. 2004. *Temporal spatio-velocity transform and its application to tracking and interaction.* Comput. Vision Image Understand. 96, 2, p. 100–128.

[39]
BERTALMIO, M., SAPIRO, G., AND RANDALL, G. 2000. *Morphing active contours.* IEEE Trans. Patt. Analy. Mach. Intell. 22, 7, p. 733–737

[40]
YILMAZ, A., LI, X., AND SHAH, M. 2004. *Contour based object tracking with occlusion handling in video acquired using mobile cameras*. IEEE Trans. Patt. Analy. Mach. Intell. 26, 11, p. 1531–1536.

[41]
WELCH, G. AND BISHOP, G. 2001. *An introduction to the Kalman Filter*. Cap. 4, p. 19-24. Department of Computer Science, University of North Carolina at Chapel Hill

[42]
ROESSER, R. P. 1975. *A discrete state-space model for linear image processing*. IEEE Transactions on Automatic Control AC-20, p. 1-10.

[43]
ISARD, M. AND BLAKE, A. 1998. *Condensation—conditional density propagation for visual tracking*. International Journal of Computer Vision

[44]
MACKAY, D. J. C. 1998. *Introduction to Monte Carlo methods*. Learning in Graphical Models, M. I. Jordan. Kluwer Academic Press, p. 175–204.

[45]
SANJEEV, M., MASKELL, S. AND GORDON, N. 2002. *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking*. IEEE Transactions on Signal Processing.

[46]
DOUCET, A., GODSILL, S. AND ANDRIEU, C. 2000. *On sequential Monte Carlo Sampling methods for Bayesian filterin*g. Statistics and Computing, 10(3), p. 197-208.

[47]
LIU, J.S. AND CHEN, R. 1998. *Sequential Monte Carlo methods for dynamical systems*. J. Amer. Statist. Assoc., vol.93, p 1032-1044.

[48]
PORIKLI, F., TUZEL, O. AND MEER, P. *Covariance tracking using Model Update based on means on Riemannian manifolds*. Mitsubishi Electric Research Laboratories.

[49]
FÖRSTNER, W. AND MOONEN, B. 1999*. A metric for covariance matrices*. Techincal report, Dept. of Geodesy and Geoinformatics, Stuttgart University.

[50]
Caviar Project. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

[51]
PHILLIPS, I. AND CHHABRA, A. 1999. *Empirical performance evaluation of graphics recognition systems*. IEEE Trans. Pattern Anal. Mach. Intell. 21(9), p. 849-870.