UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

INGENIERO DE TELECOMUNICACIÓN

DETECCIÓN Y SEGUIMIENTO DE OBJETOS MÓVILES EN SECUENCIAS DE VIDEO

AUTOR: RAFAEL ORMAECHEA IZQUIERDO

TUTOR: SANTIAGO AJA FERNÁNDEZ

Febrero 2015

RESUMEN DEL PROYECTO

Los avances en el campo de procesado de señales han posibilitado extender el uso de algoritmos complejos de detección y seguimiento a distintas aplicaciones, tales como la video-vigilancia, la visión artificial, etc. Actualmente es posible disponer de estos avances en dispositivos de pequeño tamaño.

En el siguiente proyecto se realiza una recopilación los métodos tradicionales existentes para resolver los problemas de detección y seguimiento; y presentamos el desarrollo de un sistema de vigilancia propio aplicado a blancos móviles en secuencias de vídeo, que pueda convertirse posteriormente en una aplicación para dispositivos móviles.

El método que se ha empleado para la detección de movimiento consiste en la aplicación de una medida de la calidad de imagen, la comparación estructural, y respecto al seguimiento se utilizan algoritmos tradicionales, tal como el filtro de Kalman.

PALABRAS CLAVE

Video-vigilancia, Detección de movimiento, Algoritmos de seguimiento, Comparación estructural, SSIM, filtro de Kalman, filtro alfa-beta

ABSTRACT

Advances in the field of signal processing have made possible the use of complex algorithms (e.g. motion detection and tracking) to different applications, such as video surveillance or artificial vision. Nowadays it is possible to have those new applications in small size mobile devices.

In the former paper we review the most common methods that solve the problems of tracking and movement detection. We as well introduce the development of our own surveillance system applied to moving targets in video sequences, which can be adapted into an application for mobile devices.

The method used for motion detection is based on the application of an image quality measurement, structural similarity, whereas traditional algorithms are used for the tracking problem, such as the Kalman filter.

KEY WORDS

Video surveillance system, Motion detection, Tracking algorithms, Structural comparison, SSIM, Kalman filter, Alfa-beta filter

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. INTRODUCCIÓN

En la actualidad el número de cámaras de video que se instalan alrededor del mundo aumenta cada día, ya sea con motivo de vigilancia y seguridad, o con otros propósitos. Por ejemplo, grandes ciudades como Londres o Chicago cuentan con amplias redes de cámaras con las que pretenden reducir la criminalidad con capacidad para la detección de presencia, seguimiento, análisis de conducta o reconocimiento facial. Otras aplicaciones de la videovigilancia son la detección de intrusos en zonas de seguridad, el guiado de vehículos, el control de tráfico en carreteras, aduanas, puertos y aeropuertos, vigilancia de procesos industriales, control de multitudes… [1, 2, 3, 4].

En cualquier caso, todas estas redes pueden incluir un elevado número de cámaras recogiendo datos de forma continuada. Estas circunstancias hacen muy importante la necesidad de desarrollar métodos que procesen la todos los datos recibidos automáticamente, lo que ha incrementado en los últimos años la investigación en técnicas y procedimientos para desarrollar nuevos sistemas de videovigilancia más eficientes [5, 6]. Todas las aplicaciones mencionadas anteriormente se basan, en definitiva, en el conocimiento de la información de las trayectorias de los objetos móviles y, por tanto, es crucial disponer de métodos robustos de detección y seguimiento en escenarios complejos del mundo real. Algoritmos de detección que se utilizan con frecuencia en videovigilancia por su efectividad son, por ejemplo, modelado y sustracción de fondo (Gaussian mixture model, dynamic textures) y *video segmentation* (Flujo óptico, mean shift) [7].

Debido a la extendida presencia de cámaras en los dispositivos móviles, aparte de los sistemas “profesionales” o comerciales de videovigilancia basados en circuitos cerrados de televisión (CCTV) y cámaras IP, también es posible disponer de sistemas de videovigilancia más sencillos para el uso personal. Buscando una solución de este tipo para aplicaciones personalizadas, en el trabajo anterior [8] se propuso un nuevo mecanismo de detección de movimiento para aplicaciones de videovigilancia. Esta propuesta se basa en el concepto de similitud estructural (SSIM), establecido por Z. Wang en [9,10], que detecta los posibles cambios que se producen entre dos imágenes consecutivas de la secuencia de vídeo.

Se pretende mejorar dicho trabajo añadiendo la funcionalidad de seguimiento de los objetos móviles en secuencias de vídeo, para poder establecer un sistema de videovigilancia de cámara móvil con capacidad de seguir el desplazamiento de los objetos. El seguimiento en video es un problema complejo debido a la pérdida de información provocada al pasar del mundo en 3D a una imagen 2D, sobre la que se añaden problemas como el ruido en la imagen, movimientos complejos, oclusiones, cambios de iluminación, requerimientos de procesado en tiempo real, etc. Entre los métodos propuestos más frecuentemente existen: la correspondencia de puntos, la coincidencia de regiones geométricas o evolución del contorno [7]; así como los basados en filtros de coeficientes fijos (alfa-beta) [11, 12, 13], filtro de Kalman [13, 14] y filtros de partículas [15, 16].

En este proyecto se pretende desarrollar un método de detección de movimiento basado en SSIM sobre el que se implementará un mecanismo de seguimiento utilizando los principios del filtrado, que nos permita disponer de la base para una aplicación de videovigilancia para una cámara móvil. Además, el método estará adaptado para que la detección y seguimiento pueda realizarse en cámaras que giren, simulando el funcionamiento de cámaras que siguen a personas y objetos en movimiento dentro de la zona de cobertura (*pan-tilt camera*).

1.2. OBJETIVOS

El principal objetivo del proyecto es el siguiente:

Desarrollo de un método que permita la detección de objetos en movimiento y el seguimiento de dichos elementos móviles en grabaciones de vídeo, y que sea robusto y ligero para una posible implementación en dispositivos móviles.

Concretamente, este método está dirigido a que pueda emplearse sobre una cámara móvil con capacidad de girar respecto a su eje vertical para seguir el movimiento de personas u objetos móviles dentro de un área de vigilancia. Para simular este comportamiento, se supone una grabación de video fija cuya totalidad es el área de vigilancia y la detección de movimiento se restringe a una zona limitada que simula la imagen obtenida por la cámara.

A partir del objetivo principal podemos concretar los siguientes objetivos secundarios:

1. Desarrollo y adecuación de un algoritmo de detección de movimientos, que sea capaz de funcionar adecuadamente teniendo en cuenta el movimiento en todas direcciones de la cámara y corregir el efecto del desplazamiento del fondo de la escena.
2. Implementación de un esquema de seguimiento sencillo y de poca carga computacional que permita la determinación con fiabilidad de la posición del objetivo, en presencia de ruido o de posibles oclusiones o pérdidas de información momentáneas.
3. Desarrollo de un programa basado en lenguaje Matlab que conjugue de forma eficiente los algoritmos de detección y seguimiento y que cumpla con los requisitos especificados anteriormente.

4. Realización de un estudio sobre la idoneidad del método propuesto para la detección y seguimiento de blancos en movimiento, en distintos entornos, situaciones y tipos de movimientos.

1.3. FASES Y MÉTODOS

Para el desarrollo del trabajo y la consecución de los anteriores objetivos, se han seguido los siguientes puntos:

1. Planteamiento del problema: Estudio de la literatura relacionada con seguimiento de blancos móviles. Análisis de las diversas técnicas y métodos disponibles.
2. Estudio de filtros de seguimiento: Caracterización matemática y teórica del filtro alfa-beta y del filtro de Kalman. Evaluación de su comportamiento mediante la variación de los distintos parámetros que controlan cada filtro. Estudio comparativo de ambos filtros en relación a su precisión y velocidad de seguimiento.
3. Aplicar la detección de movimiento a un escenario móvil: Caracterización del comportamiento del algoritmo de detección SSIM al aplicarse sobre imágenes que se desplazan en todas direcciones. La detección sólo se realiza sobre una parte de la imagen, que se va desplazando sobre la imagen total.
4. Utilización conjunta de los algoritmos de detección y seguimiento: Estudio del funcionamiento del filtro de seguimiento sobre un video real. No se considera que la cámara se mueve siguiendo al objetivo, sino que la detección se realiza sobre toda la imagen.
5. Desarrollo del prototipo en Matlab: La implementación final combina los dos puntos anteriores, se realiza el seguimiento al blanco detectado y se considera que la cámara se está moviendo siguiendo al objetivo, de forma la detección sólo se realiza sobre una zona determinada de la imagen.
6. Entrenamiento y validación de la aplicación. El entrenamiento se realiza mediante una serie de vídeos prueba que contienen las características básicas que queremos evaluar para determinar el comportamiento inicial del programa.
7. Corrección de problemas y refinamiento de la aplicación. Solución de los problemas detectados durante la fase de entrenamiento.
8. Evaluación del funcionamiento final de la aplicación. Para esta fase se empleará un conjunto de vídeos variados en distintos entornos y condiciones.
9. Recopilación y estudio de los resultados obtenidos. El análisis de los resultados permite extraer las conclusiones finales y determinar si el método empleado es apropiado.

1.4. ESTRUCTURA DEL TRABAJO

El resto del presente trabajo se estructura como sigue: después de la introducción que acabamos de terminar, se inicia una serie de capítulos referidos al estado del arte de temas

que son de relevancia para este proyecto. En el capítulo 2 se hace un repaso de cómo funcionan los sistemas de video-vigilancia, las fases que los comprenden y principales métodos empleados. En el capítulo 3 se estudian en detalle algunos de los algoritmos de seguimiento más utilizados en aplicaciones de seguimiento similares a la que pretendemos implementar. Por último en el capítulo 4 se hace una presentación de los métodos de análisis de calidad de imagen y de cómo se pueden reconvertir para utilizarlos en nuestro sistema.

El segundo bloque de capítulos se centra en nuestro sistema de detección y seguimiento propiamente dicho. El capítulo 5 explica de forma detallada el diseño del sistema, describiendo el funcionamiento de cada elemento y procedimiento involucrado. En el capítulo 6 se analizan las distintas partes del sistema, como el funcionamiento de los filtros, etc. Finalmente, en el capítulo 7 se realizan pruebas sobre el sistema final y se analizan los resultados, y en el capítulo 8 se recogen las conclusiones.

Por último, se añaden unos anexos que amplían los contenidos desarrollados durante el resto del trabajo. El anexo A recoge un conjunto de pruebas realizadas sobre el algoritmo de detección. El anexo B muestra las pruebas realizadas en relación con el análisis de los parámetros de los filtros de seguimiento, alfa-beta y Kalman. El anexo C incluye los códigos desarrollados en Matlab.

CAPÍTULO 2

ESTADO DEL ARTE: SISTEMAS Y TÉCNICAS DE VIGILANCIA

2.1. SISTEMAS DE VIGILANCIA

En este apartado vamos a comentar la evolución de los sistemas de vigilancia. Según Valera y Velastin [1], podemos clasificar los sistemas de vigilancia en tres generaciones, dependiendo de la tecnología que emplean y las funcionalidades que proporcionan.

La evolución tecnológica de los sistemas de videovigilancia comenzó con los sistemas de CCTV (sistemas de circuito cerrado de televisión). Estos sistemas consisten en un número de cámaras situadas en múltiples localizaciones remotas y conectadas a un conjunto de monitores, normalmente ubicados en una habitación de control.

Actualmente, la mayoría de los sistemas CCTV usan técnicas analógicas para la distribución y almacenamiento de imágenes. Las cámaras convencionales de CCTV normalmente usan dispositivos de carga acoplada (CCD) digitales para capturar las imágenes, pero también cámaras térmicas y de visión nocturna. Luego, la imagen digital se transforma en una señal de vídeo compuesta y analógica que se conecta a la matriz CCTV, a los monitores y a los equipos de grabación mediante cable coaxial.

Sin embargo, el hecho de tratarse de tecnología totalmente analógica implica algunos problemas, ya que la conversión digital-analógica provoca cierta degradación en la imagen y la señal analógica es por si misma susceptible al ruido. Otro problema es que estos sistemas dependen totalmente de un operario que debe encargarse personalmente de la vigilancia de todo el panel de control, tarea que se complica al aumentar el número de cámaras del sistema. Sin embargo, estos sistemas CCTV son muy utilizados en muchos ámbitos, debido a su sencillez de funcionamiento. Una mejora de los sistemas CCTV consiste en incluir tecnología digital. Estos sistemas CCTV más modernos basados en el uso de cámaras IP conectadas a una red permiten mayores prestaciones, como almacenamiento digital, facilidad de uso y de integración en otros sistemas…

Por tanto, es posible disponer de sistemas CCTV digitales, tomando ventaja del formato digital inicial de las imágenes capturadas, usando ordenadores de altas prestaciones y combinándolos con tecnología de visión artificial. La mejora tecnológica proporcionada por estos sistemas ha llevado al desarrollo de sistemas semiautomáticos, conocidos como sistemas de segunda generación. La mayor parte de la investigación en sistemas de videovigilancia de segunda generación se basa en el desarrollo de algoritmos para la detección automática y en tiempo real de eventos, ayudando al usuario a reconocer dichos sucesos.

Los llamados sistemas de vigilancia de tercera generación hacen referencia a los sistemas capaces de manejar un gran número de cámaras, muchos puntos de vigilancia, recursos dispersos geográficamente y reflejar la naturaleza jerárquica y distribuida del proceso humano de vigilancia. Desde el punto de vista del procesado de imagen, estos sistemas se basan en la distribución de las capacidades de procesado sobre la red, y en caso de dispositivos de procesado de señal integrados, deben proporcionar las ventajas de escalabilidad propias de los sistemas distribuidos. Los principales objetivos que los sistemas de vigilancia de tercera generación deben procurar son: proporcionar un buen entendimiento de la escena y atraer la atención del operador responsable en tiempo real hacia las situaciones de mayor importancia.

En líneas generales, el objetivo último de las generaciones de sistemas de vigilancia consiste en diseñar sistemas inteligentes capaces de sustituir a la videovigilancia tradicional basada en la observación pasiva de imágenes, que es ineficiente cuando el número de cámaras excede la capacidad del operario de controlarlas, y capaces de analizar y comprender los comportamientos de personas en secuencias de vídeo.

A continuación se recogen algunas de las características principales de las generaciones de sistemas de vigilancia

1º generación

Técnicas	Sistemas analógicos CCTV.
Ventajas	Proporcionan buenas prestaciones en algunas situaciones. Emplean una tecnología que ya es conocida y madura.
Problemas	Uso de tecnología analógica para el almacenamiento y distribución de las imágenes.
Avances posibles	Tecnología digital vs analógica. Grabación de vídeo digital Compresión de vídeo CCTV.

2º generación

Técnicas	Videovigilancia automatizada combinando tecnología de visión por computadora con sistemas CCTV.
Ventajas	Incrementa la eficiencia de la vigilancia de los sistemas CCTV.
Problemas	Se requieren robustos algoritmos de detección y seguimiento para el análisis de comportamiento y conducta.
Avances posibles	Algoritmos de visión artificial robustos en tiempo real. Aprendizaje automático de variabilidad en la escena y de modelos de comportamiento.

3º generación

Técnicas	Sistemas de vigilancia automatizados de área amplia.
Ventajas	Información más precisa como resultado de combinar diferentes tipos de sensores. Distribución.
Problemas	Distribución de la información. Metodología de diseño/diseño de la metodología. Plataformas multi-sensores.
Avances posibles	Inteligencia distribuida frente a centralizada. Fusión de datos. Esquema de razonamiento probabilístico. Técnicas de vigilancia multicámara.

Tabla 2.1. Evolución de los sistemas de vigilancia

2.1.1. APLICACIONES

La creciente demanda de seguridad por parte de la sociedad conduce a la necesidad de actividades de vigilancia en muchos entonos. Actualmente, la demanda de vigilancia remota por motivos de seguridad ha recibido especial atención, sobre todo en los siguientes campos:

- Aplicaciones de transporte, como aeropuertos, entornos marítimos, vías de ferrocarril y metro, autovías, donde se utiliza para medir el volumen de tráfico y reunir en tiempo real información sobre el tráfico.
- Reconocimiento basado en el movimiento, esto es, identificación de personas basado en su modo de andar, detección automática de objetos…
- Navegación de vehículos, es decir, emplear rutas adquiridas por vídeo y capacidades para evitar obstáculos.
- Lugares públicos como bancos, supermercados y aparcamientos, y también hogares privados.
- Vigilancia remota de actividades humanas, como asistencia a partidos de fútbol, manifestaciones…
- Vigilancia para obtener cierto control de calidad en procesos industriales.

2.2. TÉCNICAS EMPLEADAS EN SISTEMAS DE VIGILANCIA

A continuación pasamos a reseñar las diferentes etapas que conforman un sistema inteligente de videovigilancia, según se definen en la bibliografía. Según [1] y [20] las etapas más comunes son: detección y reconocimiento de objetos, seguimiento de objetos, análisis de comportamiento y almacenamiento en base de datos.

- Detección de objetos: Es el primer paso en la mayoría de sistemas. Consiste en determinar y separar las regiones que corresponden con objetos en movimiento del resto de la imagen. Este paso es muy importante para etapas posteriores, como el seguimiento y análisis, ya que limita las zonas de la imagen que deben ser procesadas.
- Seguimiento de objetos: Implica el seguimiento y el cálculo de la trayectoria de los objetos detectados entre fotogramas consecutivos de la secuencia de vídeo.
- Análisis de comportamiento: Consiste en el reconocimiento de patrones de movimiento considerados como peligrosos dentro de los vídeos analizados.
- Base de datos: La etapa final es el almacenamiento de forma eficiente y ordenada de los distintos tipos de datos obtenidos durante la vigilancia para luego recuperarlos en caso de necesidad.

Otros posibles elementos dentro de un sistema de vigilancia son:

- Clasificación de objetos: Diferentes regiones en movimiento pueden corresponder con diferentes tipos de objetos. Para un posterior seguimiento o evaluación de comportamiento de los elementos de interés, puede ser importante clasificar correctamente los distintos tipos de objetos en movimiento.
- Identificación de personas: Es posible modelar características físicas y biométricas de ciertas personas, como la forma de la cara o la forma de andar, para identificar a personas que tienen o no permitida la presencia en una determinada zona.
- Fusión de datos: La fusión de datos es necesaria cuando disponemos de un sistema formado por varios dispositivos de entrada. En estos casos tenemos varias cámaras que permiten cubrir un área mayor y eliminar ambigüedades, y cada una proporciona una información diferente. Por tanto es necesario generar una única medida a partir de múltiples observaciones.

A continuación vamos a ver algunas de las posibles implementaciones o técnicas existentes en la literatura para llevar a cabo las etapas de detección, seguimiento y análisis de conducta. Para realizar este repaso a los algoritmos más comunes empleados en visión artificial, vamos a seguir la clasificación realizada por Yilmaz, Javed y Shah en [7] y Ko en [20].

En la siguiente figura se muestra un esquema de las distintas fases que conforman un sistema genérico de vigilancia inteligente y automático.

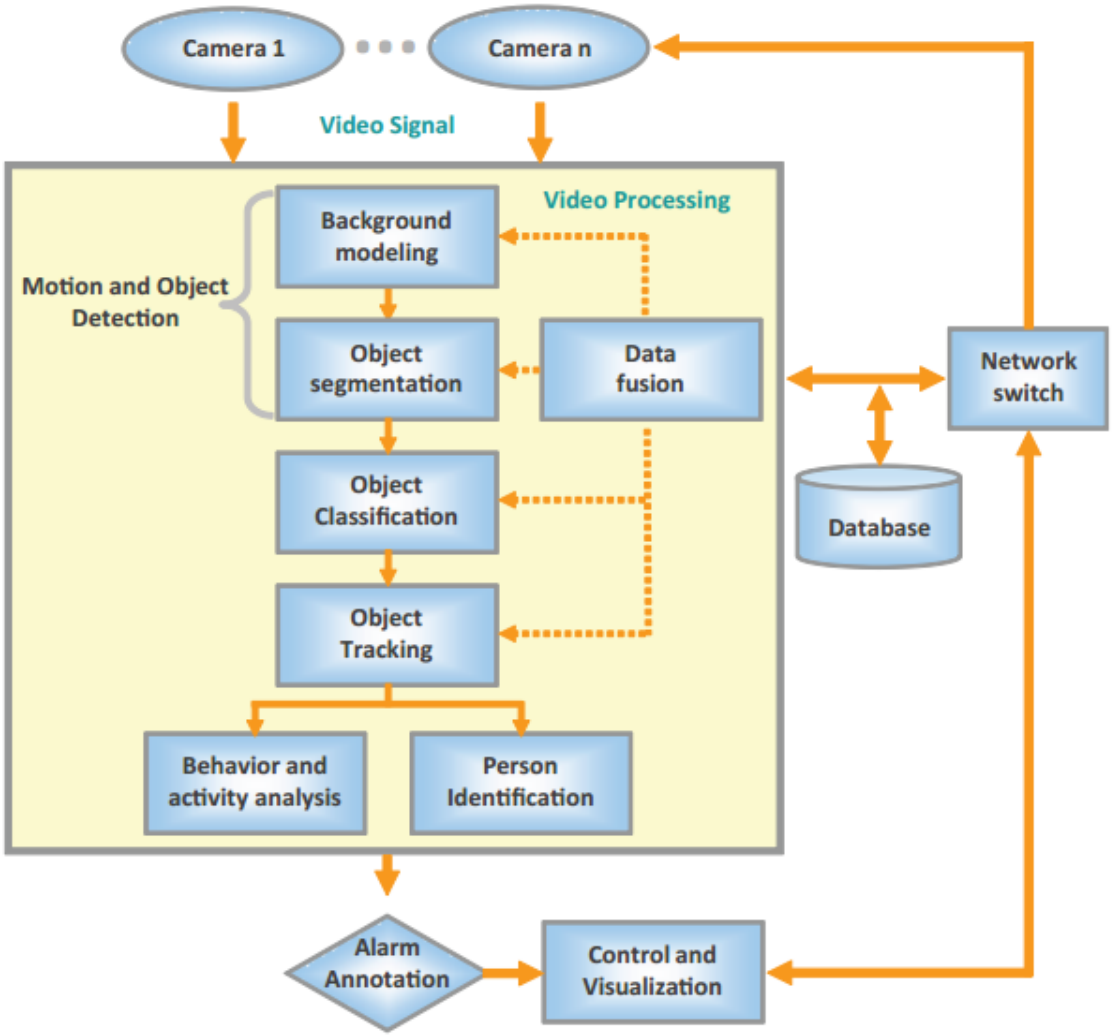


Figura 2.1. Esquema de las etapas de un sistema de vigilancia. Imagen obtenida de [20]

2.2.1. DETECCIÓN DE OBJETOS

El primer paso de la mayoría de sistemas de videovigilancia comienza con la detección de objetos o personas en movimiento. La detección de objetos principalmente consiste en separar regiones que corresponden con objetos en movimiento del resto de la imagen. El resto de procesos, como el seguimiento, dependen profundamente de esta etapa. A continuación veremos diferentes técnicas de detección:

- **Modelado y substracción del fondo**

El concepto básico consiste en mantener un modelo estático del fondo del entorno en el que se está grabando, y comparar la secuencia actual del vídeo con el fondo. Los objetos en movimiento se detectan encontrando los cambios en la imagen diferencia, obtenida al restar la imagen actual y el fondo. Los píxeles que han sufrido algún tipo de cambio se marcan para un

procesado posterior. Normalmente, se aplica un algoritmo de conexión o relleno para obtener regiones conectadas que se corresponden con los objetos.

Para llevar a cabo el modelado y substracción del fondo existen varios métodos:

- Modelo Gaussiano (Gaussian model)

El primer método que dio popularidad al modelado del fondo consistía en modelar el color de cada píxel del fondo como una función Gaussiana 3D (en espacio de color YUV). La media y la covarianza se calculan en los primeros fotogramas consecutivos. Una vez que se dispone del fondo, se calcula para cada píxel entrante la probabilidad de su color, y los píxeles que se desvían del fondo se catalogan como objetos en movimiento.

Sin embargo, a veces un modelo basado en una sola función Gaussiana no es suficiente, especialmente en entornos de exterior, ya que en estos casos pueden percibirse varios colores debido a sombras o reflejos. Una mejora se consigue usando un modelo multimodal, como una mezcla de varias funciones Gaussianas, para caracterizar el color de cada píxel.

- Modelo no paramétrico

En este caso, además de emplearse información del color de cada píxel, se incorpora información espacial. Así, el píxel actual no sólo se compara con su correspondiente píxel del fondo, sino que también se compara con los píxeles próximos. En consecuencia, este método puede soportar pequeños movimientos de la cámara o en el fondo.

- Eigen-background

Esta técnica parte de un punto de vista diferente, ya que modela el fondo mediante el uso de autovectores, que recogen todos los cambios de iluminación posibles en la imagen. El modelo se crea tomando varias muestras de la imagen y eligiendo los autovectores más importantes. Los objetos en movimiento no pueden ser bien descritos por este modelo, por tanto la detección se realiza proyectando la imagen actual sobre el modelo de fondo y encontrando las diferencias entre ambas imágenes.

En la siguiente imagen, tomada de [7], se puede ver cómo funciona este método. En (a) se tiene la imagen de entrada con objetos, en (b) la imagen creada a partir de proyectar la imagen de entrada sobre el autoespacio, y en (c) la imagen de diferencia.

(a) (b) (c)

Figura 2.2. Substracción del fondo por descomposición en autoespacios

En conclusión, los métodos de modelado y substracción del fondo son muy utilizados porque son capaces de caracterizar cambios de iluminación, ruido y movimientos periódicos del fondo y pueden detectar objetos en gran variedad de situaciones, además de ser eficientes computacionalmente.

Sin embargo, en la práctica a veces las regiones que corresponden a objetos se encuentran incompletas. Puede darse que los objetos estén divididos en varias regiones o que tengan huecos en su interior debido a que no hay garantías de que las características del objeto sean diferentes de las características del fondo.

Por otro lado, la principal limitación del modelado del fondo es que necesita que las cámaras se encuentren inmóviles, ya que el movimiento distorsiona la caracterización del fondo. En estos casos habría que utilizar métodos que fueran capaces de regenerar el fondo cada cierto intervalo de tiempo.

- **Segmentación**

La segmentación de video puede definirse como la división de las secuencias de imágenes en regiones perceptualmente similares. Cada algoritmo de segmentación se enfrenta a dos problemas: establecer un criterio para determinar una buena partición y un método para lograr particiones eficientes.

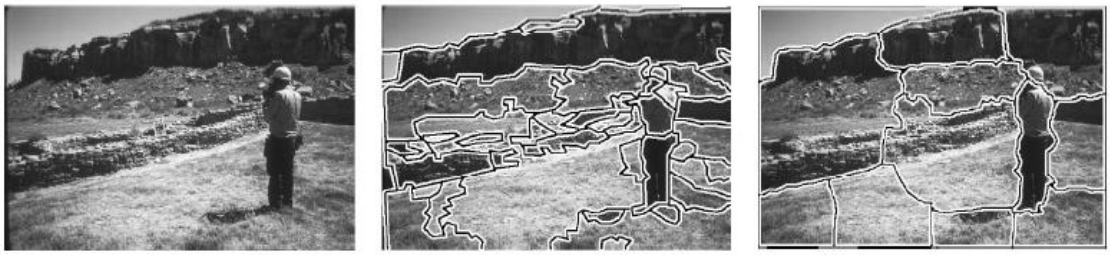
- Mean Shift (Cambio medio)

Desarrollado por Camaniciu [21], el método se basa en detectar grupos (*clusters*) en la unión del espacio formado por el color y las coordenadas espaciales. Dada una imagen, el algoritmo se inicializa eligiendo un gran número de centros de grupo elegidos aleatoriamente entre los datos. Cada centro se desplaza al punto que se corresponde con la media de los datos que se encuentran dentro de un elipsoide centrado en el centro del grupo. El vector que se origina entre el nuevo centro y el original se llama vector de cambio medio (*mean shift vector*) y se calcula de forma iterativa hasta que el centro del grupo no se desplaza.

Su principal limitación es que necesita una elección precisa de diversos parámetros para funcionar de manera más eficiente, además de ser costosa desde el punto de vista computacional.

- Graphcut

Este método se plantea como un problema de partición de conjuntos, donde cada píxel se considera como un vértice de un gráfico. La similitud entre dos píxeles se considera como el peso del borde entre estos dos vértices, y se computa como la semejanza entre el color, brillo o textura entre los píxeles. La segmentación se lleva a cabo formando bordes que se consiguen uniendo puntos con un peso elevado.



(a) *(b)* *(c)*

Figura 2.3. Segmentación de la imagen en (a), mediante la técnica mean-shift (b) y graph-cuts (c). Imagen tomada de [7].

Un inconveniente de este sistema es que para imágenes grandes los requisitos de memoria y procesado pueden ser elevados. Por el contrario, requiere seleccionar menos parámetros de forma manual que el modelo anterior.

- Contornos activos

Esta forma de realizar la segmentación consiste en obtener el contorno del objeto, de forma que éste encierre fielmente el borde del objeto. La formación del contorno se gobierna mediante un funcional de energía que define la adecuación del contorno a la región del objeto. El funcional de energía tiene comúnmente la siguiente forma:

$$E(\Gamma) = \int_0^1 E_{int}(V) + E_{im}(V) + E_{ext}(V)\, ds$$

Donde s es la longitud de la curva del contorno Γ, E_{int} incluye las limitaciones de regularización, E_{im} incluye energía basada en apariencia y E_{ext}especifica limitaciones adicionales. E_{int} normalmente incluye un término de curvatura para encontrar el contorno más pequeño. E_{im} se puede calcular de forma local o global. Localmente, se calcula mediante el gradiente evaluado en el contorno y las características globales (color y textura) se calculan dentro y fuera de los objetos.

Un punto importante en estos métodos es la inicialización del contorno. Una de las maneras más comunes consiste en situar el contorno fuera de la región del objeto y encogerlo sucesivamente hasta encontrar el borde del objeto. Esta aproximación, sin embargo, requiere conocer información previa sobre el objeto o el fondo.

- Flujo óptico

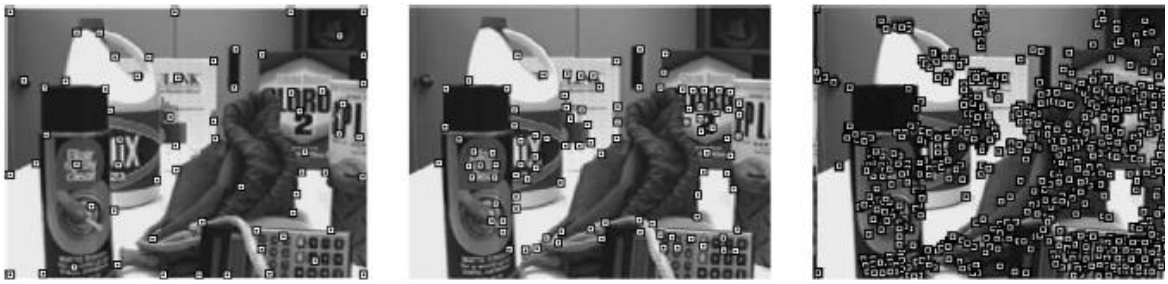
Los métodos basados en flujo óptico solucionan los problemas de la segmentación utilizando las características de los vectores de flujo de objetos en movimiento en el tiempo para detectar las regiones que cambian en una secuencia de imágenes.

El flujo óptico consiste en un campo formado por vectores de desplazamiento que define la traslación de cada píxel en una región. Se calcula mediante la limitación de brillo, que supone que el brillo de los píxeles se mantiene constante en secuencias sucesivas. Las técnicas más conocidas para implementar el flujo óptico son las propuestas por Horn y Schunck o Lucas y Kanade.

- **Detectores de puntos**

Los detectores de puntos se utilizan para encontrar puntos de interés en imágenes que tienen una textura destacada. Una cualidad deseada de los puntos de interés es que sean invariantes a cambios de iluminación y del punto de vista de la cámara.

En la literatura, los métodos de detección de puntos de interés más destacados son el operador de interés de Moravec, el detector de puntos de interés de Harris, el detector de KLT y el detector SIFT (Scale Invariant Feature Transform)



(a) *(b)* *(c)*

Figura 2.4. Detección de puntos de interés, aplicando los operadores Harris (a), KLT(b), y SIFT(c). Imagen tomada de [7]

- **Aprendizaje supervisado**

La detección de objetos puede realizarse aprendiendo automáticamente diferentes vistas de los objetos a partir de una serie de ejemplos. El aprendizaje de diferentes puntos de vista de los objetos exime de la obligación de almacenar un conjunto completo de modelos. Dado un conjunto de ejemplos, los métodos de aprendizaje supervisado generan una función que mapea las entradas a las salidas deseadas.

Categoría	Ejemplos
Modelado de Fondo	Modelo Gaussiano (Gaussian mixture) Modelo no paramétrico Eigen-background
Segmentación	Mean-Shift Contornosactivos Graph-cut Flujo óptico
Detectores de Puntos	Operador de Moravec Operador de Harris Detector KLT Detector SIFT (Scale Invariant Feature Transform)
Aprendizaje Supervisado	Clasificador SVM (Support Vector Machines) Adaptative Boosting

Tabla 2.2. Técnicas de detección de objetos

2.2.2. SEGUIMIENTO

La etapa de seguimiento de objetos pretende generar la trayectoria de un objeto durante el tiempo conociendo su posición en cada secuencia del video, además de proporcionar la región del espacio que ocupa en cada momento. Estas dos operaciones pueden hacerse de manera conjunta o separada.

En cualquier caso, los objetos se representan utilizando diversos modelos de apariencia y formas:

- Puntos. El objeto puede representarse mediante un sólo punto, llamado centroide o baricentro, o mediante múltiples puntos. Esta representación es más adecuada para el seguimiento de objetos de pequeño tamaño.
- Formas geométricas. La forma del objeto se representa como un rectángulo o una elipse. El movimiento del objeto en este tipo de representaciones se modela por traslación, afinidad o trasformación proyectiva (homografía). Aunque las formas geométricas son más adecuadas para representar objetos rígidos, también pueden utilizarse para el seguimiento de objetos no rígidos.
- Contorno y silueta del objeto. La representación del contorno define el borde del objeto, mientras que la región dentro del contorno se denomina silueta. Estas representaciones son útiles para el seguimiento de formas complejas y no rígidas.
- Modelos articulados. Estos modelos están formados por diferentes partes unidas entre sí, cada una formada por elipses. La relación entre cada una de las partes está controlada por modelos cinemáticos.
- Modelo de esqueleto. El esqueleto del objeto puede obtenerse aplicando la transformada del eje medio (*medial axis*) a la silueta del objeto.

Dependiendo del tipo de representación elegido para el objeto, se pueden usar unas u otras formas de desplazamiento. Por ejemplo, si se ha elegido la representación mediante puntos

sólo puede usarse un modelo de traslación, y se ha optado por una representación mediante formas geométricas, son más apropiadas la afinidad y la transformación proyectiva.

- **Seguimiento de puntos**

Los objetos detectados en fotogramas consecutivos se representan mediante puntos, y la asociación de dichos puntos se basa en el estado previo del objeto. Este método requiere de mecanismos externos para detectar objetos en cada fotograma. El principal problema que presenta este tipo de seguimiento es determinar los puntos cuando hay fallos de detección, entradas y salidas de objetos, oclusiones.

El seguimiento de puntos es adecuado cuando tenemos en la imagen objetos de pequeño tamaño, los cuales pueden ser representados por un único punto. Cuando tenemos objetos de mayor tamaño, es necesario el empleo de varios puntos, lo cual se convierte en un problema ante la presencia de varios objetos, ya que es necesario distinguir los objetos entre sí.

Las técnicas basadas en correspondencia de puntos pueden dividirse en métodos deterministas y métodos estadísticos.

- Métodos deterministas

Estas técnicas definen el coste de asociar cada objeto del fotograma anterior a un objeto del fotograma actual, mediante una serie de restricciones (proximidad, velocidad máxima, movimientos suaves…). La solución más eficiente se encuentra empleando un algoritmo para minimizar dicho coste.

Algunos de los algoritmos propuestos en la literatura son el rastreador MGE (desarrollado por Salari) y el rastreador GOA (Veenman).

- Métodos estadísticos

Los métodos estadísticos tienen en cuenta el ruido y las perturbaciones aleatorias que pueden alterar la correcta detección de los objetos de la escena.

Dentro de estos métodos, hay que distinguir entre los que son capaces de funcionar cuando tenemos un único elemento móvil en la escena y los que son útiles con múltiples objetos.

El filtro de Kalman se emplea cuando queremos determinar el estado de un solo objeto y el ruido presente en la imagen tiene una distribución Gaussiana. Para ello hace uso de dos etapas: predicción y corrección. El filtro de Kalman es de gran importancia en el campo del procesado de señal, pero también para el control y el guiado de vehículos o en visión artificial.

Sin embargo, cuando en la escena tenemos más de un objeto hay que asociar a cada uno sus medidas correspondientes, entonces el filtro Kalman ya no es eficiente. También puede darse el caso de que los objetos estén lo suficientemente cerca como para que no sea posible hacer una asociación y una medida correcta.

Existen varias técnicas para asociación de datos que son capaces de solucionar estos problemas, como por ejemplo Joint Probability Data Association Filter (JPDAF) o Multiple Hypothesis Tracking (MHT).

- **Kernel tracking (Seguimiento de núcleo)**

Esta técnica caracteriza los objetos como plantillas rectangulares o elípticas, y el seguimiento se realiza calculando el movimiento de los objetos entre un fotograma y el siguiente. Los tipos de movimiento detectados son la traslación, rotación y afinidad.

Para realizar este tipo de seguimiento se pueden emplear varios tipos de mecanismos, que se diferencian dependiendo de cómo se representan los objetos, el número de objetos que se siguen, cómo se determina el movimiento de los objetos… En este apartado sólo vamos a nombrar algunos de los más significativos, como son el método de mean-shift, que ya vimos en el apartado 2.2.1 de detección de objetos pero esta vez aplicado al seguimiento, y el método KLT (Kande-Lucas-Tomasi). El algoritmo KLT se basa en el concepto de flujo óptico, ya que mediante la generación del campo de vectores de flujo se calcula la traslación y el desplazamiento del objeto contenido en una forma geométrica.

El principal objetivo de los métodos de esta categoría es estimar el movimiento de los objetos. Con la representación de objetos basada en regiones, el cálculo de su movimiento define implícitamente el propio movimiento, la orientación y la forma del objeto. Una de las limitaciones de representar a los objetos mediante formas geométricas simples es que partes de los objetos pueden quedar fuera de la forma definida.

Figura 2.5. Seguimiento con plantilla elíptica
Imágenes tomadas de [22].

- **Seguimiento de la silueta**

Los métodos basados en el seguimiento de la silueta se emplean cuando los objetos tienen formas complejas y no pueden describirse adecuadamente mediante formas geométricas sencillas. El seguimiento se lleva a cabo estimando para cada fotograma la región de la imagen que corresponde con el objeto, empleando para ello un modelo del objeto que se crea a partir

de los fotogramas anteriores. Este tipo de métodos se utilizan cuando es necesario el seguimiento de la región completa del objeto, pero la ventaja más importante del seguimiento de siluetas es su flexibilidad para manejar gran variedad de formas de objetos. Se pueden clasificar en dos categorías:

- Seguimiento del contorno.

La búsqueda del contorno se realiza expandiendo iterativamente el contorno original a su nueva posición en el fotograma actual. Este método requiere que parte del objeto en el fotograma actual se solape con la región del objeto en el fotograma anterior.

El seguimiento por contorno puede realizarse mediante dos diferentes enfoques. El primero emplea distintos estados para definir los objetos en función de su forma y sus parámetros de movimiento, que se actualizan en cada instante de tiempo. El segundo determina el contorno minimizando la energía del contorno usando técnicas de minimización como el método del gradiente.

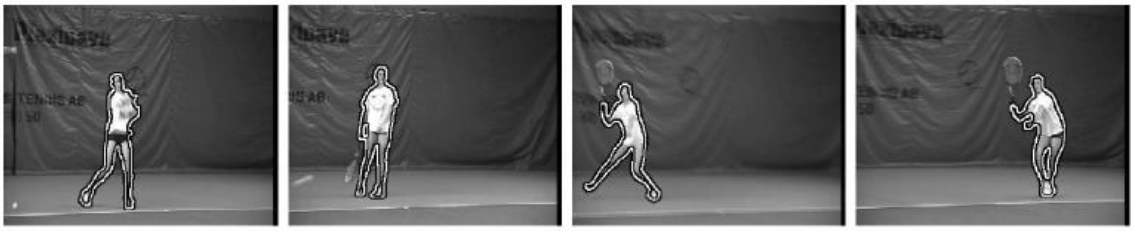


Figura 2.6. Seguimiento de contorno

- Coincidencia de la forma.

Se parte de la silueta del objeto o de su modelo asociado y se buscan coincidencias en el fotograma actual. La búsqueda se realiza calculando la similitud entre el objeto y el modelo generado a partir de la silueta del objeto en el fotograma anterior. En este enfoque, se considera que la silueta sólo se desplaza de un fotograma al siguiente, por lo que no se tienen en cuenta objetos no rígidos.

2.2.3. ANÁLISIS DE COMPORTAMIENTO

Uno de los mayores retos dentro del campo de la visión por computadora y la inteligencia artificial es el entendimiento y el aprendizaje de la conducta a partir de observar actividades en un vídeo. Las investigaciones en este campo se centran principalmente en el desarrollo de métodos para el análisis de datos de vídeo con el objetivo de extraer y procesar información relacionada con el comportamiento de objetos físicos, como por ejemplo personas, en una escena.

Categoría	Ejemplos
Seguimiento de Puntos	
■ Métodos deterministas	Rastreador MGE Rastreador GOA
■ Métodos estadísticos	Filtro de Kalman JPDAF (Joint Probability Data Association Filter) MHT (Multiple Hypothesis Tracking)
Kernel Tracking (Seguimiento de núcleo)	
■ Modelos de apariencia basados en forma	Mean – shift Algoritmo KLT (Kande-Lucas-Tomasi)
Seguimiento de Silueta	
■ Seguimiento de contorno	Métodos variacionales Métodos heurísticos
■ Coincidencia de forma	Hausdorff Transformada de Hough

Tabla 2.3. Técnicas de seguimiento de objetos

En sistemas de videovigilancia automatizados, la detección fiable de comportamiento humano sospechoso o peligroso es un asunto de gran importancia. Un sistema de este tipo normalmente necesita de la combinación eficaz de técnicas de procesado de imágenes y de inteligencia artificial. Las técnicas de procesado de imagen se utilizan para proporcionar características de la imagen de bajo nivel. Las técnicas de inteligencia artificial se emplean para proporcionar decisiones expertas. Se han realizado numerosas investigaciones sobre técnicas de procesado de imagen de bajo nivel como por ejemplo detección, reconocimiento y seguimiento de objetos; sin embargo, han sido pocos los estudios que han dado un entendimiento y una clasificación fiable del comportamiento humano a partir de secuencias de video.

La detección de comportamientos implica el modelado y la clasificación de actividades humanas según ciertas reglas, pero esto no es un proceso sencillo, dada la diversidad y complejidad de los movimientos. Aun así, la idea es dividir los movimientos observados en algunos estados discretos y luego clasificarlos adecuadamente.

Muchos enfoques en este campo del procesado de vídeo incorporan métodos para la detección de eventos específicos. El principal inconveniente de estas técnicas es que son sólo específicas para ciertas aplicaciones pero no pueden aplicarse en otras circunstancias. Otro punto de vista es dividir el procesado en dos etapas:

- Un módulo de procesado de imagen de bajo nivel se emplea para extraer señales visuales y eventos primitivos.
- Esta información se utiliza en un módulo de inteligencia artificial de alto nivel para detectar patrones de comportamiento más complejos.

Dividiendo el problema en dos etapas, se pueden usar técnicas más sencillas e independientes del entorno en cada etapa.

El reto consiste en combinar las técnicas disponibles de reconocimiento de comportamiento para acercarse a la gran diversidad de situaciones que existen en el mundo real. El aprendizaje de patrones de comportamiento puede considerarse como la clasificación de datos variables en el tiempo, como por ejemplo, asociar una secuencia desconocida a un grupo de secuencias de referencia que representan comportamientos comunes o aprendidos. El problema fundamental consiste en aprender las secuencias de referencia a partir de muestras de entrenamiento, y en diseñar los métodos de entrenamiento y de asociación para hacer frente con eficacia a pequeñas variaciones de los datos característicos dentro de cada clase de patrón de movimiento. Los principales métodos para el análisis de comportamiento son los siguientes:

- Hidden Markov Models (HMMs): Un HMM es una herramienta estadística empleada para modelar secuencias caracterizadas por un conjunto de secuencias.
- Dynamic Time Warping (DTW): La DTW (deformación dinámica del tiempo) es una técnica que alinea de manera óptima secuencias de tiempo de longitud variable. Sirve para calcular la semejanza o para encontrar correspondencias entre dos series de tiempo relacionando un patrón de prueba con un patrón de referencia.
- Finite-State Machine (FSM): FSM o autómata de estado finito, es un modelo de comportamiento compuesto por un número finito de estados, transiciones entre estos estados y acciones. Una máquina de estados finitos es un modelo abstracto de una máquina con una memoria primitiva interna.
- Nondeterministic-Finite-State Automaton (NFA): Un NFA o un autómata de estado finito no determinista es una máquina de estado finito en la cual cada par de estados y símbolos de entrada pueden existir varios estados siguientes posibles. Esto lo diferencia de los autómatas finitos deterministas (DFA), en los cuales el siguiente estado posible está unívocamente determinado.
- Time-Delay Neural Network (TDNN): TDNN es una técnica para analizar datos variables en el tiempo. En TDNN, las unidades de retardo se añaden a una red general y estática, y algunos de los valores anteriores de una secuencia variable en el tiempo se utilizan para predecir el próximo valor. A medida que el conjunto de datos disponible aumenta, se está haciendo más énfasis en las redes neuronales para la representación de información temporal. Los métodos basados en TDNN han sido aplicados con éxito en situaciones como reconocimiento de gestos y lectura de labios.

CAPÍTULO 3

ESTADO DEL ARTE: ALGORITMOS DE SEGUIMIENTO

En el capítulo anterior se detallaron diversas técnicas de detección de objetos y seguimiento; entre ellas tenemos los métodos deterministas (Rastreador MGE y rastreador GOA), métodos estadísticos (Filtro de Kalman, *Joint Probability Data Association Filter* y *Multiple Hypothesis Tracking*), entre otros.

En este nuevo capítulo nos proponemos ampliar el estudio de algoritmos de seguimiento, haciendo un repaso a métodos que tienen cierta popularidad dentro del campo del seguimiento de objetos. En concreto, vamos a tratar el filtro alfa-beta, el filtro de Kalman y el filtro de partículas.

3.1 INTRODUCCIÓN

Tradicionalmente, el problema típico de seguimiento consiste en estimar secuencialmente el estado de un sistema dinámico usando una secuencia de medidas ruidosas obtenidas sobre el sistema [15].

El vector de estado de un sistema contiene toda la información relevante necesaria para describir el sistema. Por ejemplo, en aplicaciones de seguimiento la información está relacionada con las características de movimiento del blanco. El vector de medida representa las observaciones ruidosas relacionadas con el vector de estado y, generalmente, es de orden menor que el vector de estado.

Para analizar un sistema dinámico se necesitan dos modelos:

- Modelo de sistema: Describe la evolución del estado en el tiempo.
- Modelo de medida: Relaciona las medidas al estado.

Se asume que estos dos modelos están disponibles en forma “probabilística”. Esta formulación mediante probabilidad y el requisito de actualizar la información, es apropiada para una aproximación Bayesiana, ya que proporciona un marco riguroso. En la aproximación Bayesiana, se intenta construir la función de densidad de probabilidad (*pdf*) posterior, basándose en la

información disponible incluidas las medidas recibidas. Dado que la *pdf* reúne toda la información estadística disponible, se puede considerar como la solución completa al problema de estimación.

En muchos problemas se usan filtros recursivos, donde los datos recibidos se procesan secuencialmente en lugar de en bloque. Consisten en dos etapas:

- Predicción: Se usa el modelo del sistema para predecir la siguiente *pdf*. Como el estado tiene imperfecciones (ruido) la predicción deforma la *pdf* resultante.
- Actualización: Se usa la última medida para modificar la predicción obtenida. Se consigue usando el teorema de Bayes, ya que es el mecanismo para actualizar el estado con la información procedente de los nuevos datos.

La propagación recursiva de la *pdf* entre estados requiere el almacenamiento completo de la *pdf*, que es equivalente a un vector infinito. Dado que sólo en un conjunto reducido de casos la *pdf* posterior puede caracterizarse completa y exactamente, los distintos algoritmos existentes dentro de la categoría de pueden clasificarse en dos grupos: los que proporcionan una solución óptima y los que producen una solución aproximada [15].

Algoritmos óptimos

1. Filtro de Kalman: es la solución óptima cuando se asume que la *pdf* posterior es Gaussiana y está caracterizada exacta y completamente por dos parámetros, su media y covarianza.
2. Métodos Grid-based: son la mejor solución si el espacio de estados es discreto y tiene un número finito de estados.
3. Filtros de Beneš y Daum.

Algoritmos sub-óptimos

En muchas aplicaciones prácticas, los filtros no lineales óptimos no pueden aplicarse, por ejemplo, cuando no puede asumirse estados gaussianos. En casos como estos es necesario aplicar aproximaciones:

1. Aproximaciones analíticas: Filtro de Kalman Extendido (EKF)
2. aproximaciones de muestreo: Filtros de partículas, realizan estimación de Monte-Carlo secuencial (SMC).
3. Aproximaciones numéricas

En los siguientes apartados vamos a repasar cómo se formulan algunos de los algoritmos enunciados anteriormente, haciendo especial énfasis en el filtro de Kalman; y cómo se aplican estos filtros a los problemas de seguimiento.

3.2 FILTRO DE KALMAN

3.2.1 Introducción

Desde el punto de vista teórico, el filtro de Kalman es un estimador para lo que se llama el *problema lineal cuadrático*, el cual es el problema de estimar el "estado" instantáneo de un sistema dinámico lineal perturbado por ruido blanco – usando medidas linealmente relacionadas con el estado, pero con ruido blanco añadido. El modelo matemático usado en la obtención del filtro de Kalman es una representación razonable para muchos problemas de interés práctico, incluyendo *problemas de control* así como *problemas de estimación*, por lo que el modelo del filtro de Kalman se usa para el análisis de medidas y problemas de estimación. El filtro fue desarrollado por Rudolf Emil Kalman a finales de la década de 1950, y supuso un gran acontecimiento ya que muchos logros técnicos conseguidos en años sucesivos no hubieran sido posibles sin su descubrimiento; por ejemplo, permitió los avances en tecnología espacial para la navegación eficiente de naves. Los principales usos del filtro de Kalman se encuentran en los sistemas de control modernos, en el seguimiento y navegación de todo tipo de vehículos, etc.

El desarrollo del filtro de Kalman no surge como un hecho aislado, sino que su descubrimiento viene del trabajo previo en otros métodos de estimación, como el método de los mínimos cuadrados y el filtro de Wiener-Kolmogorov. El *método de mínimos cuadrados* fue el primer método de estimación "óptimo". Fue descubierto por Gauss (entre otros) a finales del siglo XVIII, y a día de hoy sigue teniendo un amplio uso. El *filtro de Wiener-Kolmogorov* fue desarrollado en los años 40 por Norbert Wiener (usando un modelo en tiempo continuo) y por Andrei Kolmogorov (usando un modelo en tiempo discreto), trabajando de forma independiente. Es un método de estimación estadística; estima el estado de un proceso dinámico de manera que minimiza el error de estimación cuadrático medio. Toma ventaja del conocimiento estadístico sobre procesos aleatorios en términos de su densidad de potencia espectral en el domino de la frecuencia.

El modelo de "espacio de estados" de un proceso dinámico usa ecuaciones diferenciales para representar tanto los fenómenos deterministas como los aleatorios. Las variables de estado de este modelo son las variables de interés y los procesos aleatorios se caracterizan en términos de sus propiedades estadísticas en el dominio temporal, en lugar del dominio de la frecuencia. El filtro de Kalman se obtuvo como solución al problema del filtrado Wiener usando el modelo de espacio de estados para procesos aleatorios y dinámicos. El resultado es más fácil de obtener (y de usar) que el filtro de Wiener-Kolmogorov [14].

3.2.2 Funcionamiento básico

El filtro de Kalman realiza estimaciones del vector de estado del objetivo a seguir para los siguientes instantes de tiempo, utilizando un tipo de control realimentado: el filtro estima el estado del proceso en cierto tiempo y entonces obtiene realimentación en forma de medidas

(con ruido). Por supuesto existe incertidumbre en estas estimaciones. Esta incertidumbre se puede contemplar como un intervalo de confianza, que en el filtro de Kalman siempre se representa como un elipsoide.

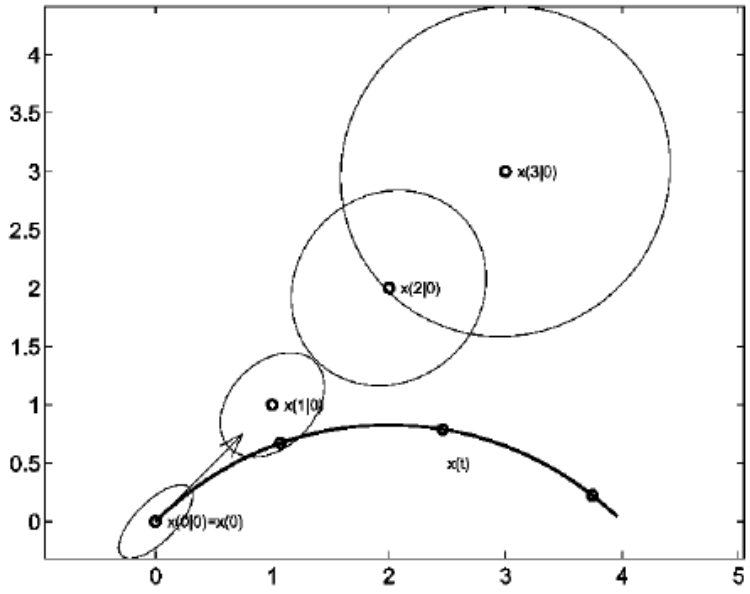


Figura 3.1. Evolución de la región de incertidumbre sin actualización

La figura 3.1 muestra cómo el modelo predice las posiciones futuras del objeto, dado el vector de estado en el tiempo 0, y cómo la región de incertidumbre crece con el tiempo. En la siguiente figura (Fig. 3.2.) se muestra cómo el filtro tiene en cuenta la primera medida corrigiendo el vector de estado y reduciendo la región de incertidumbre [23].

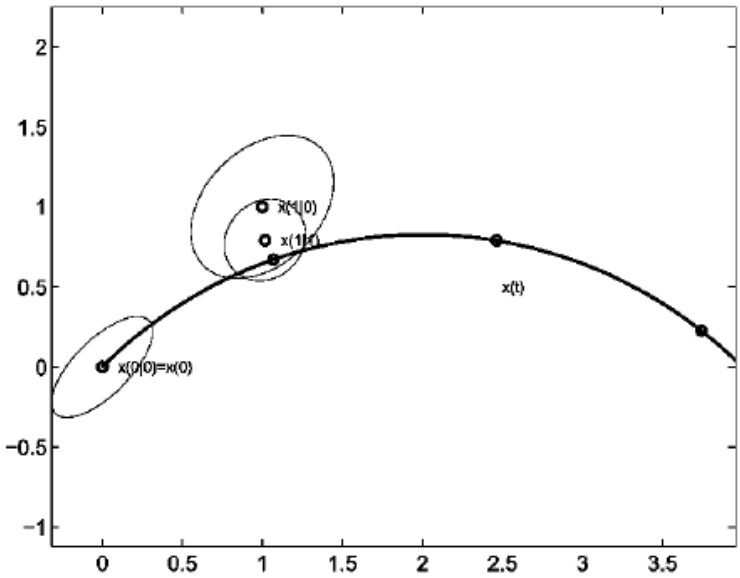


Figura 3.2. Evolución de la región de incertidumbre con actualización

Las ecuaciones del filtro de Kalman se engloban en dos grupos: ecuaciones de *actualización de tiempo* y ecuaciones de *actualización de medida*. Las ecuaciones de actualización de tiempo son responsables de propagar (en tiempo) el estado actual y las estimaciones de la covarianza del error para obtener estimaciones *a priori* para el siguiente instante de tiempo. Las ecuaciones de actualización de medida son responsables de la realimentación – incorporar una nueva medida en la estimación *a priori* para obtener una estimación *a posteriori* mejorada [24].

Las ecuaciones de actualización de tiempo también pueden verse como ecuaciones de predicción, mientras que las ecuaciones de actualización de medida pueden considerarse como ecuaciones de corrección. En efecto, el algoritmo de estimación final se asemeja a un algoritmo predictor-corrector para resolver problemas numéricos.

Ecuaciones de actualización de tiempo del filtro de Kalman en tiempo discreto

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

Las ecuaciones de actualización de tiempo proyectan las estimaciones de covarianza y de estado del instante de tiempo $k-1$ al tiempo k.

Ecuaciones de actualización de medida del filtro de Kalman en tiempo discreto

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

El primer paso durante la actualización de las medidas es calcular la ganancia de Kalman K_k. Después se mide propiamente el proceso para obtener z_k, y entonces generar la estimación del estado *a posteriori* incorporando la medida. El último paso es obtener la covarianza *a posteriori*.

Después de cada actualización en tiempo y medida, el proceso se repite con la previa estimación *a posteriori* usada para predecir las nuevas estimaciones *a priori*. Esta naturaleza recursiva es una de las características más atractivas del filtro de Kalman - su implementación práctica es mucho más factible que, por ejemplo, el filtro Wiener que está diseñado para operar en todos los datos directamente para cada estimación. El filtro de Kalman por su parte, condiciona de forma recursiva la estimación actual en todas las medidas pasadas.

La figura 3.3 representa de forma gráfica el esquema de funcionamiento del filtro de Kalman.

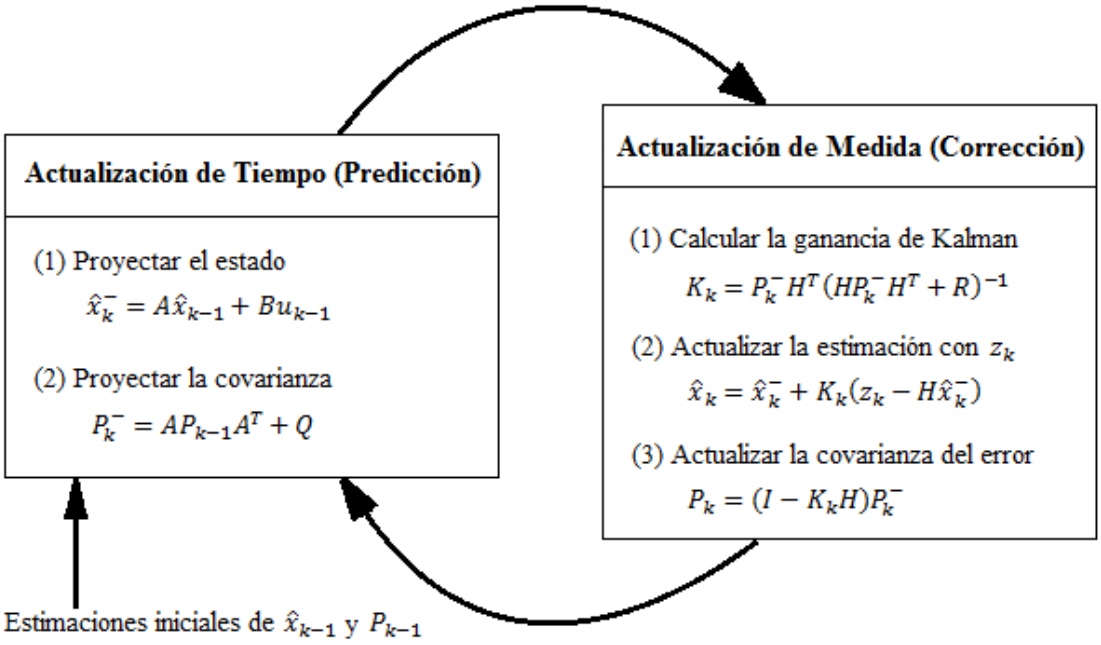


Figura 3.3. Diagrama de funcionamiento del filtro de Kalman

Un modelo de espacio de estado en tiempo variable para el filtro de Kalman puede ser el siguiente [23]:

$$x_{t+1} = A_t x_t + B_t u_t + w_t, \quad Cov(w_t) = Q_t$$

$$z_t = H_t x_t + v_t, \qquad Cov(v_t) = R_t$$

En las ecuaciones anteriores, z_t es la señal medida, A, B, H son matrices conocidas y x_t es el vector de estado desconocido. Hay tres entadas al sistema: el vector de control observable y controlable u_t, el ruido del proceso no observable w_t y el ruido en la medida v_t.

La matriz A es el modelo determinista y describe cómo simular o cómo evoluciona con el tiempo el vector de estado.

La matriz B es el modelo de control de entrada. El término estocástico en la ecuación de estado $B_v u_t$, decide cómo crece el intervalo de confianza. Un caso extremo es $Q \to \infty$, cuando cualquier cosa puede pasar en un intervalo de tiempo y el elipsoide se hace infinitamente grande. El otro caso extremo se corresponde con $Q \to 0$, cuando el elipsoide no crece nunca.

La parte determinista de la ecuación de medida $z_t = H_t x_t$ indica al filtro en qué dirección una medida puede afectar a la estimación.

La covarianza R del ruido de la medida v_t describe cómo de fiable es la información medida. Cuando $R = 0$ la medida es exacta y coincide con la posición real. Si $R \to \infty$, la medida no es útil y debe ser descartada.

3.2.3 Desarrollo del Filtro de Kalman

Para hacer un desarrollo completo del filtro de Kalman se siguen las pautas indicadas en [14] en el que a la hora de establecer la ecuación de estado se prescinde de la matriz B, no tener relevancia en el mecanismo de obtención de las fórmulas. De esta manera, el desarrollo del filtro de Kalman se realiza de la siguiente manera:

Resumen de las variables utilizadas:

x - Vector de estado del sistema dinámico.

z - Vector (o número) de los valores medidos.

w – Ruido del proceso.

v – Ruido de la medida o ruido en los valores medidos.

A - Matriz de transición de estados del sistema dinámico (en otras bibliografías se representa como Φ)

H - Matriz de sensibilidad de la medida. Define la relación entre el estado del sistema dinámico y las medidas que pueden realizarse (en otra bibliografía se representa por C).

$\overline{K}$ - Matriz de la ganancia de Kalman.

P - Matriz de covarianza de la incertidumbre de estimación del estado.

Q - Matriz de covarianza del ruido del proceso

R - Matriz de covarianza de la incertidumbre en la medida.

$\hat{x}$ - Vector de estado estimado.

$\tilde{x}$ - Vector de error en la estimación.

Problema de estimación: La estimación del siguiente estado del sistema se realiza mediante la ecuación del modelo del sistema, que representa la evolución del sistema en el tiempo, calculado a partir del estado anterior de la forma

$$x_k = A_{k-1}x_{k-1} + w_{k-1} \qquad (3.1)$$

Donde A es la matriz de transición de estado y w_k es el ruido del proceso.

Problema de actualización de las medidas: Suponer que se obtiene una medida en el tiempo t_k y que la información que proporciona se utiliza para actualizar la estimación del estado x de un sistema estocástico en el instante t_k. Se asume que la medida está linealmente relacionada con el estado mediante una ecuación de la forma

$$z_k = Hx_k + v_k \qquad (3.2)$$

Donde H es la matriz de sensibilidad de la medida y v_k es el ruido de la medida.

Estimador en forma lineal: la estimación lineal óptima es equivalente al estimado óptimo general (no lineal) si las variables x y z son conjuntamente Gaussianas. Por tanto, basta con buscar una estimación actualizada $\hat{x}_k(+)$ – basada en la observación z_k–que sea una función linear de la estimación a priori y de la medida z:

$$\hat{x}_k = K_k^1 \hat{x}_k(-) + \overline{K}_k z_k \qquad (3.3)$$

Donde $\hat{x}_k(-)$ es la estimación a priori de x_k y $\hat{x}_k(+)$ es el valor a posteriori de la estimación.

Problema de optimización: las matrices K_k^1 y $\overline{K}_k$ son todavía desconocidas. Buscamos valores tales que la nueva estimación $\hat{x}_k(+)$ satisfaga el principio de ortogonalidad siguiente:

$$E\langle[x_k - \hat{x}_k(+)]z_i^T\rangle = 0, \;\; i = 1, 2, \dots, k-1 \qquad (3.4)$$

$$E\langle[x_k - \hat{x}_k(+)]z_k^T\rangle = 0 \qquad (3.5)$$

Si se sustituye en la fórmula anterior x_k por la ecuación 3.1 y $\hat{x}_k$ por la ecuación 3.3 desarrollada anteriormente, observamos que en los datos $z_1, \dots, z_k$ no interviene el término de ruido w_k. Por tanto, como las secuencias aleatorias w_k y v_k son incorreladas, se sigue que $Ew_k z_i^T = 0$ para $1 \leq i \leq k$.

Usando este resultado, se obtiene la siguiente relación:

$$E\langle[A_{k-1}x_{k-1} + w_{k-1} - K_k^1 \hat{x}_k(-) + \overline{K}_k z_k]z_i^T\rangle = 0, \qquad i = 1, 2, \dots, k-1 \qquad (3.6)$$

Pero como $z_k = H_k x_k + v_k$, la ecuación anterior puede reescribirse de la siguiente forma:

$$E\left[A_{k-1}x_{k-1} - K_k^1 \hat{x}_k(-) - \overline{K}_k H_k x_k - \overline{K}_k v_k\right]z_i^T = 0, \qquad i = 1, 2, \dots, k-1 \qquad (3.7)$$

También sabemos que las ecuaciones 3.4 y 3.5 se mantienen para el instante anterior, esto es:

$E\langle[x_{k-1} - \hat{x}_{k-1}(+)]z_i^T\rangle = 0, \;\; i = 1, 2, \dots, k-1$ y $E\langle v_k z_i^T\rangle = 0, \;\; i = 1, \dots, k-1$

Entonces la ecuación 3.7 se reduce:

$$A_{k-1}Ex_{k-1}z_i^T - K_k^1 E\hat{x}_k(-)z_i^T - \overline{K}_k H_k \mathrm{A}_{k-1}Ex_{k-1}z_i^T - \overline{K}_k Ev_k z_i^T = 0$$

$$A_{k-1}Ex_{k-1}z_i^T - K_k^1 E\hat{x}_k(-)z_i^T - \overline{K}_k H_k \mathrm{A}_{k-1}Ex_{k-1}z_i^T = 0$$

$$E\langle\left[x_k - \overline{K}_k H_k x_k - K_k^1 x_k\right] - K_k^1(\hat{x}_k(-) - x_k)\rangle z_i^T = 0$$

$$\left[I - K_k^1 - \overline{K}_k H_k\right]Ex_k z_i^T = 0 \qquad (3.8)$$

La ecuación 3.8 se satisface para cualquier x_k si

$$K_k^1 = I - \overline{K}_k H_k \qquad (3.9)$$

Claramente, esta elección de K_k^1 causa que la ecuación 3.3 satisfaga una parte de la condición dada por la ecuación 3.4 La elección de $\overline{K}_k$ es tal que se satisface la ecuación 3.5.

Los errores se expresan de forma:

$$\tilde{x}_k(+) \triangleq \hat{x}_k(+) - x_k \qquad (3.10)$$

$$\tilde{x}_k(-) \triangleq \hat{x}_k(-) - x_k \qquad (3.11)$$

$$\tilde{z}_k \triangleq \hat{z}_k(+) - \hat{z}_k = H_k\hat{x}_k(-) - z_k \qquad (3.12)$$

Los vectores $\tilde{x}_k(+)$ y $\tilde{x}_k(-)$ son los errores de estimación antes y después de la actualización, respectivamente.

El parámetro $\hat{x}_k$ depende linealmente de x_k, que depende linealmente de z_k. Por tanto, de la ecuación 3.5:

$$E[x_k - \hat{x}_k(+)]z_k^T(-) = 0 \qquad (3.13)$$

Y también, restando la ecuación 3.5 de la ecuación 3.13:

$$E[x_k - \hat{x}_k(+)]\tilde{z}_k^T = 0 \qquad (3.14)$$

Se sustituye x_k, $\hat{x}_k(+)$ y $\tilde{z}_k$ de las ecuaciones 3.1, 3.3 y 3.12, entonces:

$$E\left[A_{k-1}x_{k-1} + w_{k-1} - K_k^1\hat{x}_k(-) - \overline{K}_k z_k\right][H_k\hat{x}_k(-) - z_k]^T = 0$$

Sin embargo, por la estructura del sistema:

$$Ew_k z_k^T = Ew_k\hat{x}_k^T(+) = 0$$

$$E\left[A_{k-1}x_{k-1} - K_k^1\hat{x}_k(-) - \overline{K}_k z_k\right][H_k\hat{x}_k(-) - z_k]^T = 0$$

Sustituyendo por K_k^1, z_k y $\tilde{x}_k(-)$ y usando el hecho de que $E\tilde{x}_k(-)v_k^T = 0$, este último resultado puede modificarse de la siguiente manera:

$$0 = E\langle[A_{k-1}x_{k-1} - \hat{x}_k(-) + \overline{K}_k H_k\hat{x}_k(-) - \overline{K}_k H_k x_k - \overline{K}_k v_k][H_k\hat{x}_k(-) - H_k x_k - v_k]^T\rangle$$

$$= E\langle\left[\left(x_k - \hat{x}_k(-)\right) - \overline{K}_k H_k(x_k - \hat{x}_k(-)) - \overline{K}_k v_k\right][H_k\tilde{x}_k(-) - v_k]^T\rangle$$

$$= E\langle[(-\hat{x}_k(-) + \overline{K}_k H_k\hat{x}_k(-) - \overline{K}_k v_k][H_k\tilde{x}_k(-) - v_k]^T\rangle$$

Por definición, la covarianza a priori (la matriz de covarianza del error antes de la actualización) es:

$$P_k(-) = E\langle\tilde{x}_k(-)\tilde{x}_k^T(-)\rangle$$

Satisface la ecuación:

$$[I - \overline{K}_k H_k]P_k(-)H_k^T - \overline{K}_k R_k = 0$$

Y por tanto, la ganancia puede expresarse como:

$$\overline{K}_k = P_k(-)H_k^T[H_k P_k(-)H_k^T + R_k]^T \qquad (3.15)$$

Que es la solución que buscamos para la ganancia como una función de la covarianza a priori.

Se puede derivar una fórmula similar para la covarianza a posteriori (la matriz de covarianza del error después de la actualización), que se define como:

$$P_k(+) = E\langle[\tilde{x}_k(+)\tilde{x}_k^T(+)]\rangle \qquad (3.16)$$

Sustituyendo la ecuación 3.9 en la ecuación 3.3, se obtiene las ecuaciones:

$$\hat{x}_k(+) = (I - \overline{K}_k H_k)\hat{x}_k(-) + \overline{K}_k z_k,$$

$$\hat{x}_k(+) = \hat{x}_k(-) + \overline{K}_k[z_k - H_k\hat{x}_k(-)]. \qquad (3.17)$$

Restando x_k de ambos lados de la última ecuación, se obtiene las ecuaciones:

$$\hat{x}_k(+) - x_k = \hat{x}_k(-) + \overline{K}_k H_k x_k + \overline{K}_k v_k - \overline{K}_k H_k\hat{x}_k(-) - x_k,$$

$$\tilde{x}_k(+) = \tilde{x}_k(-) - \overline{K}_k H_k\tilde{x}_k(-) + \overline{K}_k v_k,$$

$$\tilde{x}_k(+) = (I - \overline{K}_k H_k)\tilde{x}_k(-) + \overline{K}_k v_k \qquad (3.18)$$

Sustituyendo la ecuación 3.18 en la ecuación 3.16 y teniendo en cuenta que $E\tilde{x}_k(-)v_k^T = 0$, se obtiene:

$$\begin{aligned} P_k(+) &= E\{[I - \overline{K}_k H_k])\tilde{x}_k(-)\tilde{x}_k^T[I - \overline{K}_k H_k]^T + \overline{K}_k v_k v_k^T\overline{K}_k^T\} \\ &= (I - \overline{K}_k H_k)P_k(-)(I - \overline{K}_k H_k)^T + \overline{K}_k R_k\overline{K}_k^T \qquad (3.19) \end{aligned}$$

Esta última ecuación es la llamada “forma Joseph” de la ecuación de actualización de la covarianza derivada por P. D. Joseph. Sustituyendo por $\overline{K}_k$ de la ecuación 3.15, se puede expresar en los siguientes términos:

$$\begin{aligned} P_k(+) &= P_k(-) - \overline{K}_k H_k P_k(-) - P_k(-)H_k^T\overline{K}_k^T + \overline{K}_k H_k P_k(-)H_k^T\overline{K}_k^T + \overline{K}_k R_k\overline{K}_k^T \\ &= (I - \overline{K}_k H_k)P_k(-) - P_k(-)H_k^T\overline{K}_k^T + \underbrace{\overline{K}_k^T\left(H_k P_k(-)H_k^T + R_k\right)}_{P_k(-)H_k^T}\overline{K}_k^T \\ &= (I - \overline{K}_k H_k)P_k(-) \qquad (3.20) \end{aligned}$$

La última de las expresiones es la más usada en computación. Esto implementa el efecto que el condicionamiento en la medida tiene en la matriz de covarianza de incertidumbre de la estimación.

La extrapolación de la covarianza del error modela los efectos del tiempo en la matriz de covarianza de incertidumbre de la estimación, el cual se refleja en los valores a priori de la covarianza y las estimaciones de los estados,

$$P_k(-) = E[\tilde{x}_k(-)\tilde{x}_k^T(-)]$$

$$\hat{x}_k(-) = A_{k-1}\hat{x}_{k-1}(+) \qquad (3.21)$$

Se resta x_k de ambos lados de la última ecuación para obtener las ecuaciones

$$\hat{x}_k(-) - x_k = A_{k-1}\hat{x}_{k-1}(+) - x_k$$

$$\tilde{x}_k(-) = A_{k-1}[\hat{x}_{k-1}(+) - x_{k-1}] - w_{k-1} = A_{k-1}\tilde{x}_k(+) - w_{k-1}$$

Para la propagación del error de estimación $\tilde{x}$. Se multiplica por $\tilde{x}_k^T(-)$ en ambos lados de la ecuación y se extraen los valores esperados. Se usa el hecho de que $E\tilde{x}_{k-1}w_{k-1}^T = 0$para obtener los resultados

$$\begin{aligned} P_k(-) &\stackrel{\text{def}}{=} E\left[\tilde{x}_k(-)\tilde{x}_k^T(-)\right] \\ &= A_{k-1}E\left[\tilde{x}_{k-1}(+)\tilde{x}_{k-1}^T(+)\right]A_{k-1}^T + E\left[w_{k-1}w_{k-1}^T\right] \\ &= A_{k-1}\mathrm{P}_{k-1}^{(+)}A_{k-1}^T + \mathrm{Q}_{k-1} \qquad (3.22) \end{aligned}$$

Que da el valor a priori de la matriz de covarianza de la incertidumbre de la estimación como función del valor a posteriori previo.

3.2.4 Parámetros del filtro y su afinación

En la implementación del filtro, la covarianza del ruido de la medida R normalmente se mide antes de la operación del filtro. Medir la covarianza R es práctico (posible) porque necesitamos ser capaces de medir el proceso de todas formas, por tanto deberíamos ser capaces de tomar medidas de muestra para determinar la varianza del ruido [24].

La determinación del ruido del proceso Q es más difícil porque normalmente no tenemos la habilidad de observar directamente el proceso que estamos estimando. En ocasiones, un modelo del proceso relativamente simple puede producir resultados aceptables si se introduce suficiente incertidumbre en el proceso mediante la selección de Q. En este caso, se espera que las medidas del proceso sean fiables.

En cualquier caso, tengamos o no una base para elegir los parámetros, muchas veces un mejor rendimiento del filtro se puede conseguir *afinando* los parámetros del filtro R y Q. Este ajuste se realiza *off-line*, frecuentemente con la ayuda de otro filtro de Kalman en un proceso llamado normalmente *identificación del sistema*. En las circunstancias en las que R y Q son constantes, tanto la covarianza P_k como la ganancia de Kalman K_k se estabilizarán rápidamente y luego se mantendrán constantes. En este caso, los parámetros pueden pre-calcularse ejecutando el filtro *off-line* o determinando el valor en estado estable de P_k.

Sin embargo, es frecuente el caso en el que el error en la medida no permanece constante. Por ejemplo, en el avistamiento de balizas el error en la medida de balizas cercanas será menor que en balizas lejanas. Además, el ruido del proceso Q puede cambiar dinámicamente durante el funcionamiento del filtro – convirtiéndose en Q_k – para ajustarse a diferentes movimientos.

En estas situaciones, Q_k puede elegirse para tener en cuenta tanto la incertidumbre del movimiento del blanco como la incertidumbre en el modelo.

3.2.5 Resumen de las ecuaciones del filtro de Kalman en tiempo discreto

Los pasos básicos del proceso de cálculo del estimador de Kalman en tiempo discreto son:

1. Calcular $P_k(-)$ usando $P_{k-1}(+), A_{k-1}$ y Q_{k-1}.
2. Calcular $\overline{K}_k$ usando $P_k(-)$ (obtenido en el paso 1), H_k y R_k.
3. Calcular $P_k(+)$ usando $\overline{K}_k$ (obtenido en el paso 2) y $P_k(-)$ (paso 1).
4. Calcular sucesivos valores de $\hat{x}_k(+)$ recursivamente usando los valores calculados de $\overline{K}_k$, la estimación inicial $\hat{x}_0$ y los datos de entrada z_k.

Modelo del sistema dinámico:	$x_k = A_{k-1}x_{k-1} + w_{k-1}, \quad w_k \sim \mathcal{N}(0, Q_k)$
Modelo de medida:	$z_k = H_k x_k + v_k, \quad v_k \sim \mathcal{N}(0, R_k)$
Condiciones iniciales:	$E\langle x_0 \rangle = \hat{x}_0, E\langle \tilde{x}_0 \tilde{x}_0^T \rangle = P_0$
Supuesto de independencia:	$E\langle w_k v_j^T \rangle = 0$para todo *k y j*
Extrapolación de la estimación del estado (3.21):	$\hat{x}_k = A_{k-1}\hat{x}_{k-1}(+)$
Extrapolación de la covarianza del error (3.22):	$P_k(-) = A_{k-1}P_{k-1}(+)A_{k-1}^T + Q_{k-1}$
Actualización de la estimación del estado (3.17):	$\hat{x}_k(+) = \hat{x}_k(-) + \overline{K}_k[z_k - H_k\hat{x}_k(-)]$
Actualización de la covarianza del error (3.20):	$P_k(+) = [I - \overline{K}_k H_k]P_k(-)$
Matriz de la ganancia de Kalman (3.15):	$\overline{K}_k = P_k(-)C_k^T[C_k P_k(-)C_k^T + R_k]^{-1}$

Tabla 3.1. Resumen de ecuaciones del filtro de Kalman

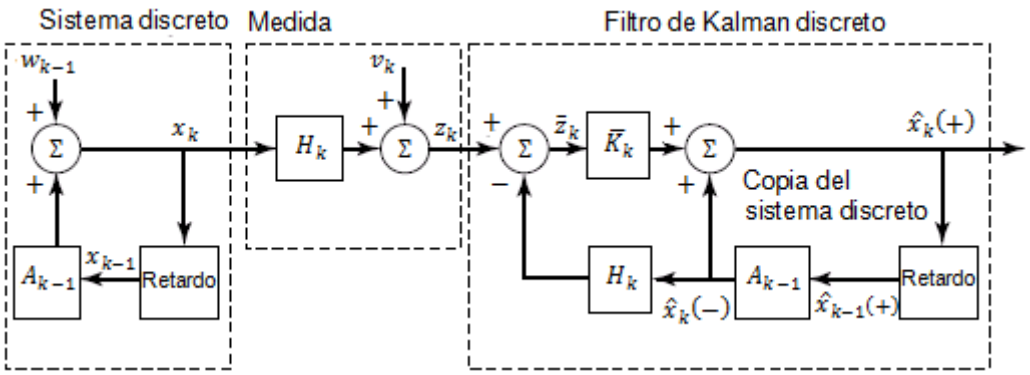


Figura 3.4. Diagrama de bloques del sistema

La relación entre el filtro y el sistema se ilustra en el diagrama de bloques de la figura 3.4.

Referente a las contraprestaciones del diseño, en la actualización de la matriz de covarianza (pasos 1 y 3) debe comprobarse que sea simétrica y definida positiva. Un fallo en cumplir alguna de las condiciones indica que algo está incorrecto – ya sea un bug del programa o un problema de un número mal condicionado *"ill conditioning"*. Para solucionar este último problema, la $P_k(+)$ se puede expresar en la "forma Joseph", como en la ecuación 3.19:

$$P_k(+) = (I - \overline{K}_k C_k) P_k(-) (I - \overline{K}_k C_k)^T + \overline{K}_k R_k \overline{K}_k^T$$

Hay que destacar que la parte derecha de esta ecuación es la suma de dos matrices simétricas. La primera de ellas es definida positiva y la segunda es definida no negativa, haciendo de este modo $P_k(+)$ una matriz definida positiva.

La figura 3.5 muestra una secuencia de tiempo típica de los valores asumidos por la *i*-ésima componente del vector de estado estimado (representado con círculos negros) y su correspondiente varianza de incertidumbre de la estimación (círculos blancos). Las flechas muestran los valores sucesivos asumidos por las variables, con la anotación entre paréntesis indicando qué variables de entrada definen las transiciones indicadas. Cada variable asume dos valores distintos en cada instante de tiempo; su valor a priori correspondiente al valor antes de que se utilice la información de la medida, y su valor a posteriori correspondiente al valor después de que la información se use.

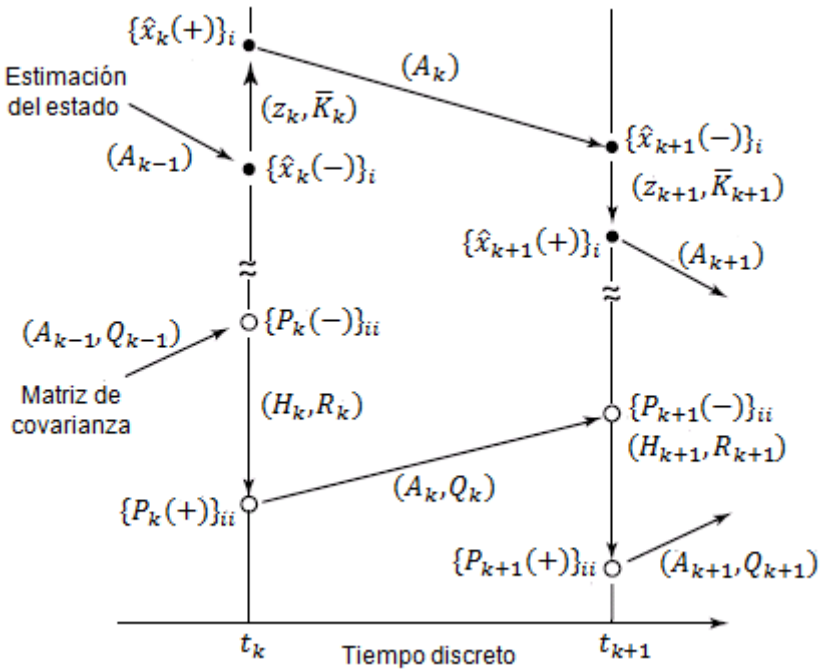


Figura 3.5. Secuencia de valores de las variables del filtro en tiempo discreto

3.3 FILTRO ALFA BETA

Como hemos indicado en el apartado anterior, el filtro de Kalman produce una estimación óptima del estado del objetivo dado el modelo de movimiento y una secuencia de medidas con ruido blanco Gaussiano. Sin embargo, la carga computacional que supone mantener el filtro puede limitar su uso en caso de dispositivos poco potentes o cuando se estén siguiendo varios objetivos simultáneamente.

En estos casos, una solución ampliamente utilizada para reducir la carga computacional es el uso de una ganancia de filtro aproximada, en lugar de la ganancia de Kalman óptima. Cuando esta nueva ganancia se basa en las ganancias del estado estable del filtro de Kalman o en una aproximación, los filtros resultantes se llaman: filtro alfa para el seguimiento de la posición, filtro alfa-beta para el seguimiento de posición y velocidad, filtro alfa-beta-gamma para el seguimiento de posición, velocidad y aceleración.

3.3.1 Ecuaciones del filtro

Consideramos un sistema mecánico, para el cual los dos estados del filtro son la posición y la velocidad. Como se indicó anteriormente, tenemos los modelos de predicción y actualización para el filtro:

Predicción: Asumiendo que la velocidad se mantiene aproximadamente constante durante el pequeño intervalo de tiempo ΔT entre medidas, el estado de posición se proyecta hacia adelante para predecir su valor en el siguiente instante de muestreo k. Dado que la variable de velocidad v se supone constante, su valor planeado para el siguiente tiempo de muestreo es igual al valor actual.

$$x_p[k+1] = x[k] + T \cdot v[k]$$

$$v_p[k+1] = v[k]$$

Debido al ruido y otros parámetros no incluidos en el modelo, la predicción de la posición para un instante $x_p[k]$y el valor de posición realmente medido $z[k]$, resultan en valores diferentes. La diferencia obtenida entre ambos es el residuo $R[k]$.

$z[k] = H \cdot x[k] + n[k]; \quad H = [1\ 0]$; $n[k]$ es el ruido en la medida, modelado como un ruido blanco de media nula, con varianza $\sigma^2 n$.

$$R[k] = z[k] - x_p[k]$$

Actualización: el filtro alfa-beta usa la constante *alfa* para corregir la posición estimada y *beta* para corregir la estimación de velocidad.

$$x[k+1] = x_p[k+1] + \alpha \cdot (z[k] - x_p[k+1])$$

$$v[k+1] = v_p[k+1] + \frac{\beta}{T} \cdot (z[k] - x_p[k])$$

De manera compacta, como un único sistema que incluya las ecuaciones de predicción y actualización, las ecuaciones anteriores pueden expresarse de la siguiente manera:

$$\begin{bmatrix} x_f(k+1) \\ v_f(k+1) \end{bmatrix} = \begin{bmatrix} 1-\alpha & (1-\alpha)T \\ \frac{-\beta}{T_m} & 1-\beta \end{bmatrix} \begin{bmatrix} x_f(k) \\ v_f(k) \end{bmatrix} + \begin{bmatrix} \alpha \\ \frac{\beta}{T} \end{bmatrix} \begin{bmatrix} x_m(k+1) \\ v_m(k+1) \end{bmatrix}$$

El vector de estado filtrado $[x_f(k+1), v_f(k+1)]^T$consiste en la salida final del sistema donde las predicciones han sido ya actualizadas con las medidas, conteniendo los estimadores de posición $x_f(k)$ y la velocidad $v_f(k)$ filtrados del móvil, en la coordenada de interés, actualizados en el instante *k* con las medidas recibidas en dicho instante.

El vector $[x_m(k+1), v_m(k+1)]^T$contiene las medidas de posición $x_m(k+1)$ y la velocidad $v_m(k+1)$, en dicha coordenada, captadas por el sensor en el instante *k+1= k*+T. Dicho vector se corresponde con el vector $z[k]$ según la notación utilizada al expresar el filtro en las dos etapas de predicción y actualización. El intervalo T representa el intervalo de tiempo transcurrido entre la recepción de dos medidas consecutivas tomadas por el sensor sobre el mismo móvil.

Teniendo en cuenta que sólo consideramos que se obtienen medidas de la posición del objeto, y no de la velocidad, el sistema queda de la siguiente forma:

$$\begin{bmatrix} x_f(k+1) \\ v_f(k+1) \end{bmatrix} = \begin{bmatrix} 1-\alpha & (1-\alpha)T \\ \frac{-\beta}{T} & 1-\beta \end{bmatrix} \begin{bmatrix} x_f(k) \\ v_f(k) \end{bmatrix} + \begin{bmatrix} \alpha \\ \frac{\beta}{T} \end{bmatrix} x_m(k+1)$$

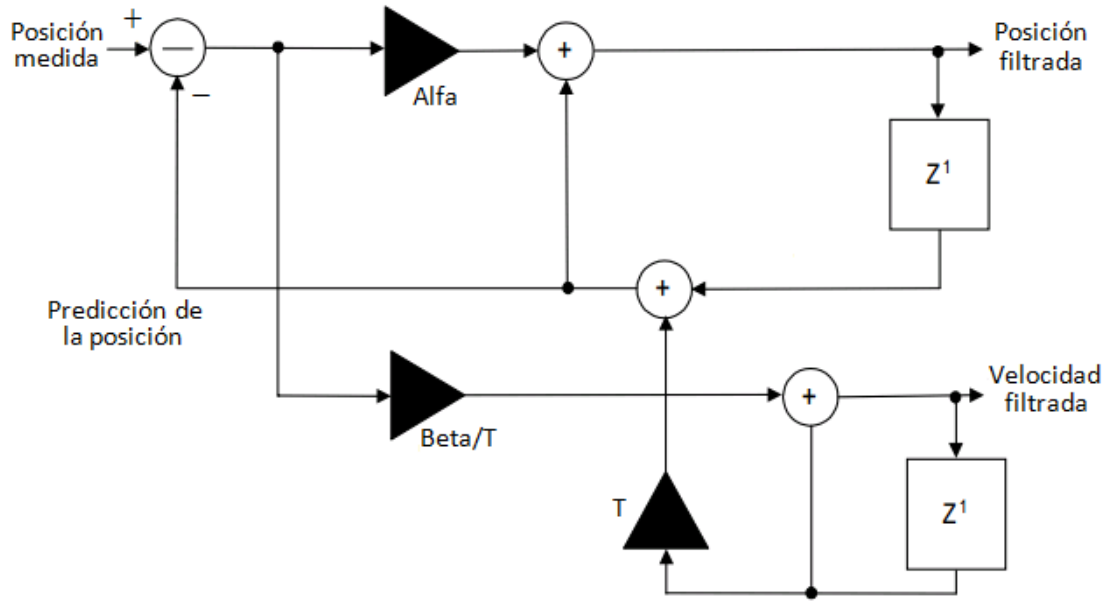


Figura 3.6. Diagrama de bloques del filtro alfa-beta

3.3.2 Características

El primer trabajo relevante sobre el filtro alfa-beta fue realizado por Sklansky (1957), donde estableció algunas claves de diseño importantes: el triángulo de estabilidad, el amortiguamiento crítico para los parámetros alfa y beta, la varianza del ruido y el error de sesgo debido a la aceleración del objetivo [11] y [13].

- **Estabilidad**

Según el trabajo de Sklansky, el triángulo de estabilidad es la región determinada por las siguientes relaciones:

$$\beta > 0 \quad 4 - 2\alpha - \beta > 0 \quad \alpha > 0$$

Dentro de esta región (representada en la figura 3.7), el filtro alfa-beta se ajusta a la posición y velocidad verdadera para un blanco ideal (sin ruido y sin maniobras)

- **Amortiguamiento crítico**

La estabilidad no es el único criterio para elegir el valor de los coeficientes, el filtro debe ajustarse rápidamente. Para un salto de escalón en la velocidad (y en un caso sin ruido) el amortiguamiento crítico de la respuesta del filtro sucede cuando:

$$\beta_{CD} = 2 - \alpha - 2\sqrt{1-\alpha}, \qquad 0 < \alpha < 1$$

Si el blanco se comporta idealmente, esta condición es una buena elección para α y β desde el punto de vista del control. En la región subamortiguada sobre la curva de amortiguamiento crítico, el filtro tendrá una respuesta más rápida pero aparecerán oscilaciones.

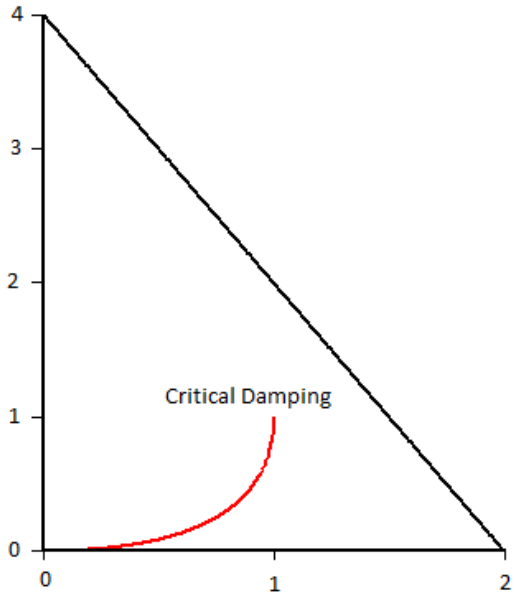


Figura 3.7. Triangulo de estabilidad

- **Reducción del ruido**

Dado que las medidas con ruido X(k) intervienen en el filtro alfa-beta, el ruido estará incluido en las posiciones y velocidades determinadas. Sklansky determinó que la varianza del error predicho se reduce de manera:

$$\sigma_{\dot{X}_p}^2 = \frac{6\alpha^2 - 3\alpha\beta + 6\alpha - \beta^2}{3\alpha(4 - 2\alpha - \beta)}\sigma_n^2$$

- **Sesgo**

Usando el filtro alfa-beta, aparecerá sesgo en la posición y en la velocidad debido a una aceleración A. El error es de la forma:

$$\begin{bmatrix} \mathrm{x_f[k+1]} \\ \mathrm{v_f[k+1]} \end{bmatrix} = \begin{bmatrix} 1-\alpha & (1-\alpha)\mathrm{T_m} \\ \dfrac{-\beta}{\mathrm{T_m}} & 1-\beta \end{bmatrix} \begin{bmatrix} \mathrm{x_f[k]} \\ \mathrm{v_f[k]} \end{bmatrix} + \begin{bmatrix} \dfrac{1}{2}T^2 \\ \mathrm{T} \end{bmatrix} A$$

En estado fijo, el vector de sesgo es:

$$\begin{bmatrix} \mathrm{x_f[k+1]} \\ \mathrm{v_f[k+1]} \end{bmatrix} = \begin{bmatrix} (1-\alpha)T^2 \\ (\alpha - \dfrac{\beta}{2})T \end{bmatrix} \frac{A}{\beta}$$

- **Criterio óptimo**

Benedict y Bordner (1962) formularon un criterio de funcionamiento del filtro basado en el ruido y el error transitorio. Las penalizaciones por el ruido de medida y el transitorio son:

	Posición	**Velocidad**
Reducción de varianza	$\dfrac{2\alpha^2 + \beta(2-3\alpha)}{\alpha(4-2\alpha-\beta)}$	$\dfrac{2\beta^2}{T^2\alpha(4-2\alpha-\beta)}$
Rendimiento transitorio	$\dfrac{2\alpha^2 + \beta(2-3\alpha)}{\alpha(4-2\alpha-\beta)}$	$\dfrac{\alpha^2(2-\alpha) + 2\beta(1-\alpha)}{T^2\alpha\beta(4-2\alpha-\beta)}$

Usando cálculos de variaciones para minimizar el rendimiento del error transitorio sujeto a un nivel de ruido constante, la relación óptima α y β entre es:

$$\beta_{BB} = \frac{\alpha^2}{(2-\alpha)}$$

Kalata (1984) propone otra solución óptima al problema de seguimiento basado en el concepto del Índice de Seguimiento Λ, propuesto en [12], de la siguiente forma:

$$\beta_K = 2(2-\alpha) - 4\sqrt{1-\alpha}$$

$$\Lambda^2 = \frac{T^2\sigma_w^2}{\sigma_n^2}$$

Donde σ^2n es la varianza del ruido de la medida y σ^2w es la varianza del ruido del proceso (magnitud desconocida). La ganancia en posición α, puede obtenerse directamente del Índice de Seguimiento mediante:

$$\alpha = \frac{-L^2 + \sqrt{L^4 + 16L^2}}{8}, \quad \text{con } L^2 = \Lambda^2 + 8\Lambda$$

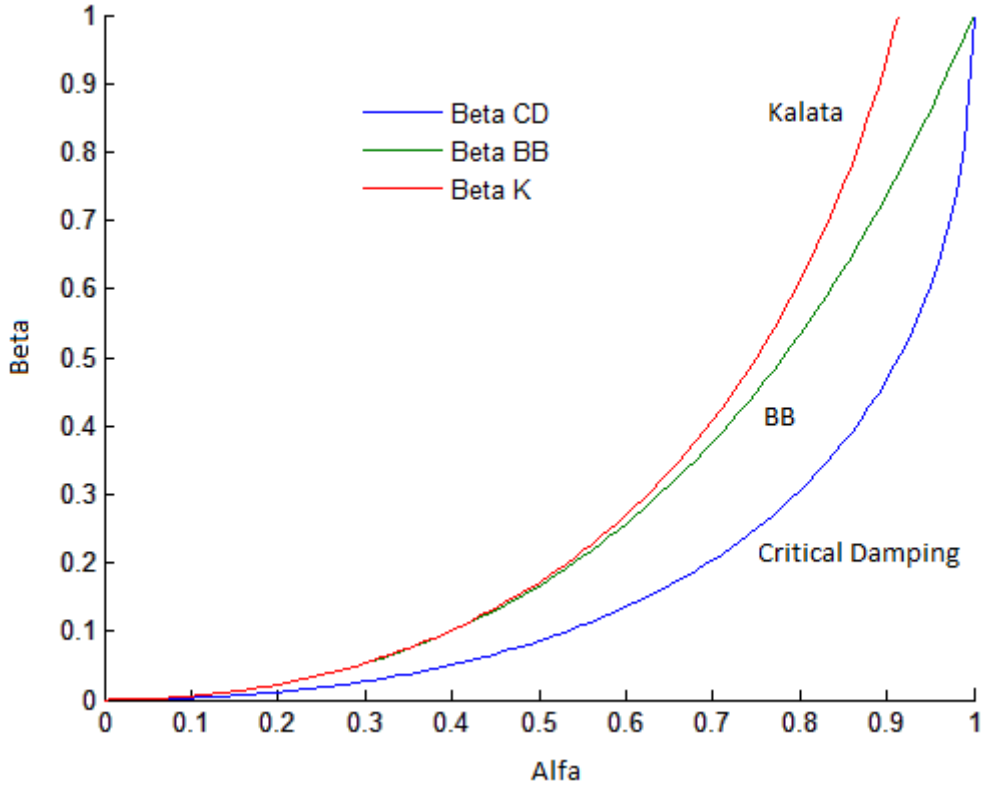


Figura 3.8. Representación de las relaciones alfa-beta propuestas

3.3.3 Aplicaciones del filtro alfa-beta

El filtro alfa-beta se utiliza principalmente en sistemas de radar. Por ejemplo, en [25], se establece un sistema que utiliza un DSP como sistema de seguimiento digital: recoge la información procedente de un radar monopulso, transforma las coordenadas esféricas a cartesianas, aplica el algoritmo del filtro alfa-beta y transmite la información de seguimiento a una plataforma móvil para que siga la posición del blanco.

3.4 MÉTODOS GRID-BASED

Este tipo de métodos proporcionan la recursión óptima de la densidad filtrada $p(\mathbf{x}_k|\mathbf{z}_{1:k)}$ si el espacio de estados es discreto y consiste en un número finito de estados [16]. Suponer que el espacio de estados en el tiempo *k-1* consiste en los estados discretos $\mathbf{x}_{k-1}^i, i = 1, \dots, N_s$. Para cada estado $\mathbf{x}_{k-1}^i$; la probabilidad condicional de ese estado, dadas las medidas hasta el tiempo $k-1$, se denomina como $w_{k-1|k-1}^i$; esto es: $\mathbf{P_r}(\mathbf{x}_{k-1} = x_{k-1}^i \mathbf{z}_{1:k-1}) = w_{k-1|k-1}^i$. Entonces, la densidad espectral de potencia posterior en $k-1$ se puede escribir como

$$p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) = \sum_{i=1}^{N_s} w_{k-1|k-1}^i \, \delta(\mathbf{x}_{k-1} - \mathbf{x}_{k-1}^i)$$

Usando la ecuación anterior, se obtienen las ecuaciones de predicción y actualización:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \sum_{i=1}^{N_s} w_{k|k-1}^i \, \delta(\mathbf{x}_k - \mathbf{x}_k^i)$$

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \sum_{i=1}^{N_s} w_{k|k}^i \, \delta(\mathbf{x}_k - \mathbf{x}_k^i)$$

Donde

$$w_{k|k-1}^i \triangleq \sum_{j=1}^{N_s} w_{k-1|k-1}^j \, p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^j)$$

$$w_{k|k}^i \triangleq \frac{w_{k|k-1}^i \, p(\mathbf{z}_k|\mathbf{x}_k^i)}{\sum_{j=1}^{N_s} w_{k|k-1}^j \, p(\mathbf{z}_k \mathbf{x}_k^j)}$$

Las ecuaciones anteriores asumen que $p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^j)$ y $p(\mathbf{z}_k|\mathbf{x}_k^i)$ son conocidas pero limitan la forma particular de las densidades discretas.

3.5 FILTROS DE PARTÍCULAS

3.5.1 Introducción

Los filtros de partículas son métodos de Monte Carlo secuenciales basados en representaciones puntuales de densidades de probabilidad (aproximan la distribución usando un elevado número de muestras o partículas), que pueden aplicarse a cualquier modelo de espacio de estado y generalizan los métodos tradicionales de filtrado Kalman. Fue propuesto en 1993 por N. Gordon, D. Salmond y A. Smith.

La idea básica es que cualquier *pdf* puede representarse como un conjunto de partículas, tal que la densidad de partículas en un área del espacio de estados representa la probabilidad de esa región. Con este método se puede representar aproximadamente cualquier distribución arbitraria no Gaussiana. A las partículas de la probabilidad anterior se aplican unos pesos "*importance sampling*", mediante el cual las partículas mantienen su situación dentro del estado pero se modifica su peso; se aplica un modelo de movimiento para que las partículas se dispersen según el conocimiento que tenemos del sistema. Finalmente se vuelve a tomar muestras (*resampling*) para eliminar las partículas de menos peso. La figura 3.9 muestra un esquema del proceso.

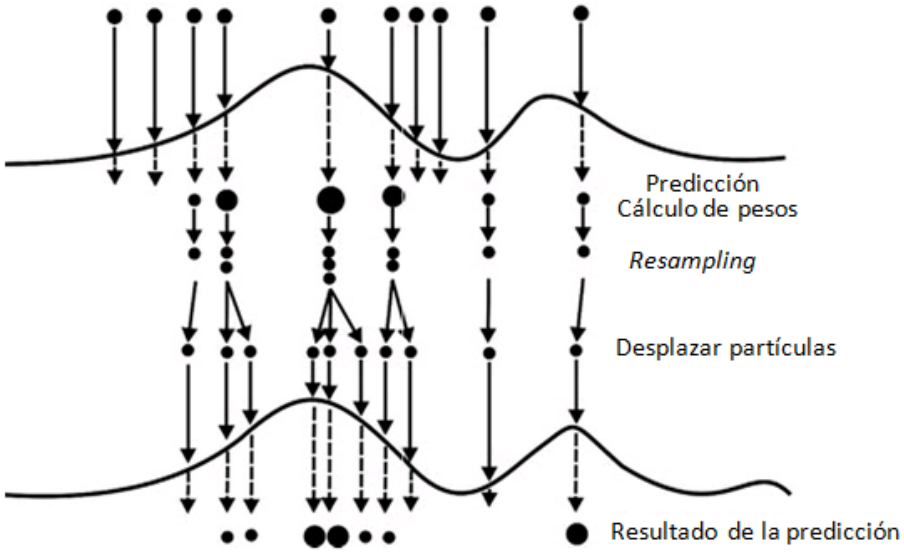


Figura 3.9. Esquema del funcionamiento del filtro de partículas

3.5.2 Características

Según los manuales de referencia [16] y [26], existen varios métodos de implementar filtros de partículas que se introducen brevemente a continuación, además de algunas ideas adicionales:

- **Método: Importance Sampling**

Este es el método más básico, y se puede resumir en los siguientes pasos: usar una densidad "de importancia" (*importance density*) y unos pesos para modelar la densidad.

$$\pi_n(x_{1:n}) = \frac{w_n(x_{1:n})q_n(1_{1:n})}{Z_n}$$

Donde $\pi_n(x_{1:n})$ es el muestreo de una secuencia de densidades de probabilidad, $W_n(x_{1:n})$ son pesos, $q_n(1_{1:n})$ es la "*importance density*" y Z_n es un factor de normalización.

La densidad puede estimarse como:

$$\widehat{\pi_n}(x_{1:n}) = \sum_{i=1}^{N} W_n^i \, \delta_{X_{1:n}^i}(x_{1:n}), \qquad \text{con } W_n^i = \frac{w_n(X_{1:n}^i)}{\sum_{j=1}^{N} w_n(X_{1:n}^j)}$$

Se muestrean N variables $X_{1:n}^i$ aleatorias independientes de cada distribución de probabilidad y se estima la distribución, con $\delta_{X_{1:n}^i}$ la delta en cada muestra.

- **Método: Sequential Importance Sampling**

El algoritmo SIS es un método Monte Carlo (MC) que forma la base de la mayoría de filtros MC secuenciales desarrollados en las últimas décadas. Es una técnica que implementa un filtro recursivo Bayesiano mediante simulaciones MC. La idea principal es representar la función de densidad posterior necesaria mediante un conjunto de muestras aleatorias con pesos asociados y calcular estimaciones basadas en las muestras y los pesos. A medida que el número de muestras se hace muy grande, esta caracterización MC se convierte en un representación equivalente de la *pdf* posterior, y el filtro SIS se aproxima a la estimación Bayesiana óptima.

Existen múltiples variantes del filtro de partículas como SIR (Sampling Importance Resampling), ASIR (Auxiliary Sampling Importance Resampling) o RPF (Regularized Particle Filter) que se incluyen dentro del marco genérico del algoritmo SIS, el cual forma la base de la mayoría de los filtros de partículas que se han desarrollado hasta la fecha.

Los pasos resumidos del algoritmo SIS son los siguientes:

1. se elige una "*importance distribution*" de forma que se cumpla $q_n(x_{1:n}) = q_{n-1}(x_{1:n-1}) q_n(x_n|x_{1:n-1})$
2. En el primer instante de tiempo, se toma una muestra de la probabilidad inicial de la distribución original $X_1^i \sim q_1(x_1)$
3. Se calculan los pesos $w_1(X_1^i)$ y W_1^i
4. Para los siguientes instantes de tiempo, se muestrea de las probabilidades condicionales $X_n^i \sim q_n(x_n|X_{1:n-1}^i)$
5. Se calculan los pesos $w_n(X_{1:n}^i)$ y W_n^i

- **Resampling:**

Los algoritmos IS y SIS proporcionan estimaciones cuyas varianzas aumentan, normalmente de forma exponencial, con *n*. Un problema común es el fenómeno de degeneración, donde después de unas iteraciones todas las partículas salvo una tendrán un peso insignificante. La varianza de los pesos siempre crece con el tiempo y, por tanto, no se puede evitar la degeneración; esto implica que se malgasta mucho esfuerzo computacional en actualizar partículas cuya contribución a la aproximación es casi nula. Para reducir la varianza, se vuelve a muestrear de las distribuciones aproximadas que se acaban de crear.

Los métodos Monte Carlo Secuenciales y los filtros de partículas son una combinación del algoritmo SIS y del *resampling*.

- **Filtros de partículas:**

Para el caso de un algoritmo genérico de un filtro de partículas, la distribución a modelar que debemos usar tiene la forma $\pi_n(x_{1:n}) = p(x_{1:n}|y_{1:n})$, la "*importance distribution*" necesaria es $q_n(x_{1:n}|x_{1:n-1}) = q(x_n|y_n, x_{n-1})$ y los pesos $\alpha_n(x_{1:n}) = p(y_n|x_{n-1})$. Hay que tener en cuenta que *q* sólo depende del estado anterior y de la observación actual.

El algoritmo queda de la siguiente manera:

- En el instante n=1
  - Se muestrea la distribución $X_1^i \sim q(x_1|y_1)$
  - Se calculan los pesos $w_1(X_1^i) = \frac{\mu(X_1^i)g(y_1|X_1^i)}{q(X_1^i|y_1)}$ y $W_1^i \propto w_1(X_1^i)$
  - Se vuelve a muestrear $\{W_1^i, X_1^i\}$para obtener *N* partículas de igual peso $\{\frac{1}{N}, \bar{X}_1^i\}$

- Para los instantes n>2
  1. Se muestrea la distribución $X_n^i \sim q(x_n|y_n, \bar{X}_{n-1}^i)$ y se establece $X_{1:n}^i \leftarrow (\bar{X}_{i:n-1}^i, X_n^i)$
  2. Se calculan los pesos $\alpha_1(X_{n-1:n}^i) = \frac{g(y_n|X_n^i)f(X_n^i|X_{n-1}^i)}{q(X_n^i|y_n, X_{n-1}^i)}$ y $W_n^i \propto \alpha_1(X_{n-1:n}^i)$
  3. Se vuelve a muestrear (*resampling*) $\{W_n^i, X_{1:n}^i\}$para obtener *N* partículas nuevas de igual peso $\left\{\frac{1}{N}, \bar{X}_{1:n}^i\right\}$

- Para cada instante de tiempo *n* tenemos las distribuciones estimadas:

$$\hat{p}(x_{1:n}|y_{1:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_{1:n}^i}(x_{1:n})$$

$$\hat{p}(y_{1:n}|y_{1:n-1}) = \sum_{i=1}^{N} W_{n-1}^i \alpha_n(X_{n-1:n}^i)$$

En el algoritmo anterior surgen algunos problemas: aunque se use la distribución de importancia óptima $p(x_n|y_n, x_{n-1})$, el modelo puede no ser eficiente. De hecho, si la varianza de $p(y_n|x_{n-1})$ es alta, entonces la varianza de la aproximación resultante también lo será y, en consecuencia, será necesario realizar la operación de *resampling* muy frecuentemente y la aproximación de partículas $\hat{p}(x_{1:n}|y_{1:n})$ será poco fiable.

Otro problema asociado a la degeneración es que las variables $\{X_n^i\}$ se muestrean en el instante *n* pero los valores $\{X_{1:n-1}^i\}$ se mantienen fijos lo que produce inexactitudes al muestrear.

Este algoritmo básico puede ser mejorado de varias maneras:

- En algunas ocasiones es posible intercambiar el orden de los pasos de *sampling* y *resampling*, ya que esto produce una mejor aproximación de la distribución al proporcionar un mayor número de partículas distintas para aproximar al blanco. En general, la operación de *resampling* debe ser realizada dentro de una iteración antes que otra operación que no influya en los pesos para minimizar la pérdida de información. Sin embargo, los pesos en general dependen del nuevo estado y no es posible aplicar directamente un cambio de orden. En este sentido, intercambiar las dos operaciones produce un algoritmo en cuya información de la siguiente observación se usa para determinar qué partículas deben sobrevivir al *resampling* en un tiempo dado; por lo que es necesario buscar métodos que hagan uso de esta información futura para obtener algún tipo de ventaja en las ocasiones en que no se pueda usar la distribución óptima.
  El algoritmo Auxiliar Particle Filter (APF) hace básicamente esto, *resampling* antes de calcular los pesos.
- Algoritmo Resample-Move: permite solucionar el problema de degeneración mencionado usando Markov Chain Monte Carlo (MCMC) para hacer fluctuar la localización de las partículas y reducir el efecto de la degeneración. Este método introduce diversidad en el conjunto de partículas después del *resampling,* no disminuye el número de intervalos de *resampling* en comparación con el algoritmo básico.
- Algoritmo Block Sampling: mejora el algoritmo anterior muestreando las componentes anteriores en bloques.

3.5.3 Usos actuales de los filtros de partículas

Desde su creación, los filtros de partículas, han encontrado aplicación en muy diversas disciplinas:

- Localizar fuentes de señales acústicas
- Vigilancia en video
- Robótica
- Vehículos automáticos
- Seguimiento: en entornos inteligentes, seguimiento visual de personas
- Aplicaciones financieras

Existen en la literatura actual numerosos artículos que utilizan filtros de partículas para detección y seguimiento de personas. Por ejemplo, en [27] los autores desarrollan un método de detección de múltiples blancos basado en el enfoque *tracking-by-detection*(este método consiste en la continua aplicación de un algoritmo de detección sobre *frames* individuales y la asociación de detecciones a través de *frames*) y utilizando un filtro de partículas, el cual establece un marco de trabajo para representar la incertidumbre del seguimiento de la manera de un modelo de Markov, considerando únicamente información de *frames* pasados, lo que permite su uso en aplicaciones en tiempo real. Para el seguimiento de varios blancos utilizan conjuntos de partículas independientes, lo que a su vez requiere de un método de asociación de datos como el Algoritmo Húngaro o el Algoritmo Voraz.

En el artículo [28] los autores describen varias aplicaciones del filtro de partículas en el campo de la automoción y navegación aérea. En concreto aplican el filtro de partículas para la determinación de la posición de automóviles, solapando la información de velocidad y dirección que proporcionan los coches sobre un mapa digital; o en otra alternativa, utilizando redes inalámbricas. Igualmente, se aplica al posicionamiento de aviones utilizando información de elevación sobre mapas digitales. Por último, también introducen los filtros de partículas en el seguimiento de blancos, para el control del tráfico aéreo y para evitar colisiones entre coches.

CAPÍTULO 4

ESTADO DEL ARTE: MEDIDAS DE CALIDAD DE IMAGEN

En este proyecto, el método que se va a utilizar para detectar movimiento en secuencias de vídeo es el algoritmo SSIM. Sin embargo, SSIM originalmente fue desarrollado por Z. Wang [10] como un método para determinar la calidad de una imagen. En el presente capítulo realizaremos un repaso sobre los mecanismos de evaluación de la calidad de imagen, entre los que se encuentra la similitud estructural (SSIM).

En los últimos años, ha habido un interés creciente en el desarrollo de métodos de evaluación objetiva de la calidad de imagen, *image quality assessment* (IQA), que permiten predecir de forma automática los comportamientos humanos en la evaluación de la calidad de imagen. Estas medidas de IQA perceptual tienen amplias aplicaciones en la evaluación, control, diseño y optimización de sistemas de adquisición de imágenes, sistemas de comunicación, de procesado y de visualización.

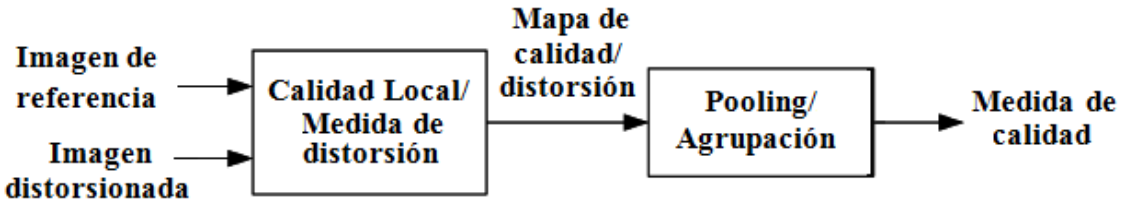


Figura 4.1. Esquema básico de un sistema de evaluación de la calidad de imágenes [29]

Además de SSIM, existen otros mecanismos para determinar la calidad de la imagen, entre ellos uno de los más conocidos es el error cuadrático medio (MSE), pero existen otros muchos que se fundamentan en distintos principios. Muchas de las implementaciones para realizar la IQA adoptan una estructura común de dos etapas: la primera consiste en la medida local de la calidad/distorsión de la imagen, y la segunda es la agrupación o puesta en común de los datos. Tales aproximaciones se estiman que son consistentes con el funcionamiento del sistema visual humano (HVS).

4.1 INTRODUCCIÓN

Las imágenes digitales están sujetas a gran variedad de distorsiones durante las etapas de procesado, compresión, almacenamiento, transmisión o reproducción; y cualquiera de ellas puede resultar en la degradación de la calidad visual. Para aplicaciones en las que el objetivo último sea que el ser humano visualice las imágenes, el único método adecuado para cuantificar la calidad de la imagen es a través de una evaluación subjetiva.

En la práctica, sin embargo, las evaluaciones subjetivas normalmente son inconvenientes y costosas. El objetivo en la investigación en la evaluación objetiva de la calidad de imagen es desarrollar medidas cuantitativas que puedan predecir automáticamente la calidad de la imagen percibida.

Un método de medición objetiva de la calidad de imagen puede jugar gran variedad de roles en aplicaciones de procesado de imágenes. En primer lugar, se puede usar para monitorizar dinámicamente y ajustar la calidad de imagen. En segundo lugar, puede emplearse para optimizar algoritmos y la configuración de parámetros en sistemas de procesado de imágenes. Tercero, se puede utilizar para referenciar algoritmos y sistemas de procesado de imágenes.

Las métricas objetivas pueden clasificarse dependiendo de si se dispone de una imagen original (sin distorsión), con la cual la imagen que está siendo medida se compara. La mayoría de los enfoques existentes se conocen como *full-reference*, lo que significa que una imagen de referencia completa es conocida con la cual podemos comparar imágenes distorsionadas. En muchas aplicaciones prácticas, sin embargo, la imagen de referencia no está disponible y es necesario un enfoque “ciego” o sin referencia (*no-reference*). En un tercer método, la imagen de referencia sólo está disponible parcialmente, en forma de un conjunto de características extraídas para ayudar a evaluar la calidad de la imagen. Se conoce como evaluación de calidad de referencia reducida (*reduced-reference*). Como hemos dicho, los métodos *full-reference* son los más abundantes y los que vamos a comentar a continuación [10].

El método *full-reference* más simple y más utilizado es el error cuadrático medio (MSE), que se calcula promediando el cuadrado de la diferencia de intensidad entre pixeles distorsionados y de referencia, junto con la cantidad correspondiente de relación señal-ruido, *peak signal-to-noise ratio* (PSNR).

Estos métodos son atractivos porque son fáciles de calcular, tienen un significado físico claro y son convenientes en el contexto de optimización. Pero no están muy bien adaptados a la calidad visual percibida. En las tres últimas décadas, una gran cantidad de esfuerzo ha ido dirigido al desarrollo de métodos de evaluación de calidad que tomen ventaja de las características conocidas del sistema de visión humano (HVS).

4.2 EVALUACIÓN DE LA CALIDAD DE IMAGEN BASADA EN LA SENSIBILIDAD DEL ERROR

En el contexto de la evaluación de la calidad de una imagen, dicha imagen puede verse como la suma de una señal de referencia no distorsionada y una señal de error. Un supuesto ampliamente aceptado es que la pérdida de la calidad de percepción está directamente relacionada con la visibilidad del error.

La implementación más simple de este método es el MSE (*Mean Squared Error*), que cuantifica de manera objetiva la potencia de la señal de error. Pero dos imágenes con el mismo MSE pueden tener diferentes tipos de error, algunos de los cuales son mucho más visibles que otros. La mayoría de enfoques propuestos en la literatura intentan ponderar los diferentes aspectos de la señal de error de acuerdo con su visibilidad, según lo determinado por medidas psicofísicas en los seres humanos o las mediciones fisiológicas en los animales.

4.2.1 ESTRUCTURA

La siguiente figura ilustra un esquema genérico de evaluación de calidad de imagen basado en la sensibilidad de error.

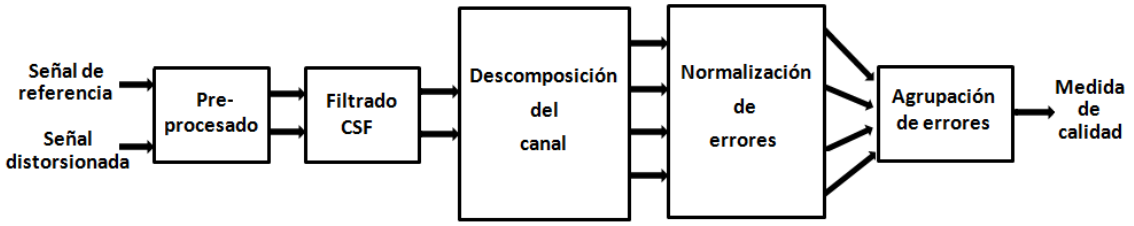


Figura 4.2. Esquema de un sistema de evaluación de calidad basado en la sensibilidad de error

Aunque varían en los detalles, la mayoría de modelos pueden describirse con un diagrama similar [10]. Las etapas principales del procedimiento son las siguientes:

- Pre-procesado: En esta etapa se suele realizar una gran variedad de operaciones para eliminar distorsiones de las imágenes que se comparan. En primer lugar, las señales distorsionadas y de referencia son correctamente alineadas y escaladas. En segundo lugar, la señal debe ser convertida a un espacio de color que sea más apropiado para el HVS. En tercer lugar, las métricas de evaluación de calidad deben convertir los valores de los píxeles almacenados en la memoria en valores de luminancia en el dispositivo de visualización a través de transformaciones punto a punto no lineales. Finalmente, las señales distorsionadas y de referencia se modifican mediante operaciones no lineales para simular adaptación a la luz.
- Filtrado CSF: La función de la sensibilidad al contraste (CSF) describe la sensibilidad del HVS a diferentes frecuencias temporales y espaciales que están presentes en los estímulos visuales. Algunas métricas de calidad de imagen incluyen una etapa que

pondera la señal de acuerdo con esta función (normalmente se implementa usando un filtro lineal que se aproxima a la respuesta en frecuencia del CSF). Por otra parte, otras métricas más recientes eligen implementar CSF como un factor de normalización, después de la descomposición del canal.

- Descomposición del canal: Las imágenes son típicamente separadas en sub-bandas (comúnmente llamados canales), que son selectivos en frecuencia y orientación. Existen varias opciones cuando se trata de implementar la descomposición: Mientras algunos métodos de evaluación de calidad realizan complejas descomposiciones de canal que intentan aproximarse a las respuestas neuronales en la corteza visual primaria, muchas métricas usan transformadas más sencillas como la transformada discreta del coseno (DCT) o transformadas wavelet separables.
- Normalización de errores: El error (la diferencia) entre la señal de referencia descompuesta y las señales distorsionadas en cada canal se calcula y normaliza de acuerdo con cierto modelo de enmascaramiento, que tiene en cuenta el hecho de que la presencia de una componente de una imagen disminuirá la visibilidad de otra componente próxima, temporal o espacialmente. El mecanismo de normalización pondera la señal de error en un canal mediante un umbral de visibilidad variable en el espacio. El umbral de visibilidad en cada punto es calculado en base a la energía de la referencia y/o a los coeficientes de distorsión en un entorno.
- Agrupación del error (*Error pooling*): La fase final de todas las métricas de calidad debe combinar las señales de error normalizadas a través de los diferentes canales en un único valor. Para la mayoría de los métodos de evaluación de la calidad, la agrupación toma la forma de una normalización de Minkowski

$$E(\{e_{l,k}\} = (\sum_{l}\sum_{k}|e_{l,k}|^{\beta})^{1/\beta}$$

Donde $e_{l,k}$ es el error normalizado del n-ésimo coeficiente en el canal n-ésimo, y β es una constante elegida típicamente entre 1 y 4. La agrupación de Minkowski puede realizarse sobre el espacio (índice k) y luego sobre frecuencia (índice l) o viceversa. Para proporcionar ponderación espacial, puede usarse un mapa espacial que indique la importancia relativa de las diferentes regiones.

4.2.2 LIMITACIONES

El principio subyacente en el que se basa el enfoque de sensibilidad del error es que la calidad percibida se estima mejor cuantificado la visibilidad de los errores. Esto se consigue, esencialmente, simulando las propiedades funcionales de las etapas tempranas del HSV, caracterizadas mediante experimentos psicológicos y psicofísicos. Aunque este enfoque ha encontrado aceptación casi universal, es importante reconocer sus limitaciones. En particular, el HSV es un sistema complejo y altamente no lineal, pero la mayoría de modelos están basados en operadores lineales o cuasi-lineales que han sido caracterizados usando estímulos

simplistas y restringidos. Así, los enfoques basados en la sensibilidad del error se basan en una serie de grandes supuestos y generalizaciones [10]:

- El problema de la definición de calidad: El mayor problema es que no está claro que el error de visibilidad deba ser equiparado con pérdidas de calidad, ya que algunas distorsiones pueden ser percibidas claramente pero no son demasiado molestas.
- El problema del umbral: Los experimentos que subyacen bajo muchos de los modelos de sensibilidad del error están diseñados para estimar el umbral bajo el cual el estímulo es apenas visible. Sin embargo, muy pocos estudios indican si estos modelos cercanos al umbral pueden ser extendidos para caracterizar distorsiones mucho mayores que el nivel de umbral.
- El problema de la complejidad de la imagen natural: La mayor parte de los experimentos se realizan empleando modelos o patrones sencillos, como puntos, barras o enrejados sinusoidales. Sin embargo, todos estos modelos son mucho más simples que las imágenes del mundo real, que podrían pensarse como la superposición de un número mucho mayor de patrones simples. Por tanto, no es posible asegurar que los modelos sean capaces de evaluar la calidad de imágenes naturales de estructura compleja.
- El problema de la no-correlación: El uso de la métrica de Minkowski para la agrupación espacial de errores supone asumir que los errores son estadísticamente independientes. Esto sería cierto si el procesado previo a la agrupación eliminara las dependencias en las señales de entrada, sin embargo, esto no ocurre para métodos de descomposición lineal del canal, como por ejemplo la transformada wavelet.
- El problema de la interacción cognitiva: La comprensión cognitiva y acciones del procesamiento visual influyen en la calidad percibida de la imagen. Es por esto que un observador humano calificará de diferente manera una imagen dependiendo de la información de que disponga, las instrucciones que haya recibido o la atención que esté prestando. La mayor parte de las medidas de la calidad de imagen no tienen en consideración estos efectos ya que son difíciles de parametrizar.

4.2.3 MSE

A continuación procedemos a comentar en qué consiste el método del error cuadrático medio, y también veremos qué propiedades son ventajosas y cuales son inconvenientes para considerar el MSE como un buen mecanismo de evaluación de calidad [30].

Se supone que X e Y son dos señales discretas de longitud finita (por ejemplo, dos imágenes), donde N es el número de muestras (píxeles, en el caso de imágenes). El error cuadrático medio (MSE) entre las imágenes es:

$$MSE(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2$$

En el MSE, nos referimos a la señal error como $e_i = x_i - y_i$, que es la diferencia entre la señal original y la señal distorsionada. Si partimos de que una de las señales es original o tiene calidad aceptable, entonces es MSE puede ser visto como la medida de la calidad de la señal. Una forma más general es la norma l_p:

$$d_p(x,y) = (\sum_{i=1}^{N}|e_i|^p)^{1/p}$$

En la literatura relacionada con el procesado de imágenes, normalmente el MSE se convierte en la relación de pico de señal a ruido (peak signal-to-noise ratio - PSNR):

$$PSNR = 10\log_{10}\frac{L^2}{MSE}$$

Donde L es el rango dinámico de la imagen. Por ejemplo, imágenes que utilizan 8 bits por cada píxel, $L = 2^8 = 255$. La PSNR es útil si se comparan imágenes con rango dinámico, pero de otra forma no proporciona información distinta a la de MSE.

- **Ventajas de MSE**

1) Es simple. Está libre de parámetros y no es costoso de calcular, ya que para cada muestra sólo se realiza una multiplicación y dos sumas. No tiene memoria, el cálculo del error de una muestra es independiente del resto.
2) Todas las normas l_p son medidas de la distancia válidas en $\boldsymbol{R}^N$, lo que satisface las siguientes condiciones:
   - No negativo: $d_p(x,y) \geq 0$
   - Identidad: $d_p(x,y) = 0$ si y sólo si X=Y
   - Simetría: $d_p(x,y) = d_p(y,x)$
   - Desigualdad triangular: $d_p(x,z) \leq d_p(x,y) + d_p(y,z)$

   El caso $p = 2$, (proporcional a la raíz cuadrada del MSE) es la medida de la distancia en un espacio euclídeo N-dimensional.
3) Tiene un claro significado físico. Tal como está expresado, MSE es la forma natural de definir la energía de la señal de error. Por el teorema de Parseval, la medida de la energía se conserva después de cualquier transformación ortogonal y lineal. Esta propiedad garantiza que la energía de una distorsión de la señal en el dominio transformado es igual que en dominio de la señal.
4) El MSE es una medida excelente en el contexto de la optimización, ya que posee las propiedades de convexidad, simetría y diferenciabilidad. Los problemas de optimización del Mínimo-MSE normalmente tienen solución analítica cerrada.

5) El MSE es apropiado para estimación y estadísticas. El MSE es aditivo para fuentes de distorsión independientes.
6) El MSE es ampliamente utilizado simplemente por el hábito y la costumbre. Históricamente, se ha empleado para optimizar una gran variedad de aplicaciones de procesado de señal como diseño de filtros, compresión de señal, restauración, reconstrucción y clasificación.

- **Inconvenientes de MSE**

MSE posee muchas propiedades que hacen su uso favorable, sin embargo no ofrece buenas prestaciones cuando se emplea para medir la calidad de una imagen ya que no se adapta bien a la percepción humana.

Esto se debe en parte a unos supuestos implícitos que se deben tener en cuenta cuando se trabaja con MSE. Estos supuestos son muy restrictivos, ya que imponen limitaciones a las muestras de señal, cómo interactúan entre ellas y con el error [20].

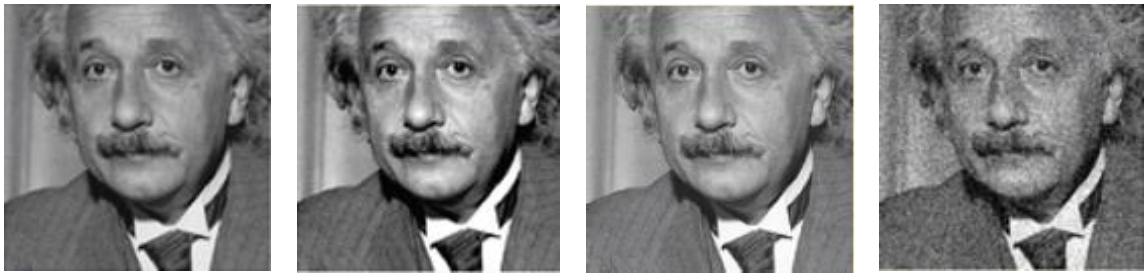
1) La calidad de la señal es independiente de las relaciones temporales o espaciales entre las muestras de la señal original. En otras palabras, si las señales original y distorsionada se reordenan aleatoriamente, el MSE entre ambas se conserva.
2) La fidelidad de la señal es independiente de cualquier relación entre la señal original y la señal de error. Dada una señal de error, el MSE es el mismo, independientemente de la señal a la que añada.
3) La calidad de la señal es independiente del signo de las muestras de la señal de error.
4) Todas las muestras de la señal tienen la misma importancia en la medida de la calidad.

En las siguientes imágenes de la figura 3.3, obtenidas de [30], vamos a ver que MSE no es un método muy adecuado para establecer un criterio objetivo de calidad en imágenes si lo comparamos con otros mecanismos, como por ejemplo SSIM, el cual se explicará en el próximo apartado.

Ante diferentes tipos de alteraciones de la señal, el valor de calidad proporcionado por MSE es muy parecido. Sin embargo, es fácilmente apreciable a simple vista que la calidad de las imágenes varía ostensiblemente. Por otra parte, el índice SSIM aporta unos valores más cercanos a la realidad, ya que tiene un funcionamiento más cercano al comportamiento de la percepción humana.

La figura (a) se corresponde con la imagen de referencia, y se le asignan los valores de índice máximos, MSE=0 y SSIM=1. La imagen (b) ha sufrido una variación de contraste. Para el índice SSIM la calidad de esta señal es bastante elevada (SSIM=0.928) y para MSE =306. En el caso de las siguientes imágenes (c) y (d) se ha producido un cambio de iluminación y se ha añadido ruido blanco Gaussiano, respectivamente. Mientras que MSE otorga a ambas imágenes el mismo valor (MSE=309), y similar al caso anterior, la calidad de las señales es claramente

diferente y consecuentemente SSIM las evalúa de distinta forma: SSIM= 0.987 y SSIM=0.576, respectivamente.



(a) MSE=0 SSIM=1 | (b) MSE=306 SSIM=0.928 | (c) MSE=309 SSIM=0.987 | (d) MSE=309 SSIM=0.576

Figura 4.3. Comparación de índices MSE y SSIM frente a distintos tipos de distorsión

4.3 EVALUACIÓN DE LA CALIDAD DE IMAGEN BASADA EN LA SIMILITUD ESTRUCTURAL

Las señales de imágenes naturales están altamente estructuradas: sus píxeles presentan fuertes dependencias, sobre todo cuando están espacialmente próximos. Estas dependencias tienen información importante sobre la estructura de los objetos de la escena. La métrica de error de Minkowski, que mencionábamos en el apartado anterior, está basada en diferencias punto a punto de la señal, que son independientes de la estructura subyacente de la señal.

Aunque la mayoría de las medidas de calidad basadas en la sensibilidad del error descomponen la señal mediante transformaciones lineales, esto no elimina las dependencias más fuertes. La motivación del nuevo enfoque es encontrar un modo más directo de comparar las estructuras de las señales de referencia y distorsionadas.

Para presentar y comprender las características de este nuevo enfoque se pueden exponer algunos de los puntos que lo diferencian de la filosofía de sensibilidad del error [10]:

- El método de sensibilidad del error estima el error percibido para cuantificar las degradaciones de las imágenes, mientras que la nueva filosofía considera las degradaciones en imágenes como los cambios percibidos en la variación de la información estructural.
- El paradigma de sensibilidad del error es una aproximación que simula la función de las etapas tempranas en el HSV. El nuevo paradigma es una aproximación arriba-abajo, imitando el supuesto funcionamiento de todo el HSV. Esto, por otra parte, supera el problema del supra-umbral mencionado anteriormente porque no se basa en umbrales psicofísicos para cuantificar la distorsión percibida.
- Los problemas de complejidad de la imagen natural y de no-correlación también se evitan en cierta medida debido a que la nueva filosofía no trata de predecir la calidad de imagen mediante la acumulación de errores. En cambio, el nuevo método propone evaluar los cambios estructurales directamente entre dos señales estructuradas de forma compleja.

4.3.1 SSIM

El índice SSIM (*Structural SIMilarity*), que como hemos dicho sirve para realizar la medida de la calidad de una imagen, puede tomar variedad de formas dependiendo de si se implementa en una escala o sobre varias escalas. La aproximación mediante SSIM fue originalmente motivada por la observación de que las señales de imágenes naturales están altamente estructuradas, sus píxeles presentan fuertes dependencias, sobre todo cuando están espacialmente próximos, y estas dependencias contienen información importante de la estructura de los objetos en la escena [10].

La filosofía principal subyacente en el índice SSIM original es que el sistema visual humano está adaptado a extraer la información estructural de las escenas y por tanto, al menos para la medida de la fidelidad de una imagen, la conservación de la estructura de la señal es un punto importante. De forma equivalente, con el objetivo de conseguir medidas de calidad, un algoritmo puede buscar medir la distorsión estructural que sufre una imagen. Las distorsiones que afectan a una imagen pueden clasificarse como estructurales y no estructurales. A continuación vamos a determinar las diferencias existentes entre ambas [30].

- Las distorsiones no estructurales (un cambio de luminancia o luminosidad, un cambio de contraste, distorsión Gamma, y un desplazamiento espacial) son causadas por las condiciones que suceden durante la toma de las imágenes y su visualización. Estas distorsiones no cambian las estructuras de las imágenes de los objetos en la escena.
- Sin embargo, otras distorsiones (ruido aditivo y compresión con pérdidas y emborronamiento) cambian significativamente la estructura de los objetos. Si consideramos que el sistema visual humano busca identificar y reconocer objetos dentro de una escena, entonces debe ser altamente sensible a las distorsiones estructurales y compensar automáticamente las no estructurales. Por tanto, una medida de la calidad de la señal objetiva debe simular este comportamiento.

- **Aplicaciones**

SSIM se ha utilizado para evaluar los resultados del procesado de imágenes en un creciente número de aplicaciones [30]. Como por ejemplo:

- Fusión de imágenes.
- Compresión de imágenes.
- Calidad de imagen cromática.
- Transmisión (streaming) de video inalámbrico.
- Vigilancia.
- Imágenes de radar.
- Imágenes de infrarrojo.
- Imagen de resonancia magnética (MRI).
- Imágenes de cromosomas.

- Teledetección.
- Reconocimiento de objetos.

■ Funcionamiento

La idea básica de SSIM consiste en partir de dos imágenes X e Y que se van a comparar. Si consideramos que una de las señales tiene calidad perfecta, entonces la medida de la similitud puede servir como una medida cuantitativa de la calidad de la segunda señal. El índice SSIM local mide las similitudes entre tres elementos de las dos imágenes: la luminancia o valores de brillo $\mathrm{l}(\mathrm{x},\mathrm{y})$, el contraste $c(x,y)$ y la estructura $s(x,y)$. Estas semejanzas locales se expresan usando medidas estadísticas y se combinan para formar el índice SSIM local [10].

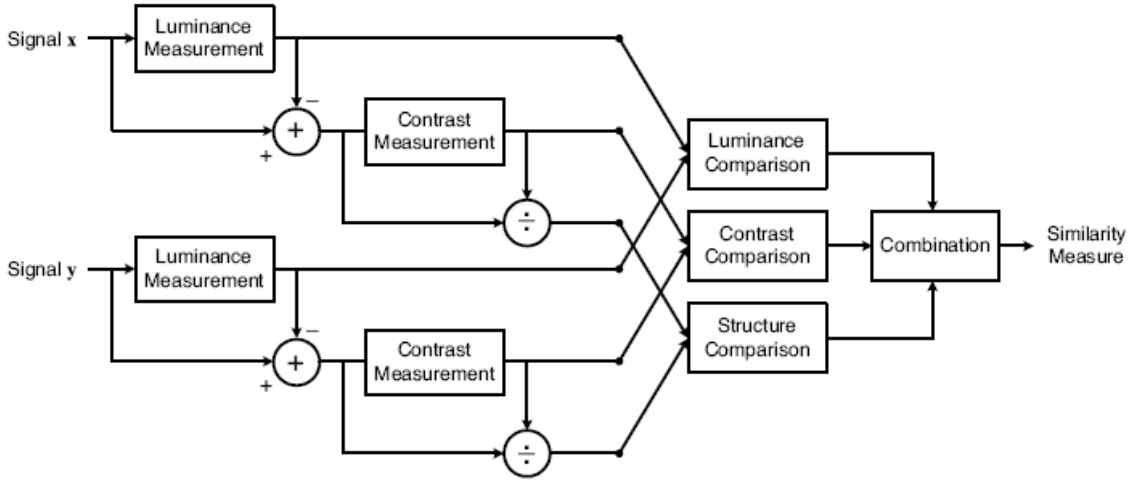


Figura 4.4. Esquema de funcionamiento de SSIM [10]

En primer lugar, se compara la luminancia de cada señal. Asumiendo señales discretas, ésta se estima como la intensidad media:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

La función de comparación de la luminancia $\mathrm{l}(\mathrm{x},\mathrm{y})$, es función de μ_x y μ_y, y se define:

$$\mathrm{l}(\mathrm{x},\mathrm{y}) = \frac{2\mu_x\mu_y + C1}{\mu_x{}^2 + \mu_y{}^2 + C1}$$

Las constantes C1 se han incluido para evitar inestabilidad cuando $\mu_x{}^2$ y $\mu_y{}^2$ es muy próximo a cero. Con el mismo fin se han incluido las constantes C2 y C3 que se aplican en la comparación de contraste y estructura, descritas más adelante.

En segundo lugar, se elimina la intensidad media de la señal. En forma discreta, la señal resultante $x - \mu_x$ se corresponde con la proyección del vector x sobre el hiperplano definido por:

$$\sum_{i=1}^{N} x_i = 0$$

Se utiliza la desviación estándar (la raíz cuadrada de la varianza) como una estimación del contraste de la señal:

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}$$

Por tanto, la comparación del contraste $c(x, y)$, es la comparación entre σ_x y σ_y.

$$c(x, y) = \frac{2\sigma_x\sigma_y + C2}{{\sigma_x}^2 + {\sigma_y}^2 + C2}$$

La comparación de la estructura $s(x, y)$ se lleva a cabo después de la resta de la luminancia y de la normalización de la varianza. La señal es normalizada (dividida) por su propia desviación estándar, de modo que las dos señales comparadas tienen desviación estándar unitaria. Específicamente, se asocian dos vectores unitarios $(x - \mu_x)/\sigma_x$ y $(y-\mu_y)/\sigma_y$. La correlación (producto interno) entre ellos es una forma simple y efectiva para cuantificar la semejanza estructural. Entonces, la función de comparación de estructura se define:

$$s(x, y) = \frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3}$$

Como en la medida de la luminancia y el contraste, se introduce una constante tanto en el denominador como en el numerador. En la forma discreta σ_{xy} puede estimarse como:

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

Finalmente, las tres componentes se combinan para producir una medida de similitud general y se obtiene el índice SSIM entre las dos señales x e y:

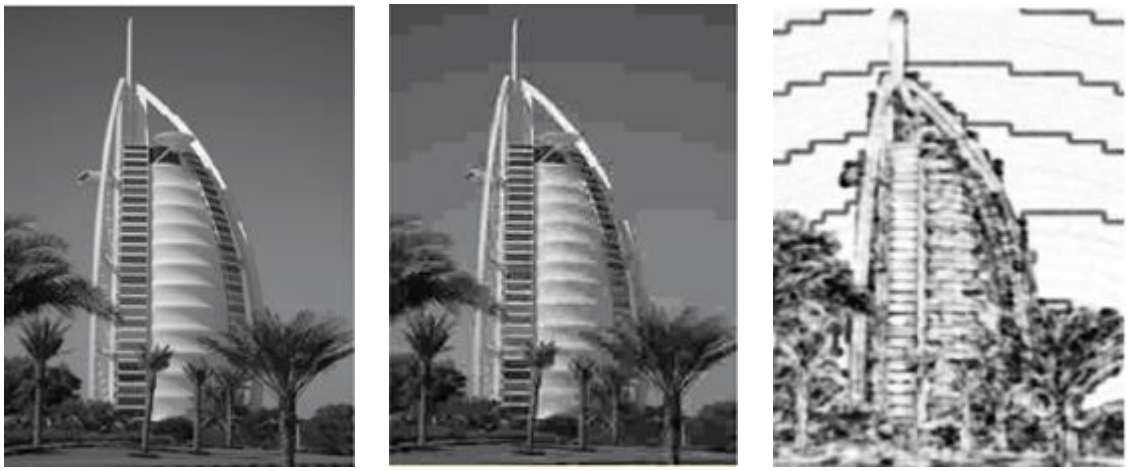
$$SSIM(x, y) = l(x, y) \cdot c(x, y) \cdot s(x, y) =$$
$$= \left(\frac{2\mu_x\mu_y + C1}{{\mu_x}^2 + {\mu_y}^2 + C1}\right) \cdot \left(\frac{2\sigma_x\sigma_y + C2}{{\sigma_x}^2 + {\sigma_y}^2 + C2}\right) \cdot \left(\frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3}\right)$$

El índice SSIM se calcula localmente dentro de una ventana cuadrada deslizante que se mueve píxel a píxel a través de la imagen, generando un mapa SSIM. El índice SSIM de toda la imagen se calcula entonces por la agrupación del mapa SSIM, por ejemplo, simplemente promediando los valores SSIM a través de la imagen.

Un punto importante es que las tres componentes son relativamente independientes. Por ejemplo, cambios de luminancia y/o contraste no afectarán a la estructura de la imagen. También deseamos que la medida de similitud cumpla las siguientes condiciones:

1) Simetría: S(x,y) = S(y,x)
2) Acotación : S(x,y)<=1
3) Máximo único: S(x,y)=1, si y solo si X=Y

En la siguiente figura, tomada de [30], se muestra el funcionamiento del método SSIM. La primera imagen (a) se corresponde con la imagen original y que se considera que tiene calidad perfecta. La segunda imagen (b) es la imagen original a la que se ha aplicado compresión JPEG. Para comprobar la calidad de la señal comprimida respecto a la de referencia se genera el mapa SSIM (c), que puede verse en la última imagen. SSIM captura correctamente los efectos de falsos contornos en la zona del cielo y los bloques que producen en los bordes de los objetos.



(a) (b) (c)

Figura 4.5. Evaluación de calidad mediante SSIM

4.4 EVALUACIÓN DE LA CALIDAD DE LA IMAGEN BASADA EN VARIANZA LOCAL

Además de SSIM existen otros métodos que utilizan en su análisis la información estructural de la imagen, como el método *Quality Index based on Local Variance* (QILV) [31].

Este método se basa en la comparación de la varianza local de dos imágenes, con el objetivo de manejar de forma más apropiada la no estacionaridad de las imágenes que se están comparando. Los procesos no estacionarios surgen de forma natural en imágenes en las cuales

estructuras están presentes, y cambios en el comportamiento estructural implican un cambio en la no estacionaridad. Este método puede verse como un nuevo procedimiento independiente o como un complemento de otros, como por ejemplo SSIM.

El principio en el que se basa este nuevo mecanismo es que una gran cantidad de la información estructural de una imagen se encuentra codificada en su distribución de varianza local. En el índice SSIM, se calcula la varianza local de las dos imágenes, pero el índice global conjunto únicamente considera la media de ambos valores y, por lo tanto, la no estacionaridad de las imágenes no se tiene en cuenta.

La varianza local de una imagen I se define como:

$$Var(I_{i,j}) = E\{(I_{i,j} - \overline{I_{i,j}})^2\}$$

Siendo $\overline{I_{i,j}} = E\{I_{i,j}\}$ la media local de la imagen. La varianza también puede ser estimada utilizando una estructura ponderada de vecindad $\eta_{i,j}$ centrada en el píxel que está siendo analizado.

El tamaño de la vecindad $\eta_{i,j}$ debe estar relacionado con la escala de las estructuras previstas en una situación determinada. La varianza local estimada se utiliza como medida de la calidad de la semejanza estructural entre dos imágenes. Para determinarla, se emplearán algunos de sus estadísticos. En primer lugar, la media de la varianza local μV_I se estima como:

$$\hat{\mu}V_I = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} Var(I_{i,j})$$

La desviación estándar de la varianza local se define:

$$\sigma_{V_I} = \left(E\{(Var(I_{i,j}) - \mu V_I)^2\}\right)^{1/2}$$

Y puede estimarse como:

$$\hat{\sigma}_{V_I} = (\frac{1}{MN-1}\sum_{i=1}^{M}\sum_{j=1}^{N}(Var(I_{i,j}) - \mu V_I)^2)^{1/2}$$

Finalmente, la covarianza entre las varianzas de dos imágenes I y J se define como:

$$\sigma_{V_I V_J} = E\{(Var(I_{i,j}) - \mu V_I)(Var(J_{i,j}) - \mu V_J)\}$$

Y se estima como:

$$\hat{\sigma}_{V_I V_J} = \frac{1}{MN-1}\sum_{i=1}^{M}\sum_{j=1}^{N}(Var(I_{i,j}) - \mu V_I)(Var(J_{i,j}) - \mu V_J)$$

Finalmente, se define el índice de calidad basado en varianza local (QILV) entre dos imágenes I y J de la siguiente manera:

$$QILV(I,J) = \frac{2\mu_{V_I}\mu_{V_J}}{{\mu_{V_I}}^2 + {\mu_{V_J}}^2} \cdot \frac{2\sigma_{V_I}\sigma_{V_J}}{{\sigma_{V_I}}^2 + {\sigma_{V_J}}^2} \cdot \frac{\sigma_{V_I V_J}}{\sigma_{V_I}\sigma_{V_J}}$$

En la expresión, el primer término realiza una comparación entre la media de las distribuciones de la varianza local de ambas imágenes. El segundo compara la desviación estándar de las varianzas locales. El último introduce coherencia espacial.

Aunque la expresión de QILV se ha definido de forma semejante a la expresión del índice SSIM, este último es la media de los estadísticos locales de la imagen, mientras que QILV opera con los estadísticos globales de las varianzas locales de las imágenes.

4.5 EVALUACIÓN DE LA CALIDAD DE IMAGEN BASADA EN LA FIDELIDAD DE LA INFORMACIÓN VISUAL

Además de métodos que capturan la pérdida de estructura de la señal, existen otras posibilidades para implementar técnicas *full-reference.* Para lograr mejores predicciones de calidad, se estima que el modelado del sistema visual humano puede ser el planteamiento más adecuado. La premisa subyacente es que la sensibilidad del sistema visual es diferente para distintos aspectos de la señal recibida, como brillo, contraste, contenido frecuencial, interacción entre distintos componentes de la señal, y tiene sentido calcular el peso del error entre las señales de test y de referencia una vez que se han tenido en cuenta las distintas sensibilidades del HVS.

Los métodos de fidelidad de la información visual (*visual information fidelity*, VIF) incorporan modelos estadísticos de todos los componentes del sistema de comunicación: el transmisor, el canal y el receptor, [30] y [32]. Esta aproximación intenta relacionar la fidelidad o calidad de la señal con la cantidad de información que es compartida entre dos señales.

Las imágenes naturales de calidad perfecta pueden modelarse como la salida de una fuente estocástica. Si no existen distorsiones, esta señal pasa a través del canal HVS de un observador humano antes de entrar en el cerebro, que extrae información cognitiva. En el caso de imágenes distorsionadas, se supone que la señal de referencia pasa a través de un canal de distorsión antes de llegar al HVS.

La información compartida se cuantifica usando el concepto de información mutua, una medida ampliamente utilizada en teoría de la información. La medida propuesta en [32] se basa en la cuantificación de dos tipos de información mutua: la información mutua entre la entrada y la salida del canal HVS cuando el canal de distorsión no está presente (información de la imagen de referencia) y la información mutua entre la entrada del canal de distorsión y la salida del canal HVS para la imagen test. Debido a que la información mutua es una medida estadística que únicamente puede ser relacionada vagamente con la percepción humana de la

información, esto establece límites en el cantidad de información relevante que puede extraerse de una señal, siempre que los modelos hipotéticos de la fuente, de la distorsión del canal y de la distorsión del receptor sean precisos.

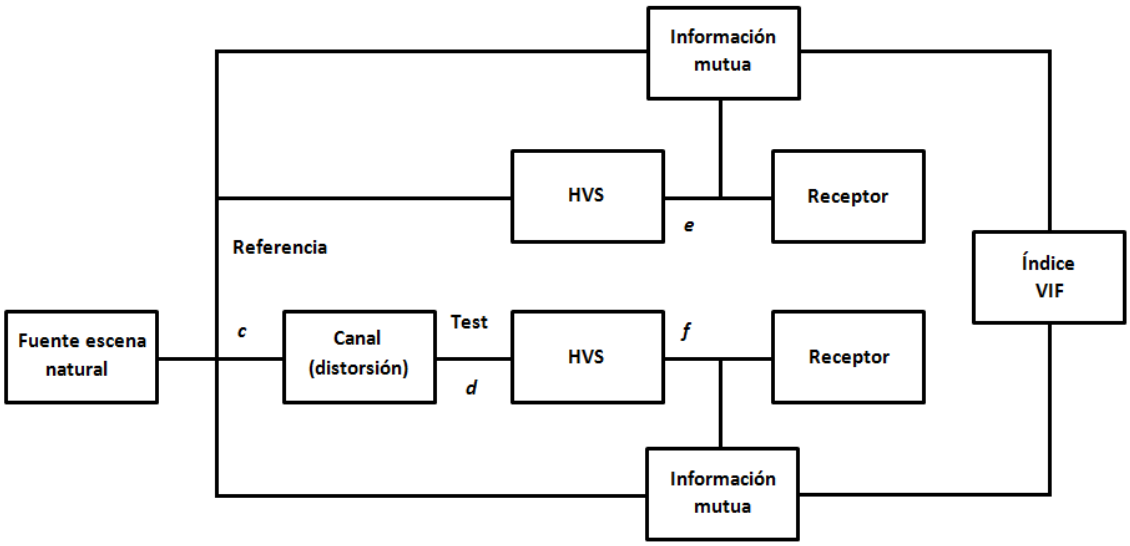


Figura 4.6. Esquema de un sistema VIF

La imagen de referencia se modela como una mezcla Gaussiana (GSM) en el dominio wavelet, que ha demostrado ser capaz de modelar eficientemente las distribuciones marginales no Gaussianas de los coeficientes wavelet de imágenes naturales, además de capturar las dependencias entre las magnitudes de coeficientes wavelet adyacentes. A continuación se presentan los modelos que caracterizan los elementos de un sistema de comunicaciones.

1) Modelo de fuente o transmisor: Sea $\boldsymbol{c}$ una colección de M coeficientes wavelet adyacentes extraídos de una parte de la sub-banda wavelet. Entonces el modelo GSM es sencillo: se caracteriza $\boldsymbol{c}$ como $\boldsymbol{c} = \sqrt{z}\boldsymbol{u}$, donde $\boldsymbol{u}$ es un vector Gaussiano de media nula y $\sqrt{z}$ es una variable aleatoria, escalar e independiente.
2) Modelo de distorsión: El propósito de un modelo de distorsión es describir cómo los estadísticos de una imagen son alterados por el operador de distorsión genérico. Se emplea para caracterizar todas las distorsiones que pueden ocurrir entre las señales de referencia y de prueba, incluyendo distorsiones artificiales como la compresión.
   El modelo asume que la distorsión de una imagen puede ser descrita como la combinación de la atenuación de energía uniforme en dominio wavelet con ruido aditivo: $\boldsymbol{d} = g\boldsymbol{c} + v$, donde $\boldsymbol{c}$ y $\boldsymbol{d}$ son vectores aleatorios extraídos de la misma localización en la misma sub-banda wavelet en las imágenes de referencia y distorsionada, respectivamente. Aquí, g es un factor de ganancia determinístico y escalar que representa una distorsión, mientras que v es ruido blanco Gaussiano de media nula, aditivo y con covarianza $C_v = \sigma_v^2 \boldsymbol{I}$. Aunque este modelo puede ser considerado demasiado general por no considerar ningún tipo específico de distorsión, proporciona una buena aproximación.

3) Modelo de receptor: El modelo de receptor, o modelo HVS, se considera como un "canal de distorsión" que limita la cantidad de información que puede atravesarlo, y su propósito es cuantificar la incertidumbre que el HVS añade a la señal. Por simplicidad, se agrupa todas las fuentes de incertidumbre del HSV en una componente de ruido aditivo, llamado *virtual noise* y se modela como ruido blanco Gaussiano de media nula estacionario en dominio wavelet:

$$\boldsymbol{e} = c + n\,;\ \boldsymbol{f} = d + n$$

Aquí, $\boldsymbol{e}$ y $\boldsymbol{f}$ son vectores de coeficientes aleatorios en la misma sub-banda wavelet en las imágenes de referencia y distorsionada, respectivamente, que denotan las señales a la salida del modelo HVS y de donde el cerebro extrae la información cognitiva. Por último, n es ruido blanco Gaussiano independiente con matriz de covarianza $C_n = \sigma_n^2 \boldsymbol{I}$.

4.5.1 ÍNDICE VIF

La información mutua $I(\boldsymbol{c}|\boldsymbol{e})$ cuantifica la cantidad de información que puede ser extraída de la salida del HVS por el cerebro cuando la imagen test está siendo visualizada. Sin embargo, estamos interesados en la calidad de un par de imágenes test-referencia particular, y no en la calidad media del conjunto de imágenes cuando pasan a través del canal de distorsión. Es, por tanto, razonable sintonizar el modelo de escena natural a una imagen de referencia específica tomando $I(\boldsymbol{c};\boldsymbol{e}|z)$ en lugar de $I(\boldsymbol{c}|\boldsymbol{e})$. De la misma forma para $I(\boldsymbol{c};\boldsymbol{f}|z)$.

Una vez conocidos los modelos estadísticos del transmisor, el canal y el receptor, la información mutua entre $\boldsymbol{c}$ y $\boldsymbol{e}$, y entre $\boldsymbol{c}$ y $\boldsymbol{f}$, es dado por:

$$I(\boldsymbol{c};\boldsymbol{e}|z) = \frac{1}{2}\log\frac{|zC_u + \sigma_n^2\boldsymbol{I}|}{|\sigma_n^2\boldsymbol{I}|} = \frac{1}{2}\sum_{j=1}^{M}\log\left(1 + \frac{z\lambda_j}{\sigma_n^2}\right)$$

$$I(\boldsymbol{c};\boldsymbol{f}|z) = \frac{1}{2}\log\frac{|g^2 zC_u + (\sigma_v^2 + \sigma_n^2)\boldsymbol{I}|}{|(\sigma_v^2 + \sigma_n^2)\boldsymbol{I}|} = \frac{1}{2}\sum_{j=1}^{M}\log\left(1 + \frac{g^2 z\lambda_j}{\sigma_v^2 + \sigma_n^2}\right)$$

En estas expresiones, la matriz de covarianza $\boldsymbol{C}_u = \boldsymbol{Q}\Lambda\boldsymbol{Q}^T$ se ha factorizado, donde Λ es una matriz diagonal cuyas entradas de la diagonal son los autovalores $\lambda_1, \lambda_2, \ldots, \lambda_M$. La información mutua se calcula en cada localización espacial en cada sub-banda de la imagen, utilizando estimaciones de probabilidad máxima de z, g y σ_v. Si asumimos que existe independencia entre los parámetros de distorsión, la información mutua total puede calcularse como una simple suma.

Intuitivamente, la medida de calidad debe relacionar la cantidad de información de la imagen que el cerebro puede extraer de la imagen test en relación con la cantidad de información que puede extraer de la información de referencia. Esto se puede conseguir como la razón entre ambas medidas. Por tanto, finalmente, el índice VIF se define como la relación entre la información mutua sumada.

$$VIF = \frac{I(\boldsymbol{C};\boldsymbol{F}|z)}{I(\boldsymbol{C};\boldsymbol{E}|z)} = \frac{\sum_{i=1}^{N} I(c_i;f_i|z_i)}{\sum_{i=1}^{N} I(c_i;e_i|z_i)}$$

4.5.2 PROPIEDADES DE VIF

1) El índice VIF está limitado por debajo de cero, que sucede cuando $I(\boldsymbol{C};\boldsymbol{F}|z) = 0$ y $I(\boldsymbol{C};\boldsymbol{E}|z) \neq 0$, lo que significa que toda la información sobre la imagen se ha perdido.
2) Cuando la señal no está distorsionada de ninguna forma, entonces $g = 1$ y $\sigma_v^2 = 0$. Esto implica que $I(\boldsymbol{C};\boldsymbol{F}|z) = I(\boldsymbol{C};\boldsymbol{E}|z)$, y VIF es la unidad.
3) Para todos los tipos de distorsión en la práctica, el índice VIF está acotado en [0,1].
4) Un aumento del contraste de la imagen de referencia que no añade ruido, provoca que el índice VIF tome un valor mayor que la unidad, lo que significa que la imagen mejorada tiene mejor calidad que la imagen de referencia. Teóricamente, el aumento del contraste implica mayor SNR a la salida de las neuronas, permitiendo al cerebro mayor capacidad para distinguir objetos es la señal visual. El índice VIF es capaz de incluir esta mejora de la calidad visual. Esta característica lo diferencia de otros métodos tradicionales de medida de calidad.

En las siguientes imágenes, obtenidas de [32], se pueden comprobar las propiedades mencionadas. La figura (a) representa la imagen original que sirve de referencia para comparar con las demás, y consecuentemente se le asigna un índice VIF=1. En la imagen (b) se ha producido un realce del contraste, lo que mejora la calidad de la imagen. En este caso, la imagen recibe un índice VIF=1.10, lo cual es coherente con la propiedad 4 porque la imagen obtenida es de mejor calidad que la imagen original. En las figuras (c) y (d) la imagen de referencia ha sufrido operaciones de emborronamiento y de compresión mediante el algoritmo JPEG, respectivamente.

En ambos casos la calidad de la imagen se ha visto afectada y el valor de VIF será menor que la unidad. Concretamente, el emborronamiento provoca un VIF=0.07 y la compresión un VIF=0.10.

(a) Imagen de referencia VIF=1

(b) Mejora de contraste VIF=1.10

(c) Emborronamiento VIF=0.07

(d) Compresión JPEG VIF=0.10

Figura 4.7. Ejemplo de funcionamiento del índice VIF

4.6 EVALUACIÓN DE LA CALIDAD DE IMAGEN BASADA EN COMPARACIÓN DE HISTOGRAMAS

Una aproximación diferente a la medida de la calidad de imágenes consiste en utilizar medidas de similitud de lógica difusa (*fuzzy*) para comparar los histogramas de imágenes digitales. Comenzaremos presentando algunos conceptos sobre lógica difusa antes de ver cómo se aplican estos conceptos al análisis de imágenes, que se encuentran detallados en [33] y [34].

4.6.1 INTRODUCCIÓN

- **Conjuntos difusos (*fuzzy sets*)**

Un conjunto difuso A en un universo X se caracteriza por X- [0,1], que asocia con cada elemento x de X un grado de pertenencia $\mu_A(x)$ de x en el conjunto A. Vamos a denotar el grado de pertenencia como $A(x)$, y la clase de conjuntos difusos en un universo X como $F(X)$.

- **Medidas de semejanza**

Existen numerosas medidas propuestas para expresar la semejanza entre conjuntos difusos. La definición más común se puede ver como la relación binaria difusa en $F(X)$, tal que $F(X)$ x $F(X) \rightarrow [0,1]$, que cumple las propiedades de:

- Reflexión.
- Simetría.
- Mínimo transitiva.

- **Operadores de lógica difusa**

Las operaciones de la teoría de conjuntos *co* (complemento), ∩ (intersección), ∪ (unión) pueden extenderse a operaciones de conjuntos difusos. Basados en los operadores de la lógica tradicional, negación, conjunción, disyunción, aparecen operadores de *fuzzy logic* como *negator*, *conjunctor*, *disjunctor* o *implicator*.

De estos operadores de lógica difusa se derivan un gran número de clases de medidas de semejanza.

1) Un *negator* es un mapeo $[0,1] - [0,1]$ decreciente N, que satisface las condiciones de borde $N(0) = 1$ y $N(1) = 0$. El operador más popular es el *standard negator* $\mathrm{N_s}$, que se define como $\mathrm{N_s}(x) = 1 - x$para todo x en $[0,1]$.
   El complemento A^c de un conjunto difuso A en X se define: $A^c(x) = 1 - A(x)$ para todo x en X.
2) Un *conjunctor* es un mapeo $[0,1]^2 - [0,1]$ creciente T que satisface las condiciones de borde $T(0,0) = T(0,1) = T(1,0) = T(1,1) = 1$. Los más comunes son:
   - $T_M(x,y) = \min(x,y)$.
   - $T_P(x,y) = x \cdot y$.
   - $T_W(x,y) = \max(0, x + y - 1)$ para todo (x,y) en $[0,1]^2$.

   La intersección $A \cap_T B$ de dos conjuntos A y B en un universo X se define como: $(A \cap_T B)(x) = T\big(A(x), B(x)\big) = \mathrm{Min}[A(x), B(x)]$.
3) Un *disjunctor* es un mapeo $[0,1]^2 - [0,1]$ creciente S que satisface las condiciones de borde $S(1,0) = S(0,1) = S(1,1) = 1$ y $S(0,0) = 0$. Los más populares son:
   - $S_M(x,y) = \max(x,y)$.
   - $S_P(x,y) = x + y - x \cdot y$.
   - $S_W(x,y) = \min(1, x + y)$ para todo (x,y) en $[0,1]^2$.

   La unión $A \cup_S B$ de dos conjuntos difusos A y B en un universo X se define como: $(A \cup_S B)(x) = S(A(x), B(x)) = \mathrm{Max}[A(x), B(x)]$.
4) Un *implicator* es un mapeo $[0,1]^2 - [0,1]$, I, con mapas parciales decreciendo primero y aumentando después, que satisface las condiciones de borde $I(0,0) = I(0,1) = I(1,1) = 1$ y $I(1,0) = 0$. Los implicadores más importantes son:
   - $I_{KD}(x,y) = \max(1 - x, y)$.
   - $I_W(x,y) = \min(1, 1 - x - y)$.
   - $I_R(x,y) = 1 - x - x \cdot y$, para todo (x,y) en $[0,1]^2$.

- **Propiedades relevantes para el procesado de imagen**

A continuación se exponen algunas características que deben cumplir las medidas de semejanza para poder aplicarse a la comparación de imágenes [34]:

- Reflexividad: Si se tienen dos imágenes idénticas, la medida de la semejanza proporciona una salida igual a la unidad.
- Simetría: La salida obtenida por la medida de similitud es independiente del orden en que consideren las dos imágenes de entrada.
- Reacción al ruido: Una medida no debe verse demasiado afectada por el ruido, ya que la imagen distorsionada debe ser similar a la original, y debe disminuir con respecto a un aumento del porcentaje de ruido.
- Reacción al oscurecimiento y a la iluminación: Si se aumenta o disminuye la luminosidad de una imagen con un valor constante, la medida debe generar un valor elevado.

4.6.2 COMPARACIÓN DE HISTOGRAMAS

- **Medidas clásicas de comparación de histogramas**

El histograma de una imagen en escala de grises es un gráfico que muestra la distribución de los diferentes niveles de gris. El valor del histograma de una imagen A en un nivel de gris g es el número total de píxeles en la imagen que tienen el nivel g, y se denota como $h_A(g)$. Así, la expresión del histograma de la imagen A se representa como:

$$h_A(g) = \sum_{(i,j)\in X} \delta(g - A(i,j))$$

Normalmente se emplean mediciones de distancia para determinar cuánto difieren dos histogramas. Como los histogramas pueden considerarse vectores, las métricas más comunes empleadas para histogramas son las basadas en la métrica de Minkowski (L_p), que se utiliza en espacios de vectores. Dos ejemplos son la *distancia Manhattan* y la *distancia Euclídea*, que tienen como expresiones:

$$d_{L1}(h_A, h_B) = \sum_{g=0}^{|G|-1} |h_A(g) - h_B(g)|$$

$$d_{L2}(h_A, h_B) = \sqrt{\sum_{g=0}^{|G|-1} |h_A(g) - h_B(g)|^2}$$

Otra técnica para la comparación de histogramas es la llamada intersección de histogramas. Para dos histogramas $h_A(g)$ y $h_B(g)$ la intersección de histogramas se define como:

$$d_I(h_A, h_B) = \sum_{g \in G} \min(h_A(g), h_B(g))$$

Sin embargo, si A y B tienen el mismo tamaño puede probarse que la suma de $d_{L1}(h_A, h_B)$ y $2d_I(h_A, h_B)$ es igual a una constante. Por consiguiente, no hay una diferencia significativa entre la intersección de histograma y la distancia Manhattan.

- **Aplicación directa de medidas de semejanza a histogramas**

Comparar dos histogramas en el contexto de la teoría de conjuntos difusos es razonable porque el histograma de una imagen puede transformarse en un conjunto difuso en el universo de los niveles de gris [34], dividiendo los valores del histograma para cada nivel de gris por el máximo número de píxeles con el mismo nivel de gris. De esta forma el valor de gris más común consigue el grado de pertenencia 1 en el conjunto difuso asociado al histograma, y cualquier otro valor de gris menos frecuente obtiene un grado de pertenencia más pequeño.

Por consiguiente, un histograma normalizado está en concordancia con la idea intuitiva detrás de un conjunto difuso: el elemento más típico de un universo tiene grado de pertenencia 1, mientras que los elementos menos comunes pertenecen al conjunto difuso en menor medida, que se expresan mediante grados de pertenencia menores que la unidad. De esta manera, se obtiene la siguiente expresión del grado de pertenencia del valor de gris g en el conjunto difuso Fh_A asociado con el histograma $h_A$de la imagen A:

$$Fh_A(g) = \frac{h_A(g)}{h_A(gM)}$$

Con $h_A(gM) = \max_{g \in G} h_A(g)$. Como los histogramas de imágenes digitales pueden ser vistos como conjuntos difusos en el universo del rango de valores de grises, es interesante investigar si las medidas de semejanza introducidas originalmente para comparar dos conjuntos difusos pueden aplicarse a histogramas normalizados.

- **Aplicación de medidas de semejanza a histogramas ordenados**

Las medidas de similitud pueden aplicarse también de otra manera en histogramas asociados a imágenes digitales [34]. Los valores de un histograma pueden ordenarse de manera que el valor de gris menos frecuente se sitúa en la primera posición del histograma y el resto se ordenan de forma creciente. A continuación el histograma es normalizado de igual forma que en el caso anterior; y, por consiguiente, el nivel de gris más frecuente adopta el grado 1 del conjunto asociado al histograma y el resto de grados de pertenencia son menores que la unidad y están dispuestos en orden creciente.

Al contrario que ocurría en el caso de histogramas normalizados, donde las frecuencias de los distintos niveles de gris se comparaban nivel a nivel, en este caso la frecuencia del nivel de gris más recurrente en la imagen A se compara con el nivel de gris más recurrente en la imagen B, la frecuencia del segundo nivel de la imagen A se compara con el de la imagen B, y así

sucesivamente se comparan todas las frecuencias de los diferentes niveles de gris en orden creciente de frecuencia. Puede ocurrir que dos frecuencias de dos valores de gris distintos se comparen, dependiendo de la posición que ocupen dentro del histograma ordenado. Si expresamos el histograma ordenado de una imagen A como o_A, obtenemos la siguiente expresión para el conjunto difuso asociado al histograma ordenado. Desde $i = 1, \dots |G|$, con G el universo de niveles de gris y con $o_A(g) = \max_{g \in G} h_A(g)$.

$$Oh_A(i) = \frac{o_A(i)}{o_A(|G|)}$$

4.7 MEDIDAS DE SEMEJANZA BASADAS EN VECINDAD

En este apartado veremos cómo se realizan medidas de semejanza basadas en vecindad utilizando lógica difusa, continuando con el punto de vista del apartado anterior.

Para ello, se comparan dos imágenes A y B, dividiendo ambas imágenes en partes de 8x8 y se calcula la semejanza entre las dos partes. Para calcular la semejanza se emplean las medidas basadas en píxeles. Algunas de estas medidas, desarrolladas en [35] son las siguientes:

$$S_1(A,B) = 1 - (\frac{1}{MN}\sum_{x \in X} |A(x) - B(x)|^r)^{\frac{1}{r}}$$

$$S_6(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Si la imagen es dividida en P partes y la semejanza entre la parte A_i de la imagen A y la parte B_i de la imagen B se denota por $S(A_i, B_i)$, entonces la semejanza entre las dos imágenes A y B se obtiene mediante la media ponderada de las similitudes en todas las partes:

$$S^n(A,B) = \frac{1}{P}\sum_{i=1}^{P} w_i \cdot S(A_i, B_i)$$

El peso w_i se define como la semejanza entre la homogeneidad h_{A_i} de la parte i en la imagen A y la homogeneidad h_{B_i} de la parte i en la imagen B. La homogeneidad se calcula como la semejanza entre el nivel de gris del píxel con mayor intensidad y el nivel de gris del píxel con menor intensidad, usando una relación de parecido s. Estos parámetros se definen de la siguiente forma:

$$s(x,y) = min(1, \max(0, \frac{6}{5} - 2|x - y|))$$

$$h_{A_i} = s(\max_{(x,y) \in A_i} A(x,y), \min_{(x,y) \in A_i} A(x,y))$$

$$w_i = s(h_{A_i}, h_{B_i})$$

Por último se considera la unión de las medidas de similitud basadas en vecindad y en histogramas para incorporar las características de la imagen de manera óptima. Esta nueva medida de la calidad se consigue simplemente multiplicando ambas componentes:

$$Q_{i,j}(A,B) = S_i^n(A,B) \cdot H_j(A,B)$$

De esta forma se obtiene un índice de calidad de imagen definido matemáticamente y universal, es decir, que no depende de las imágenes que están siendo comparadas, de las condiciones de visualización o de los observadores individuales.

4.8 PASO DE IMÁGENES A VÍDEOS

Después de haber visto distintos tipos de medidas de la calidad de imágenes, la pregunta que se plantea es si dichos métodos pueden aplicarse a imágenes en movimiento o vídeo [30]. A este respecto, existe una gran demanda por parte de la industria de comunicación multimedia de medidas de la calidad de vídeo (VQA – *video quality assessment*).

La forma más sencilla y obvia de implementación de VQA sería aplicar medidas de calidad de imagen a cada *frame* y luego calcular la media de todos los fotogramas. De hecho, SSIM se utiliza de esta forma con buenos resultados.

Sin embargo, dos aspectos importantes de las señales de vídeo no se tienen en cuenta en las implementaciones anteriores:

1) Existen fuertes correlaciones entre dos fotogramas adyacentes que definen estructuras temporales y espacio-temporales.
2) Las señales de vídeo contienen un importante movimiento estructurado.

La forma más común de tener en cuenta las correlaciones temporales es el filtrado temporal o espacio-temporal. Se aplican filtros lineales o bancos de filtros a lo largo de las direcciones espaciales y temporales, y las señales filtradas se normalizan para reflejar efectos perceptuales adicionales, como la función de sensibilidad al contraste CSF. Un ejemplo de VIF adaptado a señales de vídeo se tiene en [36].

El movimiento es uno de los aspectos más importantes de la información de los vídeos. A pesar de esto, pocos algoritmos VQA detectan movimiento explícitamente y emplean la información de movimiento directamente. El uso directo del movimiento es deseable ya que el filtrado temporal no puede capturar totalmente los movimientos. De hecho, el movimiento no crea todas las variaciones de intensidad temporal en vídeos, y la respuesta de los filtros temporales no puede proporcionar la velocidad de movimiento, ya que los atributos de intensidad afectan a la respuesta de los filtros.

Otros métodos más recientes buscan detectar movimiento y convertir la información de movimiento en factores de ponderación en la etapa de agrupación (*pooling*) de la medida de calidad.

Un enfoque diferente implica el uso de estimación del flujo óptico, para guiar de forma adaptativa el filtrado espacio-temporal utilizando bancos de filtros de Gabor tridimensionales. La diferencia de este método es que se selecciona un subconjunto de filtros en cada posición basándose en la dirección y la velocidad de movimiento, de forma que el eje principal del conjunto de filtros se orienta en la dirección de movimiento en el dominio de la frecuencia. La evaluación de la calidad se realiza sólo con los coeficientes calculados por los filtros seleccionados. Las distorsiones puramente espaciales, que se producen dentro de cada fotograma, provocan cambios en los componentes frecuenciales a lo largo del plano y son capturados en las salidas de los filtros. Las distorsiones puramente temporales, distorsiones entre fotogramas, resultan en cambios en el eje a lo largo del cual el plano intersecciona con los filtros de Gabor.

4.9 RESUMEN

	Ventajas	Inconvenientes
MSE/PSNR	- Fácil de calcular. Está libre de parámetros. Eficiente desde el punto de vista computacional. - No necesita memoria, cada muestra se calcula de forma independiente. - Tiene significado físico claro, define la energía de la señal de ruido. - Su uso está muy extendido en distintos campos.	- La forma en que MSE se representa no se corresponde con la percepción humana. - No es demasiado útil como media de calidad. - Dos imágenes con misma energía de ruido pueden tener diferente calidad, pero igual MSE. - Todas las muestras de señal tiene igual importancia y la medida no depende de las relaciones entre muestras.
SSIM	- Evalúa los cambios en la estructura de la imagen. - Los resultados generados son coherentes con la percepción humana. - Proporciona una buena medida para distintos tipos de imágenes.	- Calcula el índice global como la media de la varianza local de las dos imágenes. - No tiene en consideración la no-estacionaridad.
QILV	- Puede emplearse por separado o junto a otros métodos. - Supera las limitaciones de SSIM de no estacionaridad.	
VIF	- Se aproxima a la percepción humana mediante el modelado del sistema visual. - Tiene en cuenta posibles mejoras en la calidad de la imagen, por incremento del contraste	- El modelado de fuente, canal de dispersión y receptor puede ser simple, al no generalizar en ningún tipo de dispersión.
Fuzzy logic	- Combinar medidas de comparación de vecindad y de histogramas permite tener un índice de calidad universal y definido de forma matemática.	- Se basa en complejos desarrollos matemáticos.

Tabla 4.1. Resumen de métodos de evaluación de calidad

CAPÍTULO 5

DISEÑO DEL SISTEMA

En este capítulo nos proponemos exponer los distintos pasos que se han seguido para desarrollar el mecanismo de detección y seguimiento de objetos en movimiento dentro de secuencias de vídeo. Este mecanismo va a imitar el comportamiento de una cámara giratoria que puede moverse dentro de una zona de vigilancia, detectar el movimiento de objetos o personas y seguir su movimiento dentro de esa zona. Por tanto, el sistema implementado debe imitar esas funcionalidades de detección y seguimiento de un único blanco.

Como hemos comentado en capítulos anteriores, para la implementación de nuestro sistema necesitamos dos bloques básicos: el sistema de detección de objetos en movimiento y el sistema de seguimiento de dichos blancos. Para el algoritmo de detección no se empleará ninguno de los métodos descritos en el capítulo 2, sino que se utilizará el método SSIM explicado en el apartado anterior, ya que nos permite realizar la tarea de detección de manera sencilla y proporcionando buenos resultados. Respecto al algoritmo de seguimiento, se plantará el uso de los filtros alfa-beta y Kalman, por ser los más típicos en este tipo de aplicaciones y que menores recursos consumen, con el fin de desarrollar un método lo más sencillo posible.

La herramienta en la que nos vamos a apoyar para el desarrollo de este proyecto es el software Matlab; ya que su alta capacidad de procesado, su amplia versatilidad y la presencia de *toolboxes* de funciones como el Image Processing Toolbox (IPT) y el Computer Vision System Toolbox (CVST), le convierten en una de las mejores alternativas para el tratamiento de imágenes.

En primer lugar, se describe el funcionamiento básico del sistema que se quiere alcanzar en este proyecto. Al ser éste un proyecto continuado a partir del anterior [8], previamente se recuerda el sistema desarrollado en dicho trabajo, con sus funcionalidades tal cual se implementaron en ese momento. A continuación se presentan las modificaciones aplicadas sobre el sistema anterior para dar lugar al sistema definitivo que incluya las capacidades de seguimiento. En último lugar, se expone cómo se han implementado los algoritmos de detección y seguimiento, aplicados de manera adecuada a la aplicación desarrollada.

5.1 FUNCIONAMIENTO GENERAL

El sistema que se pretende diseñar tiene como objetivo detectar y seguir un blanco en movimiento utilizando el algoritmo de detección SSIM y de seguimiento, aplicados de forma conjunta sobre la secuencia de video. Se considera que la cámara se está moviendo siguiendo al objetivo, de forma que suponemos que la imagen completa es el área que la cámara puede barrer y la detección solo se realiza sobre una zona determinada de la imagen. La zona de detección se va a centrar sobre la posición del filtro el cual se estará moviendo continuamente, barriendo el área de vigilancia buscando blancos móviles o siguiendo a un objetivo detectado. Debido a esta circunstancia, las imágenes consecutivas que se comparan están desplazadas una respecto a la otra por lo que es necesario determinar esa diferencia para aplicar el algoritmo SSIM correctamente (separar el movimiento de la cámara entre fotogramas del movimiento de los objetos que se desplazan). Las dos imágenes se van desplazando una sobre la otra pixel a pixel y se calcula el índice SSIM en cada instante, cuando se ha terminado de desplazar las dos imágenes se determina el índice SSIM máximo. El punto con índice máximo es donde las dos imágenes coinciden, se calcula el mapa SSIM en ese instante y se detecta si hay algún movimiento.

El proceso de desplazar las dos imágenes sobre todo el área de vigilancia es un proceso muy costoso y lento, por lo que a partir de la segunda iteración del bucle de procesado este desplazamiento solo se realiza en un entorno alrededor de la última posición del blanco. Si el movimiento del objeto entre las dos imágenes cae dentro de ese intervalo, el movimiento de la cámara se puede corregir al aplicar SSIM, pero si el movimiento es superior (porque es muy rápido o porque el filtro no logra llegar a la posición correcta del objeto) y se sale del intervalo no se puede calcular el desplazamiento correctamente y el mapa SSIM obtenido es erróneo.

5.2 SISTEMA BASE

En el sistema base, que se recupera de [8], el objetivo era detectar el movimiento de objetos móviles con precisión dentro de un entorno con cámara fija, que fuera resistente al ruido y a pequeñas alteraciones. En este sistema se pueden distinguir cuatro fases que se representan en el siguiente esquema. La primera etapa se corresponde con la función de calibrado, que comprende los primeros frames del video. Su cometido es determinar las regiones de la escena en las cuales no vamos a considerar su movimiento.

La segunda fase es la detección, que se realiza durante el resto del video. Se emplea el algoritmo SSIM para comprobar dónde se han producido movimientos. A continuación, se comprueba que las zonas detectadas anteriormente corresponden con objetos en movimiento. Entre todos los puntos obtenidos, se decide cuáles se consideran como un desplazamiento.

Finalmente, la última fase corresponde con un análisis en paralelo al proceso anterior mientras un objeto se encuentra en movimiento. De esta manera es posible comprobar el punto de

partida del movimiento y trayectoria por la escena, y, adicionalmente, podemos detectar movimientos más lentos que no ha sido posible percibir en las fases anteriores.

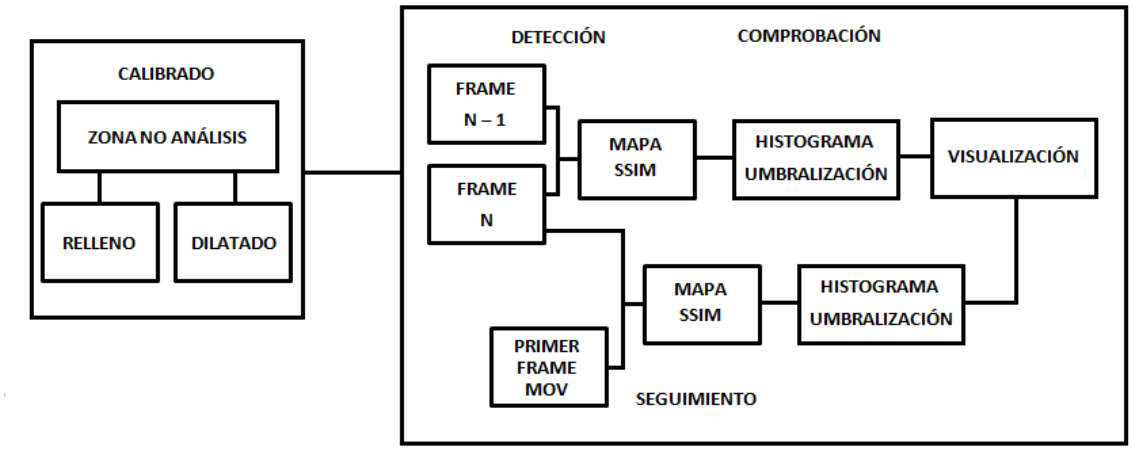


Figura 5.1. Esquema de la aplicación inicial

i) Calibrado

Dentro de una escena es frecuente que haya elementos con movimiento continuo o periódico, como por ejemplo el movimiento de las imágenes en la pantalla de una televisión o un ordenador, el movimiento de un animal en su jaula, etc. Estos movimientos no son relevantes si lo que queremos es detectar la presencia de personas u otro tipo de objetos, pero provocarían que el algoritmo de detección se activase y se produjeran falsas detecciones.

Por tanto, el primer paso del algoritmo consiste en determinar las zonas de la escena donde se producen este tipo de movimientos para descartarlos. Consideramos que previamente al inicio del proceso de detección en sí mismo existe una fase en la que la cámara captura los puntos que no deben considerarse para el tratamiento posterior.

Para ello dedicamos los primeros fotogramas del video a hacer este procesado. El número de fotogramas es variable y puede elegirse en función del grado de precisión que se desee a la hora de eliminar estos movimientos. Como estos movimientos son aleatorios, no es posible determinar con exactitud el cuántos fotogramas deben dedicarse al proceso de calibrado ni tener la certeza de que todos los movimientos de este tipo que puedan producirse hayan sido detectados.

La implementación del calibrado comprende los siguientes pasos:

- Se van leyendo sucesivamente los fotogramas que comprenden la secuencia de video y por parejas se aplica la función SSIM. Como resultado obtenemos el mapa SSIM correspondiente a las diferencias existentes entre ambas imágenes.

- El mapa SSIM es una imagen en escala de grises, cuyos píxeles toman valores entre 0 (negro) y 1 (blanco). El algoritmo SSIM funciona de tal manera que los tonos más claros de la imagen se corresponden con aquellos puntos en los que las dos imágenes comparadas tienen mayor similitud, mientras que las zonas más oscuras se corresponden con aquellas en las que la correspondencia es menor, es decir, se ha producido un movimiento.
- Para poder interpretar mejor los resultados es necesario convertir la imagen en escala de grises en una imagen binaria, para lo cual hay que realizar la elección de establecer un umbral. Para establecer el umbral debemos conocer que valores de los píxeles del mapa SSIM se corresponden con movimientos y cuales con alteraciones de la imagen provocadas por impurezas, fallos debidos a compresión…, para lo cual hacemos uso del histograma de la imagen.
- Como el movimiento que estamos tratando es aleatorio, es posible que zonas adyacentes a las que estamos detectando también sufran movimiento. Para tener esto en cuenta vamos a realizar unas operaciones de dilatación y de relleno de elementos en la imagen binaria. Ambos procedimientos pertenecen a lo que se conoce como operaciones morfológicas, que se encargan de extraer las componentes de la imagen que son útiles en la representación y descripción de la forma de los objetos.
- Realizar las operaciones anteriores genera una imagen formada por 1s (zonas sin movimiento) y 0s (zonas de movimiento). Durante todo el proceso de calibrado vamos sumando entre sí todas estas imágenes que se generan en cada iteración. El resultado final es una matriz formada de números enteros, 0 para los píxeles sin movimiento y mayor que 0 para los píxeles en los que ha habido movimiento al menos una vez. Cuanto mayor sea el valor del número, más veces el píxel de la imagen al que corresponde ha sufrido un movimiento.
- Tenemos que convertir de nuevo la imagen en binaria, para ello podemos establecer como umbral el valor 0 o elegir un número mayor. De esta manera podemos seleccionar un grado de sensibilidad a la hora de determinar las zonas de la imagen que no deben tenerse en cuenta para el procesado posterior. Podemos considerar que los valores más bajos (las zonas de movimiento que menos se ha repetido) no son significativas, porque podrían tratarse de falsas detecciones.
- Finalmente obtenemos la zona de la imagen que no va a ser evaluada en el resto de la aplicación.

ii) Detección

La segunda etapa es la parte central de la aplicación. Consiste en detectar movimiento a partir de dos fotogramas de la secuencia de vídeo utilizando SSIM. Comenzamos después del último fotograma empleado en la fase de calibrado. Este proceso funciona como un bucle y en cada pasada del bucle vamos obteniendo unos fotogramas. El sistema necesita para su funcionamiento capturar en cada ciclo el fotograma del momento actual y un fotograma anterior.

Hay que tener en cuenta que Matlab no permite el uso de cualquier formato de video. Se convierte el formato original producido por la cámara del teléfono móvil (MPEG-4) en un archivo de video en el formato estándar de Matlab, el formato .avi, utilizando un software conversor de formato. En nuestro caso hemos utilizado el programa *Kigo Video Converter*.

- **Adquisición**

Cargamos dicho archivo .avi mediante la función de Matlab ***VideoReader***, que permite leer el vídeo y crea una estructura en la que se recoge toda la información. A continuación se muestra una captura que indica los parámetros importantes de un vídeo que contiene la estructura: duración, tasa y número de *frames*, tipo de imagen, altura y anchura de fotograma, número de bits por píxel…

```
General Settings:
    Duration = 28.0800
    Name = calle7.avi
    Path = C:\Users\Rafael\Desktop\MATLAB R2011b full\Luz
    Tag =
    Type = VideoReader
    UserData = []


Video Settings:
    BitsPerPixel = 24
    FrameRate = 25
    Height = 240
    NumberOfFrames = 702
    VideoFormat = RGB24
    Width = 320
```

Utilizando el objeto creado mediante ***VideoReader*** podemos leer cualquier fotograma del vídeo mediante la función ***read***, indicando únicamente el objeto de origen y el número del fotograma que queremos obtener.

Para poder utilizar el algoritmo SSIM necesitamos a la entrada imágenes en escala de grises, por tanto debemos convertir las imágenes originales en color a imágenes en escala de grises. La conversión se realiza eliminando el color y la saturación de la imagen y manteniendo la luminancia. La función ***rgb2gray*** realiza la transformación mediante la suma ponderada de los componentes R, G y B de la siguiente manera:

$$0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$

En las imágenes siguientes se muestra el resultado de aplicar la función ***rgb2gray*** a una imagen real en color.

Figura 5.2. Transformación de la imagen RGB a escala de grises utilizando la función rgb2gray

Como la imagen original capturada por la cámara tiene utiliza 24 bits por píxel, es decir tiene profundidad de color de 24, la imagen en escala de grises contará con 8 bits, lo que permite representar una totalidad de 256 tonalidades de gris.

- **Algoritmo SSIM**

El siguiente paso consiste en aplicar el algoritmo SSIM para determinar la similitud entre las dos imágenes. En el capítulo 4 se explicaron los conceptos básicos de SSIM y las características del funcionamiento de la función en Matlab.

La función ***ssim*** (esta función es la original creada por Z. Wang, y no pertenece a Matlab) devuelve el mapa SSIM que supone la comparación píxel a píxel entre las dos imágenes. Según está definida la función, los objetos que están en movimiento aparecen representados con los colores más oscuros mientras que el fondo inmóvil se representa con los colores más claros. Es importante tener esto en cuenta ya que está denominación es contraria a la mayoría de las funciones de Matlab de procesado morfológico donde los objetos se suelen representar de color blanco (bit 1) y el fondo de color negro (bit 0).

En las imágenes de la figura 5.3 en la siguiente página, se representa un ejemplo de la aplicación del algoritmo SSIM sobre dos imágenes completas, tal como se capturan directamente de la cámara. Se puede comprobar que el mapa SSIM resultante de aplicar el algoritmo a dos fotogramas detecta el movimiento de dos coches, representando las zonas de la imagen que corresponden a los vehículos en movimiento con píxeles oscuros. También hay que tener en cuenta que el mapa SSIM obtenido es de menor tamaño que las imágenes originales, debido principalmente al tamaño de la ventana gaussiana utilizada. En el caso concreto del ejemplo anterior, las imágenes tomadas de la cámara tienen un tamaño de 240x320 píxeles mientras que el mapa SSIM es de tamaño 230x310 píxeles.

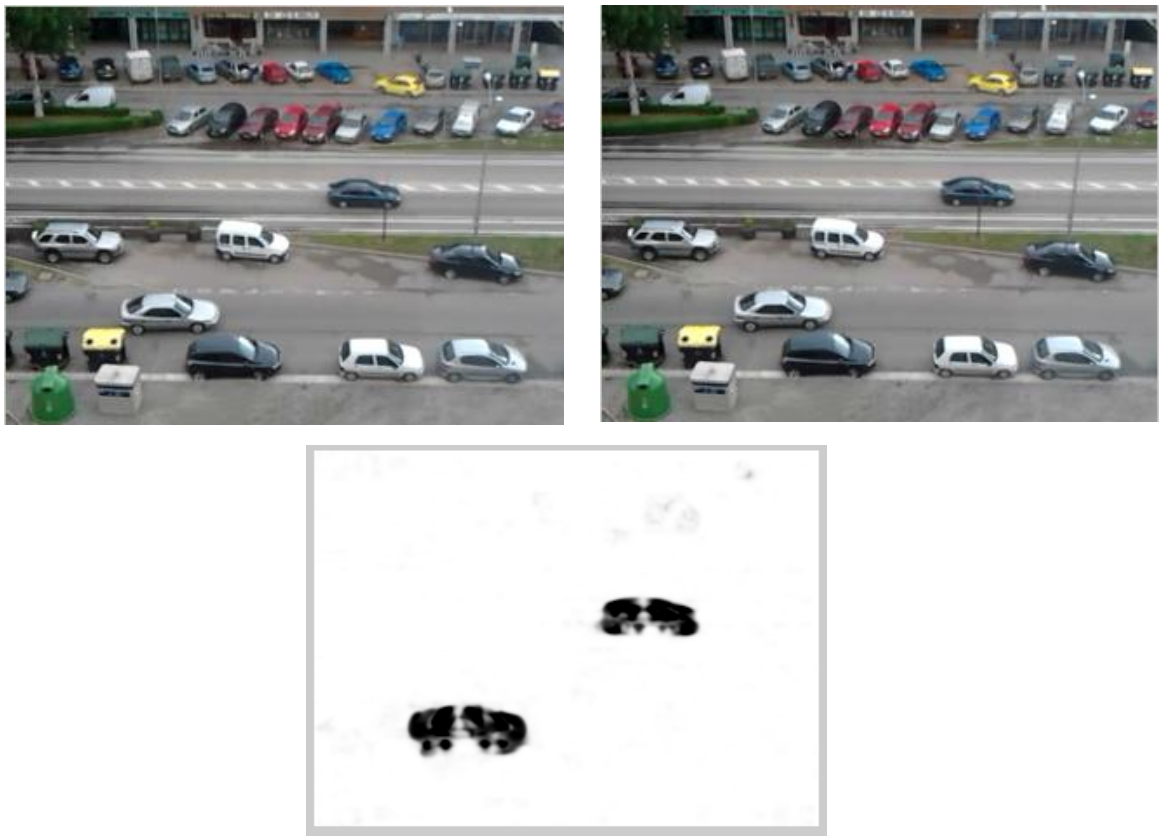


Figura 5.3. Mapa SSIM correspondiente a la comparación de las dos imágenes superiores

- **Suma del mapa SSIM con la máscara.**

Este paso consiste en sumar a la imagen *fondo* producida durante el calibrado el mapa SSIM generado anteriormente para discriminar las zonas que no deben ser analizadas. Dado que en la máscara *fondo* los bits que no deben tenerse en cuenta toman valor 1 (color blanco) y el resto son bits de valor 0 (color negro), las zonas negras del *fondo* no implican ningún cambio mientras que las blancas (valor 1) se suman y provocan que el valor del píxel final sea: 1+x donde x es el valor del píxel en el mapa SSIM. En la representación gráfica de la imagen final sigue el criterio que hemos tomado respecto a las imágenes SSIM, ya que las zonas que no se evalúan aparecen siempre en blanco (como si ni hubiera movimiento).

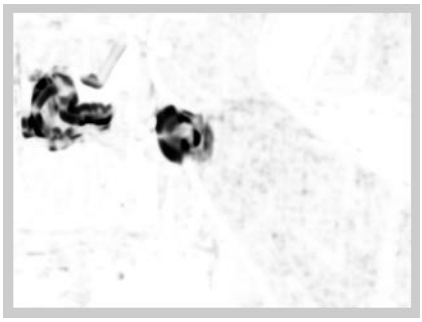
En la figura 5.4 podemos ver cómo funciona lo comentado anteriormente. Las imágenes (a) y (b) corresponden con la fase de calibrado donde el movimiento de las imágenes en la pantalla de un televisor genera una zona, que hemos llamado *fondo,* dentro de la cual no vamos a comprobar el movimiento. En (c) y (d) tenemos dos fotogramas en la fase de detección en los cuales hay un objeto, una pelota en este caso, en movimiento. En (e) se muestra el mapa SSIM generado por los dos fotogramas anteriores y en el que se puede apreciar las zonas de movimiento correspondientes al objeto y a la pantalla de la televisión. Finalmente, (f) presenta el resultado de añadir el *fondo* o máscara generado en el calibrado al mapa SSIM para eliminar el movimiento provocado por la televisión para el análisis posterior.

(a) Fotograma durante el calibrado

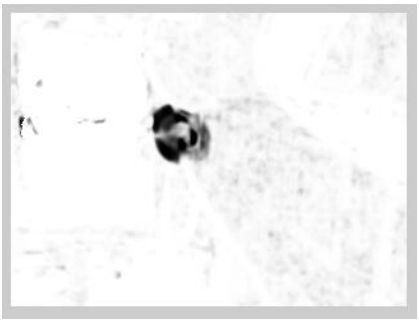
(b) Fondo generado

(c) Fotograma N-1

(d) Fotograma N



(e) Mapa SSIM



(f) Mapa SSIM+fondo

Figura 5.4. Empleo de la máscara del calibrado al análisis final

iii) Comprobación

Después de generar el mapa SSIM es necesario determinar los píxeles del mismo que se corresponden con puntos de movimiento y cuáles se deben desestimar por ser provocados por ruido, cambios de iluminación… El procedimiento es similar al que hemos indicado en la fase de calibrado, salvo que esta vez no vamos a realizar el binarizado de la imagen. Comprobamos la evolución del histograma del mapa SSIM para establecer un umbral adecuado.

Después de determinar el umbral, vamos a marcar sobre la imagen original aquellos píxeles que sufren un movimiento que están por debajo del umbral. Representaremos dichos píxeles de color rojo, y para ello tenemos que recordar que una imagen RGB está formada a su vez por tres imágenes en escala de grises. El color rojo en el espacio RGB normalizado se representa

como [1 0 0] y en imágenes de 8 bits como las que estamos utilizando por [255 0 0]. Entones tenemos que sustituir los valores de los píxeles que queremos modificar por 255 en la primera imagen del conjunto, y por 0 en la segunda y la tercera.

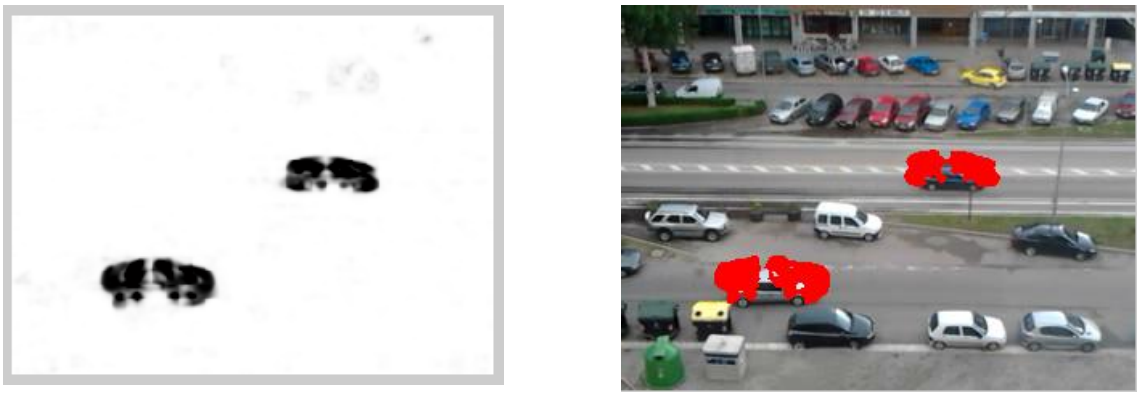


Figura 5.5. Zonas de movimiento sobre la imagen original

iv) Doble detección

La última funcionalidad implementada consiste en determinar el movimiento relativo que ha seguido un objeto o una persona por la escena. El sistema se basa en emplear el algoritmo SSIM sobre el fotograma actual que se está procesando y el primer fotograma en el que el objeto en cuestión aparece en movimiento en la escena. Dicho procedimiento se desarrolla en paralelo al comentado en el apartado anterior.

Con esto se pretende conseguir ser capaces de detectar la presencia de una persona u objeto cuando se ha detenido después de haber estado en movimiento y de haber sido detectado por el mecanismo de la etapa anterior. Adicionalmente, también es posible detectar elementos que se desplazan a una velocidad menor o tienen un movimiento más reducido. Aunque la primera fase no sea capaz de detectarlos porque la diferencia entre dos fotogramas consecutivos no sea suficientemente grande, al estar comparando fotogramas que están más separados la diferencia se hace patente.

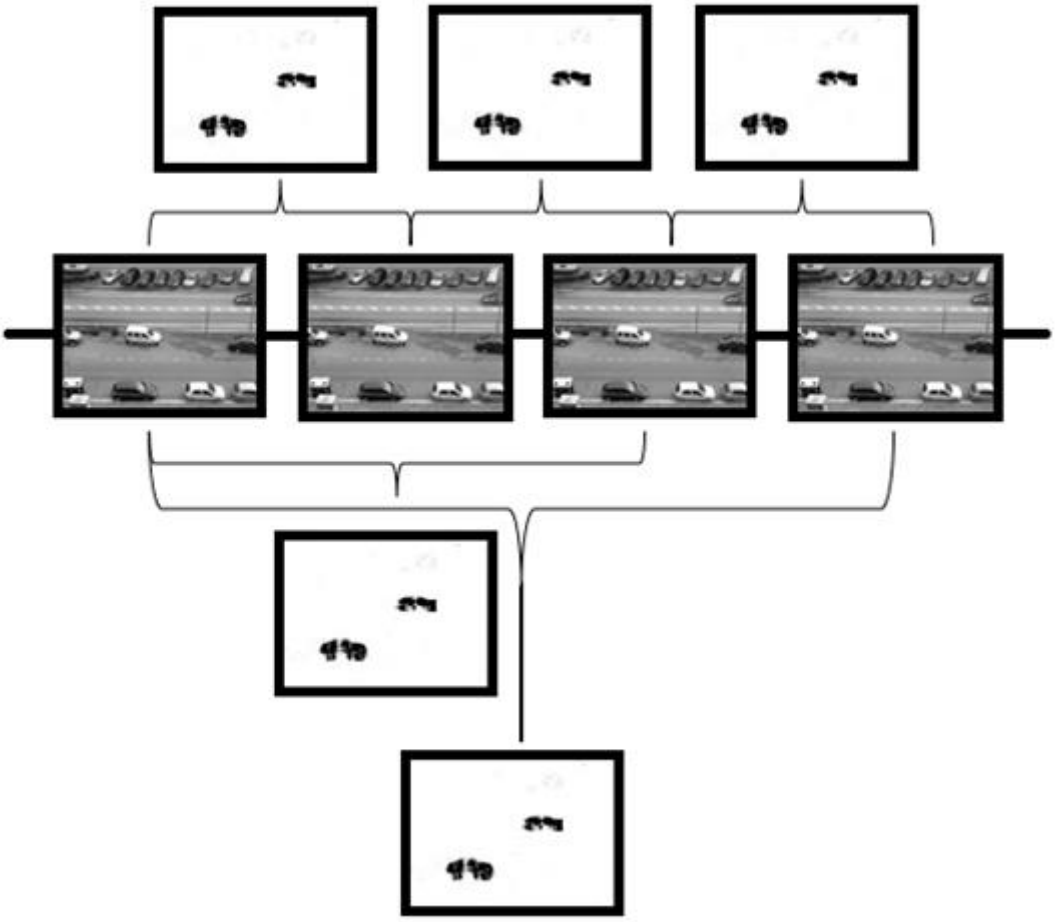


Figura 5.6. Esquema del funcionamiento de la doble detección

5.3 SISTEMA COMPLETO

Sobre el sistema anterior se van a añadir distintas modificaciones para poder adaptarse a las funcionalidades especificadas en el apartado inicial de este capítulo. Por tanto, a continuación se van a describir las modificaciones necesarias dentro de cada uno de las etapas para poder añadir el mecanismo de seguimiento y de ventana de detección móvil. En primer lugar, las etapas de calibrado y de doble detección se eliminan porque son incompatibles con el funcionamiento del nuevo sistema, ya que no se dispone de imágenes de fondo fijas ya que esta va cambiando según se desplaza la ventana. En los siguientes apartados se hará un resumen de cómo se configuran las nuevas etapas y se describen los cambios en cada una.

5.3.1 ETAPAS DEL SISTEMA

Antes de centrarnos en la descripción de cada uno de los apartados del sistema, realizaremos una presentación del funcionamiento general, lo cual nos permitirá tener una visión global del sistema. Podemos distinguir en el sistema diferentes fases.

La primera etapa se corresponde con la inicialización del sistema, que es una fase previa al núcleo principal del sistema. Consiste en la declaración de las variables generales que se van a utilizar durante todo el proceso, así como los parámetros necesarios para controlar los filtros de seguimiento.

Después de la inicialización comienza el bucle de procesado, en el que varios mecanismos se repiten sucesivamente. Durante este bucle es donde se realiza propiamente el análisis de las secuencias de video recibidas. En concreto, el bucle está formado a su vez por cuatro fases:

i) Adquisición: Se extraen dos fotogramas de la secuencia de video y se manipulan adecuadamente para poder llevar a cabo el análisis posterior.
ii) Detección: Se establece la ventana de detección sobre la zona de vigilancia y se aplica el algoritmo SSIM para determinar qué elementos se están desplazando respecto a las dos imágenes comparadas.
iii) Seguimiento: Al elemento detectado se le aplica un algoritmo de seguimiento para tener determinada y controlada su posición en todo momento, y para centrar la ventana de detección sobre ese objetivo.
iv) Visualización: Se representa sobre el fotograma actual (el cual se va a mostrar) toda la información de interés referente a la posición del filtro, la zona de detección y los elementos en movimiento.

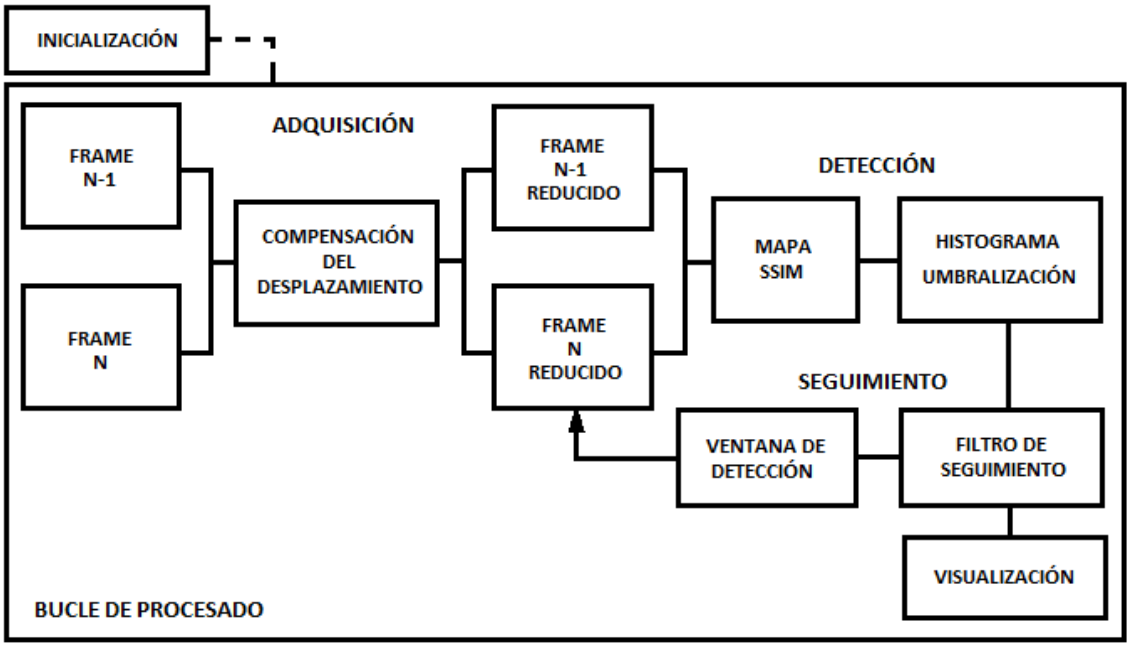


Figura 5.7. Esquema del sistema final

5.3.2 ADQUISICIÓN

Después de la inicialización del sistema, la primera etapa del bucle consiste en la adquisición de las imágenes sobre las que se va a trabajar. El sistema funciona a partir de una secuencia de video grabado, pero las operaciones las realiza sobre dos imágenes fijas o *frames*; por tanto, necesita para su funcionamiento capturar en cada ciclo del bucle el fotograma del instante actual y un fotograma anterior. Hasta aquí, el proceso de adquisición es similar al que se realiza en el sistema básico, detallado en el apartado anterior, extrayendo los *frames* del vídeo y pasándolos a una imagen de escala de grises; a continuación vienen las novedades.

Recordamos que el objetivo de este proyecto es simular el funcionamiento de un sistema de vigilancia con cámara móvil que puede girar para cubrir un área de vigilancia. Con una cámara fija (como es nuestro caso) esto se consigue considerando que la imagen obtenida por la cámara móvil ficticia se corresponde sólo con una parte del fotograma total capturado. Por tanto, el siguiente paso consiste en extraer del *frame* una imagen reducida, correspondiente a la ventana de detección, que es dónde se aplicará el algoritmo de detección y, consecuentemente, cualquier movimiento que ocurra fuera pasará totalmente desapercibido. Esta ventana estará definida por sus dimensiones (anchura y longitud) y por su punto central, que se sitúa sobre la posición indicada por el filtro de seguimiento (en siguientes apartados se verá cómo se establece dicha posición).

Una vez que se tienen las versiones reducidas de las imágenes para los instantes actual y anterior, se dispone de dos imágenes de igual tamaño pero localizadas en distintas posiciones sobre las imágenes originales de las que se obtuvieron, por lo que sus fondos fijos no van a coincidir. Esto hace necesario determinar el desplazamiento entre ambas imágenes para poder solaparlas correctamente y detectar los elementos en movimiento. Para conseguir este objetivo se han planteado dos posibles opciones diferentes que permitan corregir el desplazamiento que se produce entre las dos imágenes.

- **Opción 1: El desplazamiento se conoce**

El primer procedimiento consiste en calcular el desplazamiento relativo entre las dos imágenes, partiendo de la base de que conocemos dónde están situadas cada una sobre su fotograma inicial mediante las coordenadas del filtro de seguimiento. Así pues, en primer lugar se calcula la distancia entre los centros de la imagen actual y anterior (los centros se corresponden con la posición estimada generada por el filtro) restando sus componentes, para obtener el vector de desplazamiento. Dependiendo de la dirección del vector los ajustes que deben realizarse sobre las dos imágenes, para obtener las zonas que se solapan, variarán.

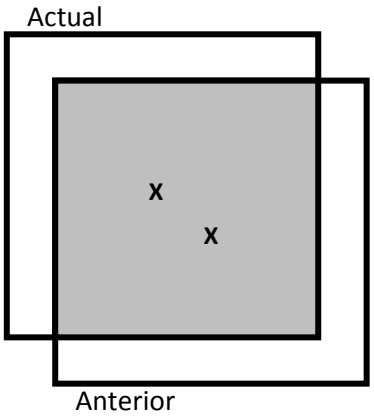


Figura 5.8. Ejemplo de desplazamiento entre imágenes consecutivas

En la figura anterior se ilustra un ejemplo en el que la imagen actual se mueve hacia arriba a la izquierda respecto a la imagen anterior, con el centro de cada una identificado como **x** que se corresponde con la posición del filtro en cada instante. Conociendo las coordenadas de los dos centros (por ejemplo, imagen anterior [80,60] y actual [50,40]), se puede calcular el desplazamiento producido: en el ejemplo, -30 en el eje horizontal y -20 en el eje vertical (recordar que en Matlab el origen de coordenadas está en la esquina superior izquierda). A continuación se extrae de cada imagen la región de solapamiento (zona sombreada), para la imagen actual en el eje vertical se toma desde el píxel 20 hasta el final, y en el eje horizontal desde el píxel 30 hasta el final. Para la imagen anterior en el eje vertical se toma desde el principio hasta 20 píxeles antes del final, y en el eje horizontal desde el principio hasta 30 píxeles antes del final. Después de realizar este proceso, se obtienen dos imágenes reducidas de igual tamaño y con el mismo fondo fijo.

En caso de que el desplazamiento se produzca en otra dirección cualquiera, los pasos a seguir son similares. Hay que tener en cuenta la distinción que existe entre la posición que marca el filtro, que suele ser un número real, y la posición que este ocupa en la imagen (el píxel exacto), que es un número entero, para que el desplazamiento se realice de forma adecuada.

- **Opción 2: El desplazamiento se desconoce**

Este procedimiento consiste en desplazar una imagen sobre otra y aplicar el algoritmo SSIM sobre la parte solapada y luego seleccionar el desplazamiento que ha producido un índice SSIM mayor (que se corresponde con las imágenes de mayor similitud). Sin embargo, este procedimiento supone aplicar el algoritmo SSIM un número elevado de veces, lo cual es computacionalmente contraproducente, así pues se opta por aplicar una versión simplificada del algoritmo, que aunque menos preciso, nos servirá para lograr este objetivo (la forma completa del algoritmo SSIM se explica en el siguiente apartado, relativo a la detección).

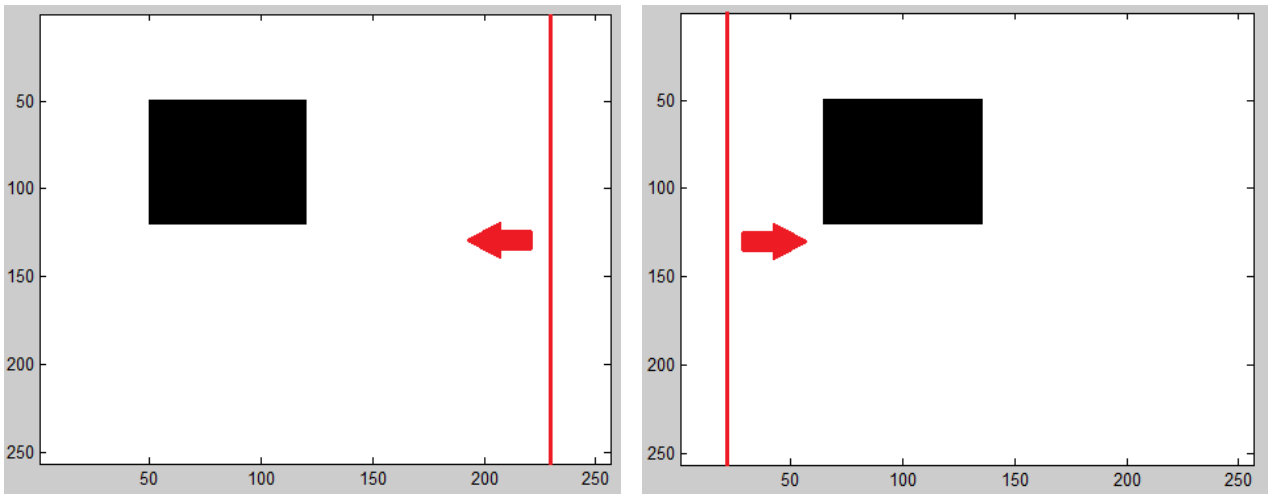
Como decíamos en el capítulo referente a calidad de imagen, el índice SSIM utiliza de forma conjunta tres variables: media, desviación estándar y covarianza. Experimentalmente

comprobamos que empleando sólo las medias y varianzas se obtienen resultados adecuados. Así, evitamos tener que calcular la covarianza repetidamente entre dos imágenes que van cambiando continuamente, lo cual ralentizaría la ejecución. La expresión final queda de la siguiente forma:

$$\text{SSIM simplificado: } \frac{(2\mu_x\mu_y + C1)\cdot(2\sigma_x\sigma_y + C2)}{(\mu_x{}^2 + \mu_y{}^2 + C1)\cdot(\sigma_x{}^2 + \sigma_y{}^2 + C2)}$$

Entonces el proceso se reduce a calcular por separado la media y varianza de las dos imágenes y, dentro de un bucle que realice el desplazamiento, aplicar la fórmula anterior sobre las zonas solapadas y calcular el desplazamiento que produce un valor máximo. Posteriormente, aplicando este desplazamiento podemos disponer de las imágenes correctamente solapadas para poder detectar cualquier movimiento que se haya producido, de manera semejante a como se explica en la opción anterior. Por último, hay que tener en cuenta que, según el desplazamiento (dependiendo de la velocidad del móvil detectado) entre los dos instantes de tiempo, las imágenes resultantes variarán en tamaño y que en la práctica no se realiza detección en toda la ventana de seguimiento, sino que se realiza en un espacio menor.

A modo de ejemplo ilustrativo, hemos creado dos imágenes con un fondo blanco y un bloque negro desplazado hacia la derecha entre ambas imágenes. Se desplazan una imagen sobre la otra tomando porciones cada vez más pequeñas de cada imagen pero siempre del mismo tamaño y comparándolas entre sí, en la dirección que indican las flechas en cada imagen. En la figura 5.9(a) se representa la imagen original y en 5.9(b) se tiene el bloque negro desplazado hacia la derecha 15 pixeles. Se muestra cómo se van reduciendo las imágenes durante el proceso de solapamiento. En cada desplazamiento se calcula el SSIM según la fórmula comentada anteriormente. Finalmente se representan en la figura 5.10 todos los valores calculados y se comprueba la posición del máximo, que en este caso en 15, que se corresponde con el desplazamiento que han sufrido las dos imágenes.



(a) *(b)*

Figura 5.9. Ejemplo de solapamiento entre imágenes

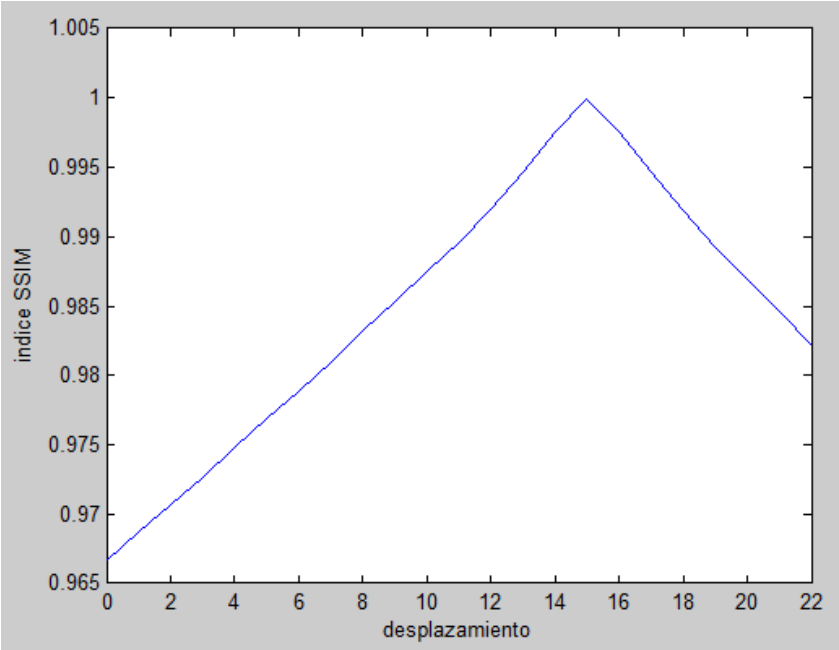
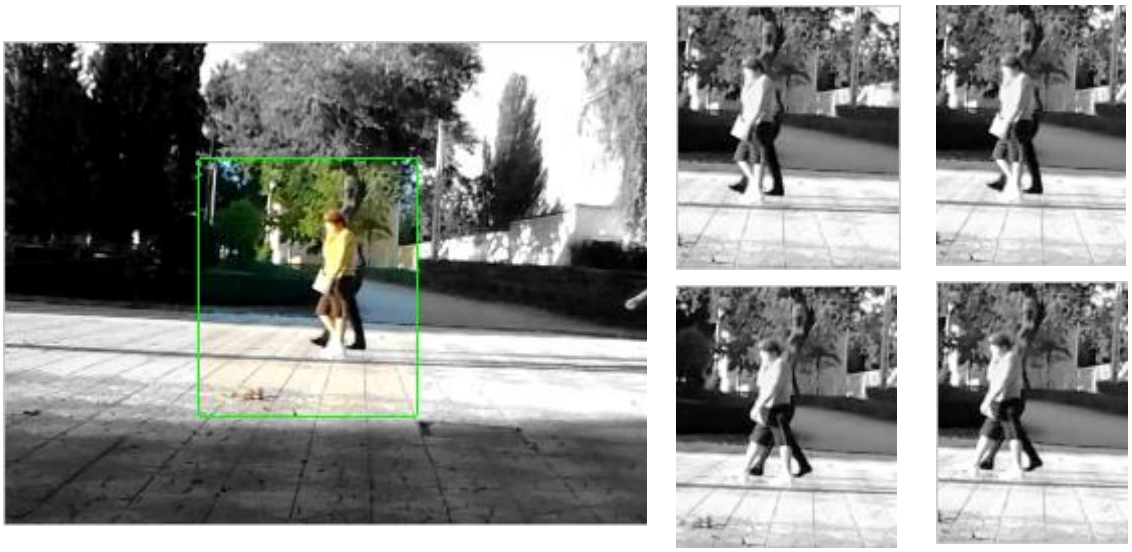


Figura 5.10 Representación del índice SSIM en función del desplazamiento

Para que el ejemplo fuera más claro, sólo se ha considerado un desplazamiento horizontal. En la práctica, el desplazamiento entre las dos imágenes puede darse en diagonal, y en consecuencia hay que realizar un barrido en horizontal y vertical de forma conjunta, para analizar todo el área en el que se haya podido producir el movimiento.

Después de plantear las dos opciones para corregir el desplazamiento entre las dos imágenes, podemos comparar los dos procedimientos para elegir el más adecuado. En el caso de la segunda opción, si se hace un barrido de 20 pixeles en horizontal y 20 en vertical para localizar el desplazamiento correcto, la fórmula de SSIM simplificado se aplica 20x20= 400 veces. Por su parte, el SSIM simplificado consta de sumas, multiplicaciones y división (el cálculo de medias y varianzas se hace fuera de este bucle), frente a una única resta que se debe hacer en la opción 1 para calcular el vector de desplazamiento. En vista de lo anterior, es evidente que la opción 1 es mucho más rápida que la opción 2 y los resultados son igual de precisos. Sin embargo, en una situación real no vamos a poder saber cuánto se ha desplazado el filtro de un instante al siguiente, por lo que la opción 2 es la que más nos conviene de cara a un escenario real.

En la siguiente figura se muestra un ejemplo de aplicación real de la compensación del desplazamiento. En 5.11(a) se tiene el fotograma completo y la zona enmarcada donde se realiza la detección. En 5.11(b) aparecen las imágenes dentro de esa ventana en el instante anterior (arriba) y actual (abajo). Por último, en 5.11(c) se muestra las imágenes reducidas (anterior y actual) obtenidas a partir de las anteriores, utilizado el procedimiento mencionado. Se observa que el tamaño de estas imágenes es menor y que el fondo aparece fijo en ambas.



(a) *(b)* *(c)*

Figura 5.11. Ejemplo de compensación del desplazamiento en un caso real

5.3.3 DETECCIÓN

El siguiente paso consiste en aplicar el algoritmo SSIM de detección a las dos imágenes reducidas para localizar los elementos en movimiento dentro de la ventana de detección. El procedimiento que se realiza en este paso es similar al explicado en el apartado equivalente del sistema básico, con algunas consideraciones que se comentan a continuación.

Figura 5.12. Ventana de detección en dos instantes consecutivos y su mapa SSIM

En la figura 5.12 tenemos un caso de aplicación de nuestro sistema, según el cual las dos primeras imágenes se corresponden con las imágenes reducidas obtenidas según el proceso explicado en el apartado 5.3.2 de adquisición. Partiendo de la región del fotograma completo que se corresponde a la ventana de detección, se han desplazado ambas imágenes hasta que

se solapan totalmente. Se puede comprobar que el desplazamiento se realiza correctamente porque el fondo en las dos primeras imágenes es el mismo. La imagen de la derecha representa el mapa SSIM resultante de la comparación.

Después de generar el mapa SSIM es necesario determinar los píxeles del mismo que se corresponden con puntos de movimiento y cuáles se deben desestimar por ser provocados por ruido, cambios de iluminación… Comprobamos la evolución del histograma del mapa SSIM para establecer un umbral adecuado, en función de su valor (recordamos, que es un valor real entre 0 y 1). Mediante el análisis de varios histogramas en diferentes situaciones, se establece que un valor suficiente para establecer un umbral que separe entre elementos móviles y fijos es 0,5. Como resultado de aplicar el umbral sobre el mapa SSIM se genera una imagen binaria, que se utiliza como patrón para marcar sobre la imagen original en color rojo aquellos píxeles que sufren un movimiento.

Por las circunstancias comentadas también en el apartado anterior, el mapa SSIM obtenido tiene un tamaño mucho menor que los fotogramas del video obtenido por la cámara fija, y como queremos situar uno sobre otro para visualizar el desplazamiento debemos añadir el número adecuado de ceros por la izquierda del mapa para que coincidan adecuadamente. El resultado de añadir un bloque de ceros antes del mapa SSIM y la visualización correspondiente sobre el fotograma completo se puede observar en la figura 5.13.

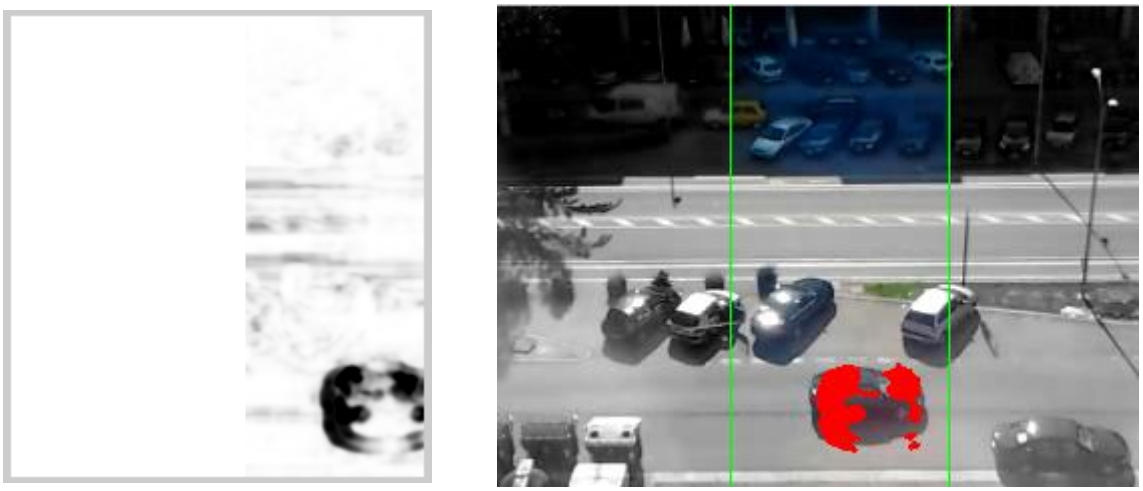


Figura 5.13. Zonas de movimiento sobre la imagen original

Como nuestra aplicación se centra en el seguimiento de un único elemento desplazando la cámara, el siguiente paso consiste en determinar un único punto detectado para utilizarlo como entrada del filtro de seguimiento, cuya función dentro del sistema del filtro será actuar como la posición medida contaminada con ruido. Utilizando la imagen binaria obtenida, se aplica la función ***bwconncomp***, que encuentra los elementos conectados de una imagen binaria. Una vez que tenemos los elementos de la imagen binaria, podemos extraer características de ellos empleando la función ***regionprops***. Con esta función es posible conocer

propiedades como el área, el perímetro, el diámetro equivalente, el centroide, etc. Vamos determinado el área de cada uno de los elementos, y elegimos como el elemento a seguir aquel con área mayor, del cual obtenemos su centroide. Al hallar el centro, obtenemos un pixel y una posición concreta que llevar al filtro.

5.3.4 SEGUIMIENTO

Durante la etapa de seguimiento se ejecuta uno de los filtros de seguimiento propuestos para mantener localizados los blancos en todo momento y para tener un punto de referencia sobre el cual producir el movimiento de la cámara. Es decir, el filtro va a determinar la posición de la ventana de vigilancia en la cual se produce la detección de movimiento.

Dentro de la etapa de seguimiento podemos distinguir varias fases en función de la presencia o no de un elemento en movimiento. Durante la primera ejecución del bucle de procesado se parte del supuesto de que no hay ningún objeto en movimiento, por lo que el algoritmo está en *fase de inicio* y la ventana de seguimiento se mueve desde la posición central del área de vigilancia derecha a izquierda. Cuando el recuadro llega a los límites del área, cambia el sentido del movimiento en la otra dirección. Esto se mantiene hasta que se detecta algún movimiento.

Cuando se detecta el movimiento de un objeto, la etapa de detección proporciona el centroide de ese elemento, actuando como la posición medida y se inicia la *fase de filtrado* en la que se ejecuta propiamente el algoritmo de seguimiento correspondiente (filtro alfa-beta o filtro de Kalman). Después de ejecutar el filtro se obtiene la predicción de posición que es la estimación de la posición del elemento en movimiento y que, además, servirá para centrar la ventana de seguimiento en la próxima iteración del bucle.

Figura 5.14 Filtro de seguimiento y su ventana asociada, centrado sobre el blanco detectado

Una de las utilidades del filtro de seguimiento es estimar la trayectoria del blanco con oclusión. Es decir, si el blanco pasa por detrás de un elemento fijo que bloquea su visión, conociendo su trayectoria previa con el filtro se puede predecir cuál será su velocidad detrás del obstáculo y en qué momento volverá a ser visible. Sin embargo, cuando dejamos de ver la posición del blanco durante un tiempo prolongado suponemos que el blanco se ha perdido (bien porque se ha detenido, ha salido del área de vigilancia o dejamos de percibir su movimiento) y pasamos a la *fase de reseteo*. En esta situación, se vuelve a la *fase de inicio*.

Por último se ejecuta la *fase de corrección*, durante la cual se limita el movimiento del filtro (la posición estimada) a los bordes del área de vigilancia; así se mantiene constante el tamaño de la ventana de detección y ajustado al borde del área de vigilancia en caso de que el blanco esté cerca de los límites. Cuando la posición del filtro se acerca a los bordes derecho e izquierdo (mitad del ancho de la ventana) se limita la coordenada horizontal, y cuando se acerca a los bordes superior e inferior se limita la coordenada vertical.

5.3.5 VISUALIZACIÓN

La última fase del bucle de procesado corresponde en reproducir mediante imágenes los resultados obtenidos, los cuales ya hemos podido ver en las figuras anteriores. A modo de resumen indicamos cómo se han representado los diferentes pasos realizados.

Después de aplicar el umbral sobre el mapa SSIM se genera una imagen binaria, que se utiliza como patrón para marcar sobre la imagen original aquellos píxeles que sufren un movimiento. Representaremos dichos píxeles de color rojo, y para ello tenemos que recordar que una imagen RGB está formada a su vez por tres imágenes en escala de grises. El color rojo en el espacio RGB normalizado se representa como [1 0 0] y en imágenes de 8 bits como las que estamos utilizando por [255 0 0]. Entones tenemos que sustituir los valores de los píxeles que queremos modificar por 255 en la primera imagen del conjunto, y por 0 en la segunda y la tercera.

Para representar la ventana de detección se ha decidido mantener esa zona en concreto de la imagen total en color con un recuadro verde que marca sus límites, mientras que las partes que quedan fuera de la ventana se muestran como imagen en escala de grises. De igual forma que actuamos para generar los píxeles rojos, para crear el recuadro verde en la imagen RGB se sustituyen los píxeles correspondientes por 0 en la primera imagen del conjunto, 255 en la segunda y 0 en la tercera imagen del grupo. Para lograr que el resto de la imagen esté en blanco y negro (escala de grises) hay que recordar que el color gris se encuentra en el vértice que une el color [0 0 0] con el [1 1 1], es decir los tres valores deben ser iguales. Entonces, lo que se hace es igualar las componentes entre las tres imágenes del conjunto RGB.

5.4 ALGORITMO SSIM EN MATLAB

Para concluir con este capítulo dedicado al diseño y la implementación de la aplicación, vamos a explicar cómo se maneja el algoritmo SSIM que hemos utilizado en nuestro trabajo. El código de Matlab que empleamos pertenece al desarrollador del método, el Dr. Zhou Wang de la universidad de Waterloo, que dispone de una versión del algoritmo de SSIM implementado en Matlab en [37]. A modo de recordatorio, el índice de semejanza estructural (SSIM) es un método para medir la semejanza y entre dos imágenes. También pude ser visto como una medida de la calidad de una de las imágenes que están siendo comparadas, considerando que la segunda imagen es de calidad perfecta.

El algoritmo permite disponer de varios argumentos de entrada. Los dos primeros se corresponden con las dos imágenes que van a ser comparadas. Opcionalmente, podemos elegir los otros parámetros: *L*, el rango dinámico de las imágenes de entrada; *window*, la ventana local dentro de la cual se realiza el cálculo de los parámetros estadísticos (media, desviación estándar y correlación); y *K*, que determina el peso de las constantes *C1*, *C2* y *C3* en la fórmula del índice SSIM. Dichas constantes se definen como:

$$C1 = (K1 \cdot L)^2 \qquad C2 = (K2 \cdot L)^2 \qquad C3 = C2/2$$

Los valores que están predeterminados por el programa para estos parámetros son: El rango dinámico de las imágenes (*L*) es 255; la ventana que se emplea es un filtro simétrico paso bajo Gaussiano de tamaño 11x11 y desviación estándar 1.5; y las constantes *K* toman como valor K1=0.01 y K2=0.03.

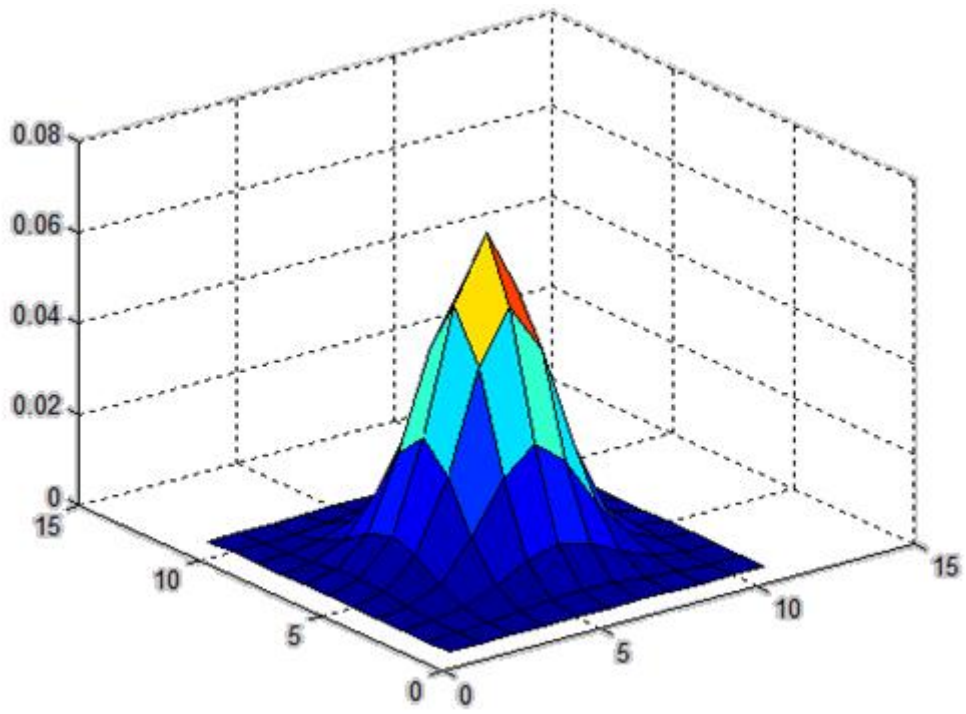


Figura 5.15. Representación de la ventana deslizante Gaussiana

La función devuelve dos valores, el índice SSIM (*mssim*) y el mapa SSIM (*ssim_map*). La variable *mssim* recoge el valor medio del índice SSIM obtenido al comparar las dos imágenes. Como ya hemos comentado anteriormente, si una de las dos imágenes se considera que es de calidad perfecta, entonces *mssim* es la medida de la calidad de la otra imagen. Por otro lado, *ssim_map* es la imagen que corresponde al índice SSIM en cada píxel de la imagen. El mapa generado tiene un tamaño menor al de las imágenes de entrada, y depende del tamaño de la ventana empleada y del factor de submuestreo.

Antes de aplicar el algoritmo SSIM tal y como se expusimos en el capítulo anterior, hay que adaptar previamente la escala de la imagen; es necesario llevar a cabo un submuestreo, para el cual hay que seguir los siguientes pasos:

1) Tomar $f = \max(1, round(N/256))$, donde N es el número de píxeles de la altura o anchura de la imagen.
2) Calcular la media local en una ventana de fxf píxeles, y realizar un proceso de submuestreo en la imagen por un factor de 3.
3) Aplicar el algoritmo SSIM.

El código de Matlab para realizar este procedimiento consiste en:

```
f = max(1,round(min(M,N)/256));
if(f>1)
lpf = ones(f,f);
lpf = lpf/sum(lpf(:));

img=imfilter(img,lpf,'symmetric','same');
img = img(1:f:end,1:f:end);
end
```

Aplicado a nuestro caso tenemos imágenes de 240x320 píxeles, entonces mediante la fórmula $f = \max\big(1, round(240/256)\big) = 1$. Vemos que el resto de pasos (media y submuestreo) no van a tener ningún efecto. Pero si por ejemplo, las imágenes fueran de 512x512, tendríamos: $f = \max\big(1, round(512/256)\big) = 2$. Entonces a la imagen se aplica una ventana 2x2:

```
lpf =

    0.2500    0.2500
    0.2500    0.2500
```

Finalmente, de la imagen filtrada se toma una muestra de cada dos, es decir, se submuestrea por un factor de 2.

5.5 IMPLEMENTACIÓN DE LOS FILTROS EN MATLAB

5.5.1 Filtro de Kalman

En el capítulo correspondiente a los algoritmos de seguimiento, se introdujeron la formulación y las características básicas del filtro de Kalman. En este apartado se van a repasar algunas de sus fórmulas y se adaptaran para aplicarlas en Matlab. Comenzamos recordando que el filtro para una aplicación de seguimiento de objetos en movimiento busca predecir la siguiente posición (el siguiente estado), en función de los anteriores. Para ello se basa en dos etapas diferenciadas: Predicción y actualización.

La fase de predicción usa la estimación del estado del instante anterior para producir una estimación del estado del instante actual. Este estado predicho en ocasiones se conoce como estimación *a priori* porque, aunque es una estimación del estado en el instante actual, no incluye información de observación del instante actual. En la fase de actualización, la predicción *a priori* se combina con información de observación para refinar el estado estimado. Esta mejora en la estimación se llama estimación del estado *a posteriori.*

El primer paso consiste en definir el modelo de estado (cómo evoluciona la posición del blanco con el tiempo) más apropiado, dependiendo de la aplicación específica a que se quiera dedicar, ya que no es lo mismo el movimiento de un objeto balístico, un avión o un peatón. Nuestro sistema está dedicado a detectar el movimiento de personas o vehículos en exteriores que se desplazan con normalidad, por lo que vamos a asumir que su movimiento será aproximadamente rectilíneo. Entonces, modelamos el estado X según la posición p y la velocidad v, (no confundir la posición p con la matriz de covarianza P). En un movimiento con velocidad y aceleración, la posición se expresa mediante la ecuación MRUA (*Movimiento Rectilíneo Uniformemente Acelerado*):

$$\mathrm{p_t} = \mathrm{p_{t-1}} + \mathrm{v_{t-1}T} + \frac{1}{2}\mathrm{a_t T^2}$$

Por otra parte, la velocidad se expresa mediante la siguiente ecuación:

$$\mathrm{v_t} = \mathrm{v_{t-1}} + \mathrm{a_{t-1}T}$$

En el capítulo de algoritmos de seguimiento expresábamos el estado de un sistema de la forma siguiente:

$$\overline{X_t} = \mathrm{AX_{t-1}} + \mathrm{BU_t} + \varepsilon_\mathrm{x}$$

Para mantener la coherencia con lo anterior, expresamos las ecuaciones de posición y velocidad en forma matricial para definir las variables A y B:

$$\overline{X_t} = \begin{bmatrix} \mathrm{p_t} \\ \mathrm{v_t} \end{bmatrix} = \begin{bmatrix} 1 & \mathrm{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{p_{t-1}} \\ \mathrm{v_{t-1}} \end{bmatrix} + \begin{bmatrix} \mathrm{T^2}/2 \\ \mathrm{T} \end{bmatrix} \mathrm{a_t} + \varepsilon_\mathrm{x}$$

El error en el estado (ε_x, o ruido del proceso w) tiene una matriz de covarianza (Q o E_x) que está formada por el cuadrado de la desviación estándar o varianza en la posición, la varianza de la velocidad, y la desviación estándar de la posición y la velocidad de la siguiente manera:

$$w_t = \varepsilon_x = Q = E_x = \begin{bmatrix} \sigma_p^2 & \sigma_p\sigma_v \\ \sigma_v\sigma_p & \sigma_v^2 \end{bmatrix}$$

La desviación estándar de la posición y de la velocidad depende de la aceleración y de los términos de la matriz $B = [T^2/2 \ T]T$, donde el primero se refiere a la posición y el segundo a la velocidad. Entonces, multiplicando los términos correctamente tenemos que la matriz de covarianza queda de la forma:

$$\varepsilon_x = E_x = \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix}$$

En segundo lugar, el modelo de medida consiste en una función C (la matriz *H* en el capítulo 3) que va a modificar la predicción del estado $\overline{X_t}$ en la predicción de la medida $\overline{Z_t}$, además de un término de error gaussiano ε_z (llamado error de medida v_t en el capítulo 3).

$$\overline{Z_t} = CX_t + \varepsilon_z$$

En la práctica vamos a considerar que los sensores empleados (la cámara) únicamente pueden medir la posición p y no la velocidad. Por tanto, la expresión queda de la siguiente forma:

$$\overline{Z_t} = [p_t] = [1 \ 0] \begin{bmatrix} p_t \\ v_t \end{bmatrix} + \varepsilon_z$$

En el caso de la medida, la matriz de covarianza (*R* o E_z) tiene un único término: la varianza en la medida, o el error cometido respecto a la posición real. Este término es más complicado de determinar con precisión, por lo que se le dará un valor aproximado

$$v_t = \varepsilon_z = R = E_z = [\sigma_z^2]$$

Finalmente, el estado estimado se obtiene como una función lineal del estado predicho y de la diferencia entre la medida real y la predicción en la medida, multiplicado por un factor de ganancia llamado ganancia de Kalman. Si la predicción del sensor es buena el segundo factor se anula y el estado estimado se corresponde con la predicción del estado; en el caso contrario, existe error en la medida y se corrige la predicción del estado para conseguir un estado estimado más preciso.

$$X_{EST} = \overline{X_t} + K(Z_t - \overline{Z}_t)$$

Las anteriores ecuaciones se refieren a un filtro en un entorno de una única dimensión. Para el caso de dos dimensiones (la posición expresada con las coordenadas x e y), las ecuaciones se expresan de la manera siguiente:

$$\overline{X_t} = AX_{t-1} + BU_t \rightarrow \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T^2/2 \\ T \\ T \end{bmatrix} a + \varepsilon_{\mathrm{x}}$$

$$\bar{Z}_t = C\overline{X_t} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \overline{X_t} + \varepsilon_{\mathrm{z}}$$

$$\mathrm{R} = E_z = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, \qquad \mathrm{Q} = E_x = \begin{bmatrix} \frac{T^4}{4} & 0 & \frac{T^3}{2} & 0 \\ 0 & \frac{T^4}{4} & 0 & \frac{T^3}{2} \\ \frac{T^3}{2} & 0 & T & 0 \\ 0 & \frac{T^3}{2} & 0 & T \end{bmatrix}$$

Con las variables que hemos definido, las ecuaciones del filtro de Kalman que vamos a implementar en Matlab son las siguientes:

Predicción

- Predicción (a priori) de la estimación del estado: $\overline{X_t} = AX_{t-1} + BU_t$
- Predicción (a priori) de la covarianza estimada: $P = AP_{t-1}A^T + E_x$

Actualización

- Ganancia de Kalman óptima: $K_t = \mathrm{P}_t C^T (C\mathrm{P}_t C^T + E_z)^{-1}$
- Estimación de estado (a post.) actualizado: $X_t = \overline{X_t} + K_t(Z_t - C\overline{X_t}) = \overline{X_t} + K_t(Z_t - \overline{Z_t})$
- Covarianza estimada (a posteriori) actualizada: $\mathrm{P}_t = (I - K_t C)\mathrm{P}_t$

5.5.2 Filtro Alfa-Beta

Para implementar el filtro alfa beta en Matlab se utilizan las ecuaciones que se exponen a continuación, y que se derivan de lo visto en el capítulo 3:

(1) $\hat{\mathrm{x}}_k \leftarrow \hat{\mathrm{x}}_{k-1} + \Delta T \hat{\mathrm{v}}_{k-1}$
(2) $\hat{\mathrm{v}}_k \leftarrow \hat{\mathrm{v}}_{k-1}$
(3) $\hat{\mathrm{r}}_k \leftarrow \mathrm{x}_k - \hat{\mathrm{x}}_k$
(4) $\hat{\mathrm{x}}_k \leftarrow \hat{\mathrm{x}}_k + \alpha \hat{\mathrm{r}}_k$
(5) $\hat{\mathrm{v}}_k \leftarrow \hat{\mathrm{v}}_k + [\Delta T/\beta] \hat{\mathrm{r}}_k$

El algoritmo utilizado se puede resumir en los siguientes puntos:

Inicialización

- Establecer los valores iniciales de *x* y *v*, usando información previa o medidas adicionales; en otro caso, establecer los valores de estados inicial a cero.

- Seleccionar los valores de la ganancia de corrección *alfa* y *beta* (*).

Actualización. Repetir para cada espacio de tiempo ΔT:

- Predecir las estimaciones del estado *x* y *v* usando las ecuaciones 1 y 2.
- Obtener una medida actual del valor de salida.
- Calcular el residuo *r* usando la ecuación 3.
- Corregir las estimaciones de los estados utilizando las ecuaciones 4 y 5.

(*)Los valores de *alfa* y *beta* se ajustan normalmente de forma experimental. En general, valores elevados de *alfa* y *beta* tienden a producir respuestas más rápidas para el seguimiento de cambios transitorios, mientras que para valores de *alfa* y *beta* pequeños se reduce el nivel de ruido en las estimaciones. Si se alcanza un buen balance entre seguimiento preciso y reducción de ruido, y el algoritmo es eficiente, las estimaciones filtradas son más precisas que las medidas directas.

5.5.3 Anotación sobre diferentes modelos de estado

El modelo de estado que hemos considerado en nuestra aplicación para desarrollar los filtros supone que se dispone de la información de la medida de posición del blanco; además, se establece como modelo de movimiento un movimiento MRUA. Sin embargo, esto no tiene que ser siempre así, ya que existen multitud de modelos que se pueden aplicar en función de la aplicación concreta, y que se adaptan mejor a la situación real que pretenden modelar. De esta manera, y a modo de ejemplo, en [15] se proponen diferentes modelos de estado, para varios tipos de radares y tipos de blancos:

1) Radar Biestático: en un radar de este tipo la información que se recibe consiste en el ángulo θ, que indica los grados de desviación respecto a la referencia donde se encuentra el blanco detectado.

Vector de estado: $\mathbf{x} = [\mathrm{x}\ \mathrm{x}'\mathrm{y}\ \mathrm{y}']^{\mathrm{T}}$,donde (x,y) indican la posición y (x',y') indican la velocidad.

Ecuación de estado:$\mathrm{x}_{\mathrm{k}+1} = \mathrm{F} \cdot \mathrm{x}_{\mathrm{k}} + \mathrm{v}_{\mathrm{k}}$

Donde $\mathrm{F} = \begin{bmatrix} 1 & \mathrm{T} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{T} \\ 0 & 0 & 0 & 1 \end{bmatrix}$, T es el periodo de muestreo y δ_{k}es ruido de media nula, Gaussiano cuya matriz de covarianza es $Q_k = \begin{bmatrix} \sum k & 0_2 \\ 0_2 & \sum k \end{bmatrix}$, con $\sum k = \begin{bmatrix} \mathrm{T}^3/3 & \mathrm{T}^2/2 \\ \mathrm{T}^2/2 & \mathrm{T} \end{bmatrix}$

Las medidas obtenidas son: ángulo θ y frecuencia Doppler δ.

Ecuación de medida: $\mathrm{z}_{\mathrm{k}} = \mathrm{h}(\mathrm{x}_{\mathrm{k}}) + \mathrm{w}_{\mathrm{k}}$

Donde $Q_k = \begin{bmatrix} h_\theta(x_k) \\ h_\delta(x_k) \end{bmatrix}$ con $h_\theta(x_k) = \arctan(\frac{x_k}{y_k})$, $h_\delta(x_k) = \delta_k$y w_k ruido con matriz de covarianza $R = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_\delta^2 \end{bmatrix}$.

2) Objeto Balístico: este modelo hace referencia a un objeto que cae desde una altura únicamente como efecto de la gravedad.

Vector de estado **x** = [h v β], donde β es el coeficiente balístico, que es una incógnita y depende de la masa, forma y área del objeto concreto.

Ecuación de estado: $x_{k+1} = x_k + \tau g(x_k) = f(x_k) + v_k$

$f(x_k) = \Phi x_k - G[D(x_k) - g]$, $\Phi = \begin{bmatrix} 1 & -\tau & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$,$G = [0\ \tau\ 0]^T$,$D(x_k) = \frac{g\rho(x_k[1])\cdot {x_k}^2[2]}{2x_k[3]}$, donde ρes la densidad del aire expresado como $\rho = \gamma e^{-\eta h}$ y g es la gravedad.

v_k es el ruido del proceso, que incluye fuerzas no consideradas sobre el objeto, como variaciones en β… ,con una matriz de covarianza del ruido

$Q = \begin{bmatrix} q_1 {\tau^3}/{3} & q_1 {\tau^2}/{2} & 0 \\ q_1 {\tau^2}/{2} & q_1\tau & 0 \\ 0 & 0 & q_2\tau \end{bmatrix}$, con constantes q1 y q2 coeficientes sobre el movimiento del objeto y sobre β, respectivamente.

Ecuación de medida: $z_k = Hx_k + w_k$, con H = [1 0 0] y w_k ruido con varianza R = σ_r^2

$$P_{0|0} = \begin{bmatrix} R/T & R/T & 0 \\ R/T & 2R/T^2 & 0 \\ 0 & 0 & \sigma_\beta^2 \end{bmatrix}$$

CAPÍTULO 6

ANÁLISIS DEL SISTEMA

En este capítulo se recogen las diferentes pruebas realizadas durante el desarrollo del proyecto, a través de sus distintas fases. Las primeras pruebas consisten en comparar la eficacia de los dos filtros que vamos a emplear en nuestra aplicación: el filtro de Kalman y el filtro alfa beta. Para ello, nos vamos a servir de unas simulaciones simples hechas en Matlab de forma que podamos apreciar con facilidad el funcionamiento de los filtros en unos entornos que sean lo más simplificado posible. Se genera una trayectoria que simula el movimiento real de un blanco ficticio que los filtros deben seguir; sobre esta trayectoria se añade ruido aleatorio correspondiente a la información captada por los sensores ruidosos, y que es la que se traslada a los filtros para funcionar. Después de las simulaciones se pasará a trabajar con secuencias de vídeos reales, donde se evaluarán por separado el funcionamiento de los algoritmo de detección y seguimiento, hasta llegar al sistema final.

6.1 Análisis de los parámetros de los filtros

Antes de comparar los dos filtros entre sí, es preciso conocer el funcionamiento propio de cada sistema. En primer lugar vamos a proceder a estudiar el comportamiento del filtro alfa beta en función de sus dos parámetros. Se realizan dos pruebas para la evaluación, las dos basadas en el seguimiento de un punto que se mueve según una trayectoria con un cambio brusco en la dirección del movimiento:

1) La primera supone que se reciben en todo momento los datos de la medición.
2) La segunda supone la pérdida de los datos medidos en el momento del cambio de dirección en la trayectoria del movimiento, simulando el efecto de oclusión.

Con el primer experimento se quiere estudiar la capacidad de respuesta inmediata del filtro, y con el último experimento se pretende comprobar la capacidad del sistema para recuperar la posición después de un tiempo sin información. La función de cada parámetro se explica brevemente a continuación, y se comprobará específicamente en las simulaciones posteriores.

- El parámetro *alfa* afecta a la capacidad de filtrado. Al aumentar el valor de *alfa* el sistema funciona peor como filtro porque sigue con precisión la señal medida, en lugar de estimar una trayectoria más uniforme.
- El parámetro *beta* afecta a la velocidad del sistema y a la capacidad de seguir al movimiento. Cuanto más pequeño es *beta* más lento es el seguimiento respecto a la posición medida, es decir, tarda más en alcanzar la posición actual.

Para el análisis se utiliza un *alfa* desde 0.1 hasta 0.6, y el parámetro *beta* que corresponde se elige según las ecuaciones:

$$\beta_{opt} = 2(2-\alpha)4\sqrt{1-\alpha} \qquad \beta_{opt} = 0.8\left(\frac{2-\alpha^2-2\sqrt{1-\alpha^2}}{\alpha^2}\right)$$

El propósito de repetir el análisis utilizando dos valores es comprobar cómo afecta la elección de una u otra fórmula para elegir *beta* al correcto seguimiento del punto. Las simulaciones correspondientes al filtro alfa beta se encuentran en las figuras 6.1 y 6.2. El conjunto de simulaciones realizadas en este apartado sobre el análisis de los parámetros de los filtros se encuentran en el anexo B, mientras que a continuación se muestran unos extractos de los mismos para reflejar a modo de ejemplo algunos de los resultados obtenidos.

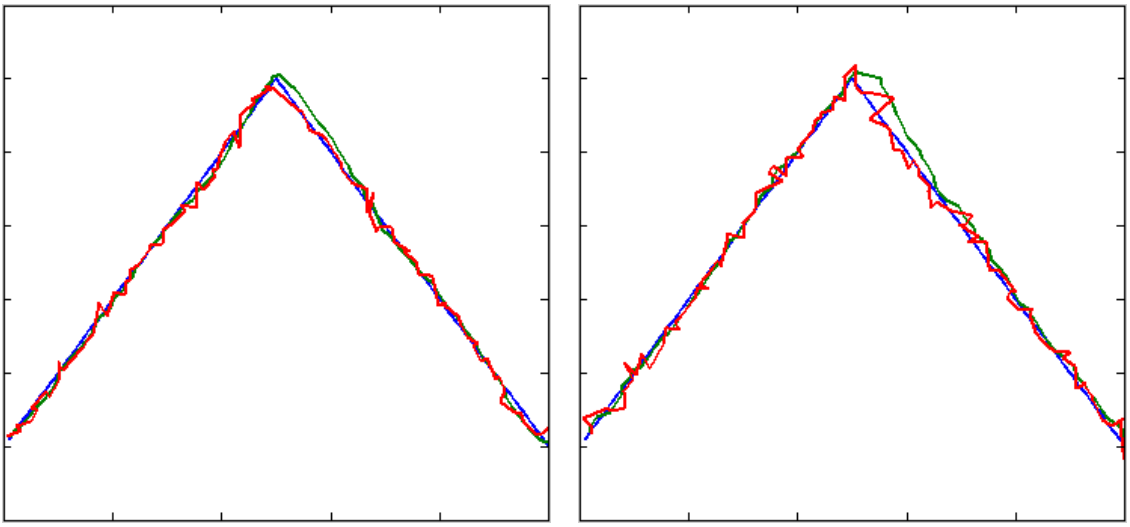
Analizando los resultados obtenidos, de manera general se concluye que a partir de un valor de *alfa*=0.3 o 0.4 se puede obtener un buen compromiso entre filtrado y seguimiento, generando resultados correctos. Si se utilizan valores inferiores para el par *alfa-beta* se producen desviaciones amplias en la trayectoria, ya que la velocidad de respuesta del filtro en estos casos es muy reducida y tarda mucho tiempo en recuperar la posición correcta. Comparando las dos fórmulas de *beta* utilizadas, la segunda de ellas produce un valor mucho menor y, de acuerdo a lo mencionado anteriormente, la velocidad del filtro con esta configuración (y la capacidad de recuperar la posición después de un giro) es menor que la velocidad si se utiliza la primera fórmula. En consecuencia, $\beta_{opt} = 2(2-\alpha)4\sqrt{1-\alpha}$ será la expresión que utilizaremos a partir de este momento.

En cuanto a las simulaciones con oclusión, cuando el filtro deja de recibir los datos sigue funcionando con la misma dirección y velocidad que tenía antes. Por eso vemos que en algunos casos la posición filtrada supera el punto de giro y sigue en esa misma dirección hasta que vuelve a recibir los datos de posición y puede reengancharse a la nueva situación. Igual que sucedía en los casos sin oclusión, dependiendo de los valores de los parámetros el filtro tardará más o menos en recuperar la posición: desde describir una curva que se aproxima lentamente hasta realizar un movimiento rápido al lugar donde reaparece la posición.

Respecto al filtro de Kalman, dado que no dispone de parámetros manualmente modificables (la ganancia de Kalman se ajusta dinámicamente durante la ejecución del filtro), vamos a repetir las mismas pruebas únicamente modificando el parámetro *dt* para analizar cómo afecta cambiar el intervalo de tiempo de actualización. Los resultados de las simulaciones para el filtro de Kalman se presentan en las figuras 6.3 y 6.4.

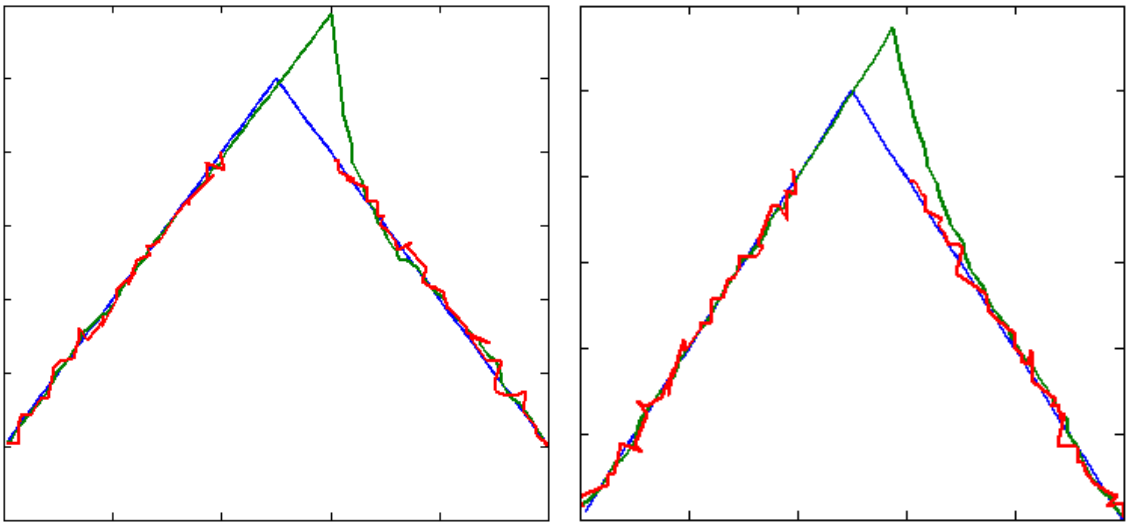
Haciendo un análisis similar al del filtro alfa beta, se observa que para valores bajos, la velocidad se ve drásticamente reducida y como tal no resulta práctico para una aplicación de seguimiento en tiempo real. Esto se debe a que un intervalo de tiempo corto no permite la correcta actualización de la posición del filtro. Es especialmente significativo el caso *dt*=0.1 si se compara la distancia (y el espacio de tiempo que ello supone) entre los puntos en que se produce el giro y dónde comienza a cambiar el filtro, además de que al finalizar la simulación no ha podido recuperar la posición. Alrededor de valores de *dt*=1 los resultados son positivos.

En todas la representaciones realizadas, la línea azul se corresponde con la trayectoria real sin ruido, la línea roja es la trayectoria medida con ruido (en ocasiones desaparece debido a la oclusión) y la línea verde se corresponde con la trayectoria producida por el filtro correspondiente. Debajo de cada gráfica se indica el valor de los parámetros empleado.



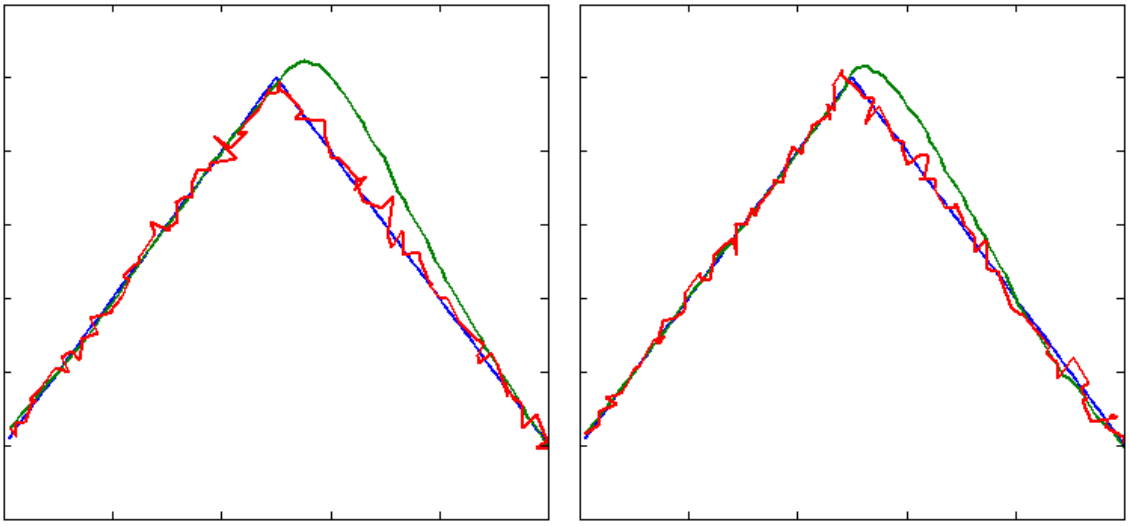
alfa= 0.3 beta= 0.0529 *alfa= 0.3 beta= 0.0189*

Figura 6.1. Filtro alfa beta (sin oclusión)



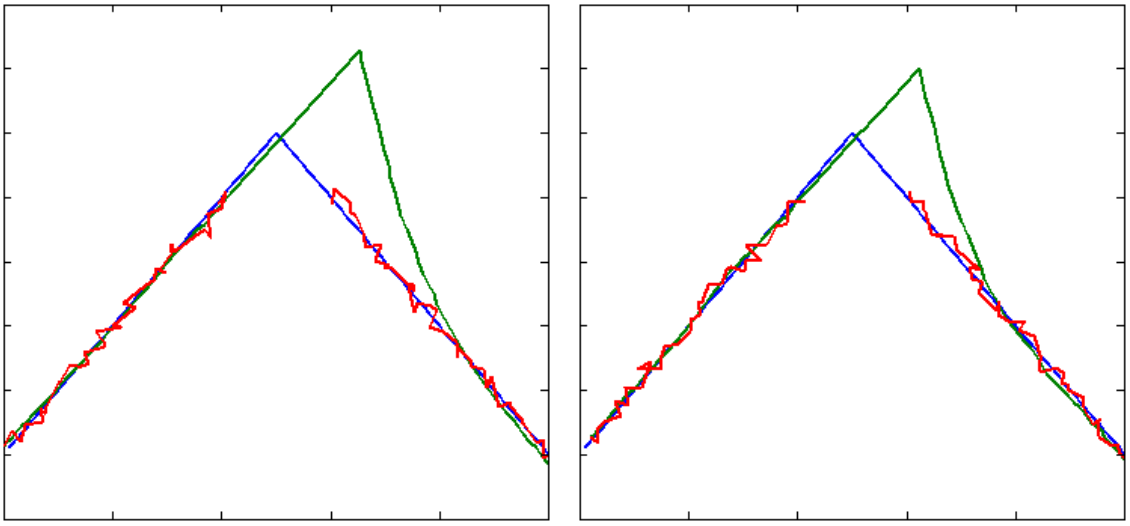
alfa= 0.3 beta= 0.0529 *alfa= 0.3 beta= 0.0189*

Figura 6.2. Filtro alfa beta (con oclusión)



dt=0.3 *dt=0.4*

Figura 6.3. Filtro de Kalman (son oclusión)



dt=0.3 *dt=0.4*

Figura 6.4. Filtro de Kalman (con oclusión)

6.2 Análisis comparativo de los filtros

El siguiente paso del análisis consiste en comparar los dos filtros simultáneamente. Para ello se crean nuevas trayectorias en Matlab que simulen movimientos más complejos, que los filtros deberán seguir. Las características más importantes de los filtros, y en las que más nos vamos a fijar para tenerlas en cuenta para el sistema final, son las relacionadas con la velocidad de seguimiento, la precisión y la capacidad de predecir la posición en caso de oclusión.

La velocidad del filtro hace referencia a la capacidad de alcanzar y seguir la posición actual con la mayor rapidez posible. Veremos que, especialmente al comenzar el seguimiento, el filtro debe desplazarse de su posición inicial hasta la posición en que se encuentra el movimiento y puede tardar cierto tiempo en lograrlo.

La precisión del filtro está relacionada con la capacidad de filtrar los datos con ruido obtenidos y proporcionar una trayectoria más próxima a la trayectoria original. Normalmente, existe un compromiso entre precisión y velocidad: si un filtro es veloz y alcanza con rapidez a la posición medida su capacidad de filtrar los datos y aproximarse a la medida real suele ser reducida, y viceversa.

A continuación vamos a presentar diferentes casos para comprobar el funcionamiento de los filtros, modificando también los parámetros del filtro alfa beta para analizar cómo afectan a su comportamiento, y asumiendo distintos niveles de ruido. Además se representarán en gráficas los siguientes datos, para establecer las comparaciones:

- Distancia entre posición estimada con el filtro de Kalman y la posición real. (Error de estimación)
- Distancia entre posición estimada con el filtro alfa beta y la posición real. (Error de estimación)
- Distancia entre la posición medida y la posición real. (Error de medida)

Caso 1. Trayectoria quebrada

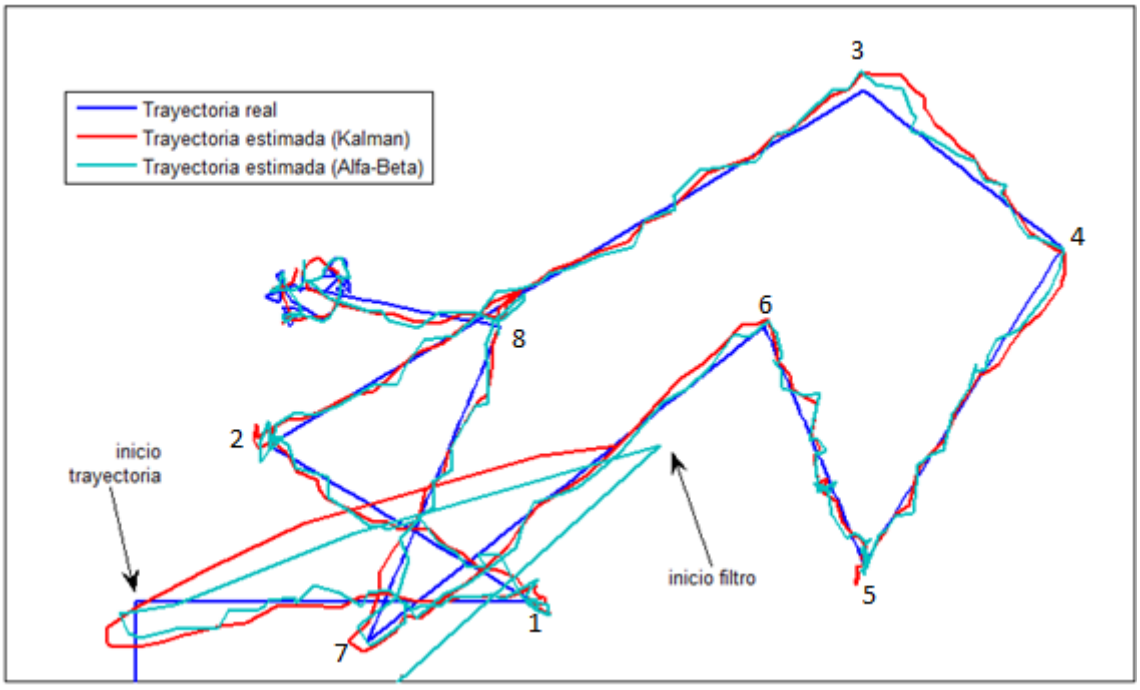


Figura 6.5. Esquema de la trayectoria real diseñada y del seguimiento por los filtros (Caso 1)

En este caso, se muestra la trayectoria rectilínea creada para probar los algoritmos de seguimiento (trayectoria real), que consiste en 8 cambios bruscos de dirección (se numeran en la figura 6.5). No se muestra la trayectoria con ruido sobre la se va a aplicar el seguimiento para no saturar el gráfico. Se puede ver que el punto de inicio de los filtros y el punto de inicio

del movimiento no es el mismo. Por tanto primero los filtros deben ajustarse a la posición, lo que se corresponde con la primera curva que describen y que puede verse en la figura.

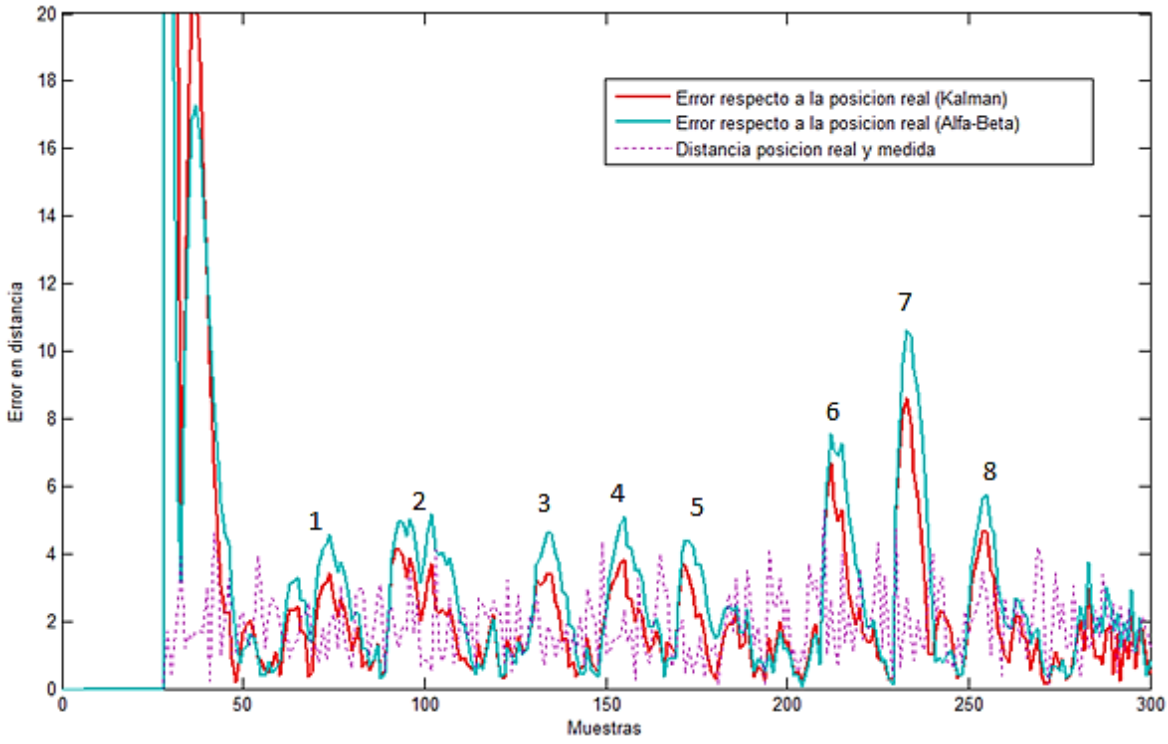


Figura 6.6. Comparativa del error entre la posición real y la estimada por los filtros (1ª simulación)

En la primera simulación se utiliza un nivel de ruido bajo. La configuración de los filtros para este caso concreto es la siguiente: DT=1, alfa=0.3, beta= 0.0534

En la figura 6.6 se puede ver que en los instantes iniciales de la simulación, cuando el filtro debe ajustarse al seguimiento del movimiento (adaptar dirección de movimiento, adaptar velocidad, etc.) el filtro de Kalman presenta más errores que el filtro alfa beta. Sin embargo, tarda menos en alcanzar el estado de seguimiento al comprobarse que su nivel de error se hace menor antes que en el caso del filtro alfa beta. En el resto de la trayectoria, vemos que los dos presentan picos de error en los mismos instantes (producidos esencialmente por cambios bruscos en la trayectoria que deben corregirse, que se encuentran numerados de acuerdo a la figura 6.5), siempre el error del filtro alfa beta es mayor que en el filtro de Kalman, además de que tarda más tiempo en volver a una situación de poco error (tarda más en recuperar la posición).

Atendiendo ahora al nivel de error, el error provocado por los filtros es mayor que el error en la medida en los casos donde se produce algún giro por lo ya comentado anteriormente, pero en las zonas de transición es notablemente menor.

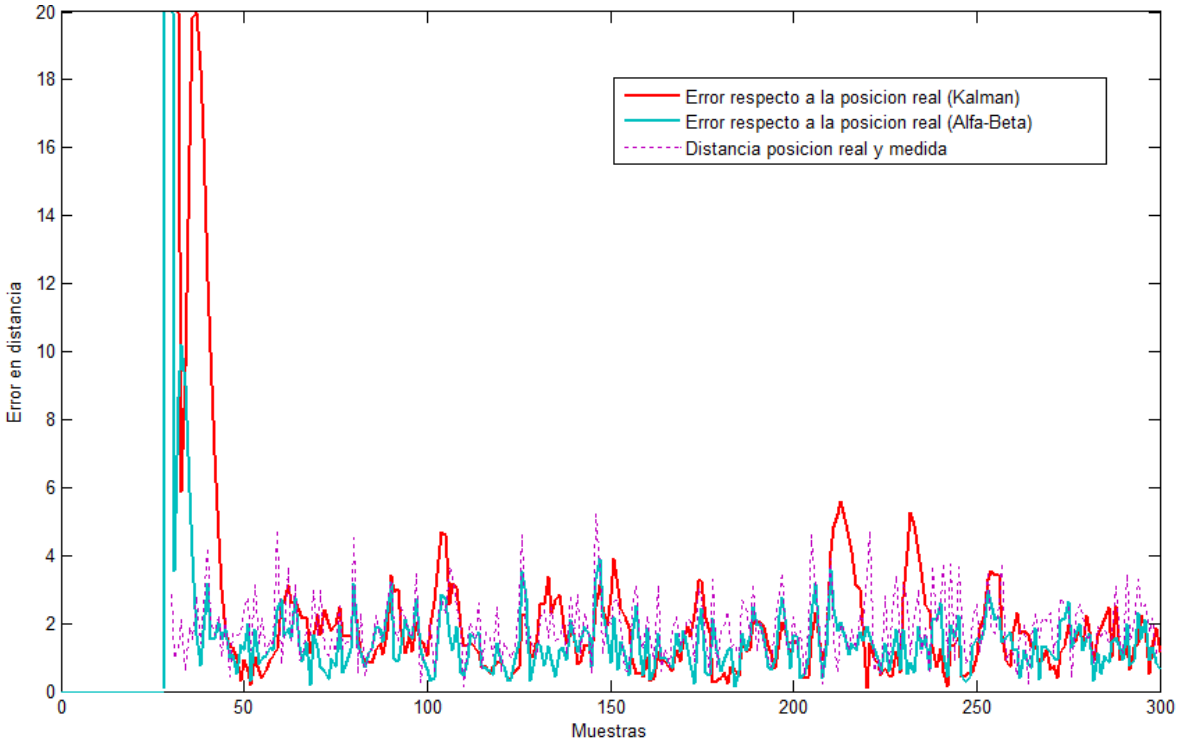


Figura 6.7. Comparativa del error entre la posición real y la estimada por los filtros (2ª simulación)

En esta segunda simulación se mantiene el mismo nivel de ruido bajo y se modifica el parámetro alfa. Los parámetros elegidos para esta simulación son: DT=1, alfa=0.6 y beta=0.2702. El resultado se ilustra en la figura 6.7.

Al aumentar alfa se consigue que el filtro alfa beta tenga mayor rapidez, pero también se hace más sensible al ruido. Esto explica que en la inicialización este filtro alcance antes un nivel bajo de error que el filtro Kalman. En los puntos en que el movimiento cambia de dirección, el filtro de Kalman necesita más tiempo para recuperar la posición y su error aumenta, mientras que el error en el filtro alfa beta es mucho más reducido. Por el contrario, en los tramos rectos el error del filtro alfa beta es mayor por ser más sensible al ruido, mientras que el filtro de Kalman produce un seguimiento más suave y con menos error.

Para las siguientes simulaciones aumentamos el nivel de ruido y se evalúan de nuevo los parámetros *alfa*=0.3 y *alfa*=0.6, respectivamente. Al tener un mayor nivel de ruido, lo que se traduce en mayor error entre posición real y posición medida, no varía significativamente el funcionamiento de los filtros. Para *alfa*=0.3 (figura 6.8) el error del filtro alfa beta es mayor que en el filtro Kalman en las zonas más problemáticas del recorrido y similar en el resto. Con *alfa*=0.6 (figura 6.9) el filtro se comporta de manera más aleatoria, siguiendo las fluctuaciones del ruido introducido.

El nivel de error obtenido es, lógicamente, mayor que en las primeras simulaciones, pero se observa que con un nivel de ruido alto el error de los filtros es inferior en toda la simulación y es sustancialmente menor que la diferencia entre la posición real y la posición medida.

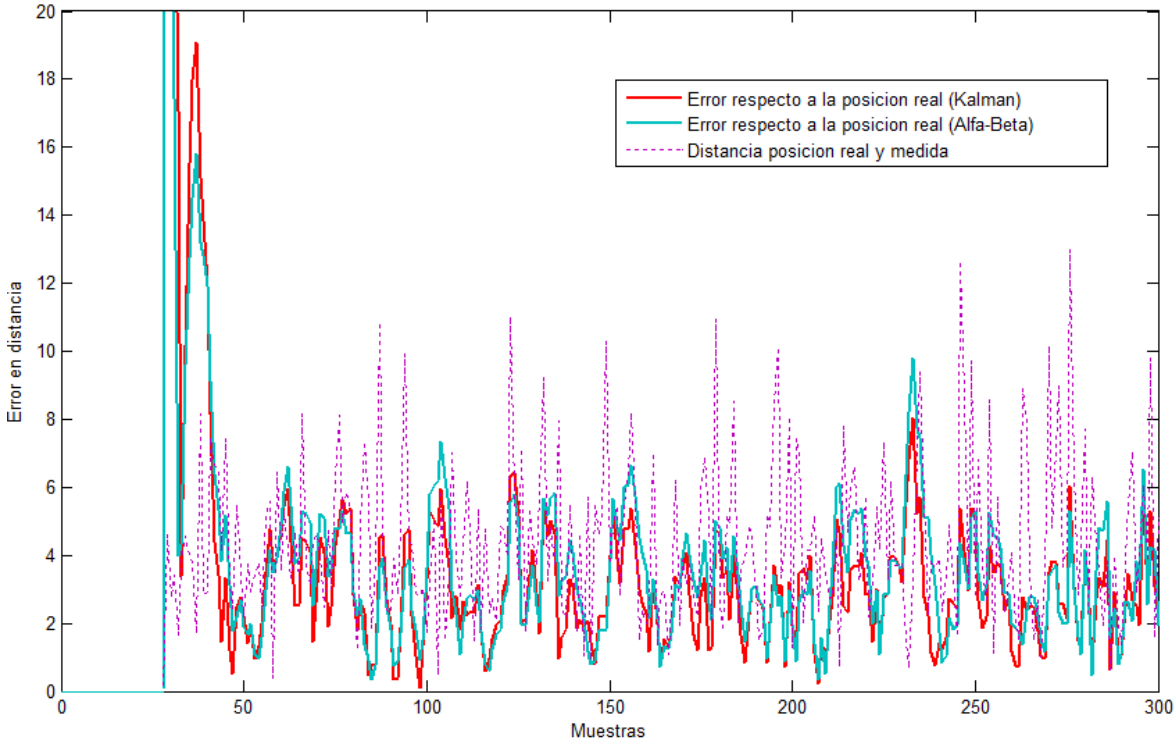


Figura 6.8. Comparativa del error entre la posición real y la estimada por los filtros (3ª simulación)

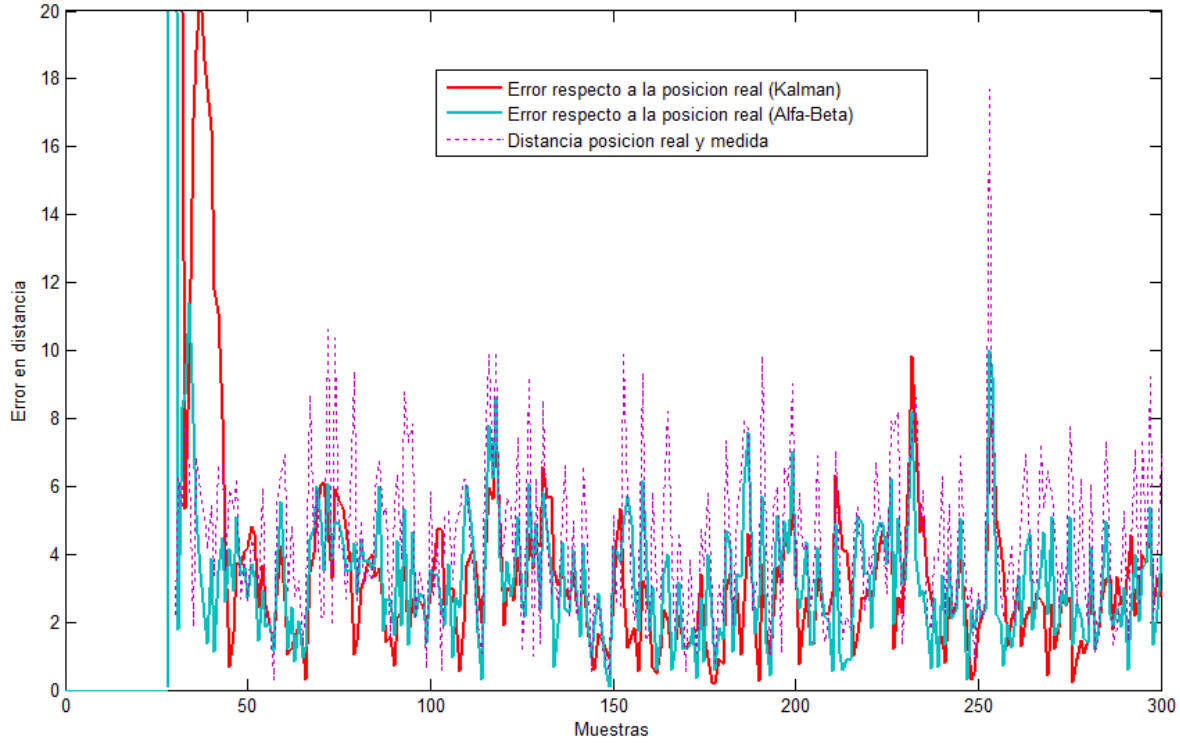


Figura 6.9. Comparativa del error entre la posición real y la estimada por los filtros (4ª simulación)

Caso 2. Trayectoria curvilínea

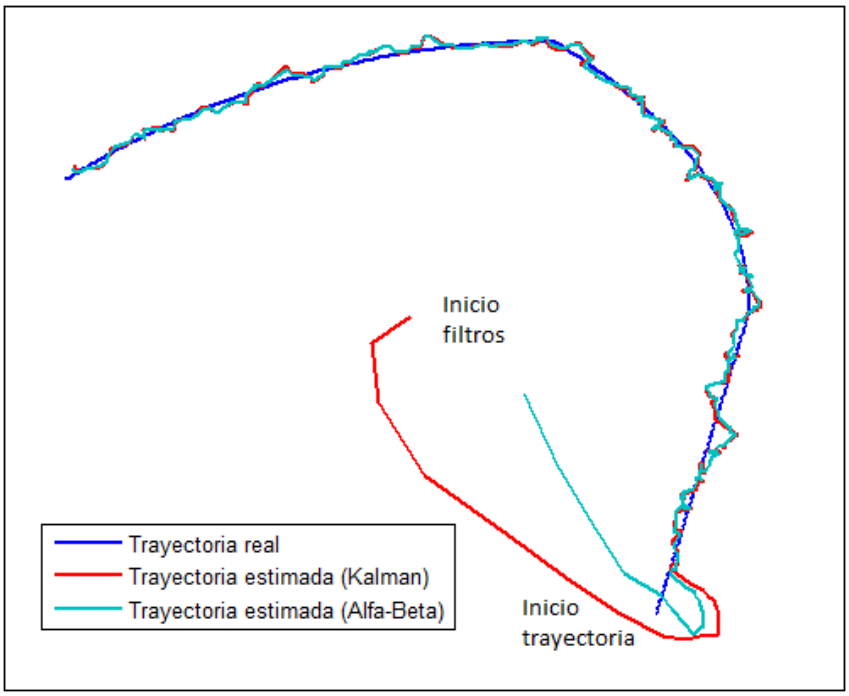


Figura 6.10. Esquema de la trayectoria real diseñada y del seguimiento por los filtros (Caso 2)

El segundo ejemplo que se quiere simular consiste en un camino más sencillo que el anterior con menos cambios de dirección, formado por tres tramos, dos de ellos curvilíneos. El objetivo de realizar este tipo de movimiento es evaluar cómo responden los filtros ante un movimiento curvo, frente al movimiento plenamente rectilíneo del caso anterior.

Las siguientes figuras recogen las cuatro situaciones posibles, que repiten las realizadas en el caso anterior: ruido bajo y alfa=0.3 (figura 6.11), ruido bajo y alfa=0.6 (figura 6.12), ruido alto y alfa=0.3 (figura 6.13), ruido alto y alfa=0.6 (figura 6.14). El filtro de Kalman no puede modificarse, y sólo se establece que *dt* tome el valor 1.

Según muestran las figuras, se comprueba que cuando alfa es 0.3 el comportamiento de los dos filtros es muy similar, presentando mayor error el filtro de Kalman cuando se producía algún pico de error. Cuando alfa es 0.6 los errores del filtro alfa beta son generalmente mayores, tienen un carácter más aleatorio, ya que siguen más los datos medidos. En todos los casos, se comprueba que el nivel de error de los filtros es menor que el error de la medida y, por tanto, los dos sistemas funcionan como filtro correctamente.

Al realizar esta prueba se comprueba que una trayectoria circular como la propuesta no afecta negativamente a los filtros pese a estar modelados como MRUA, y que son los cambios bruscos de trayectoria los causantes principales de los errores.

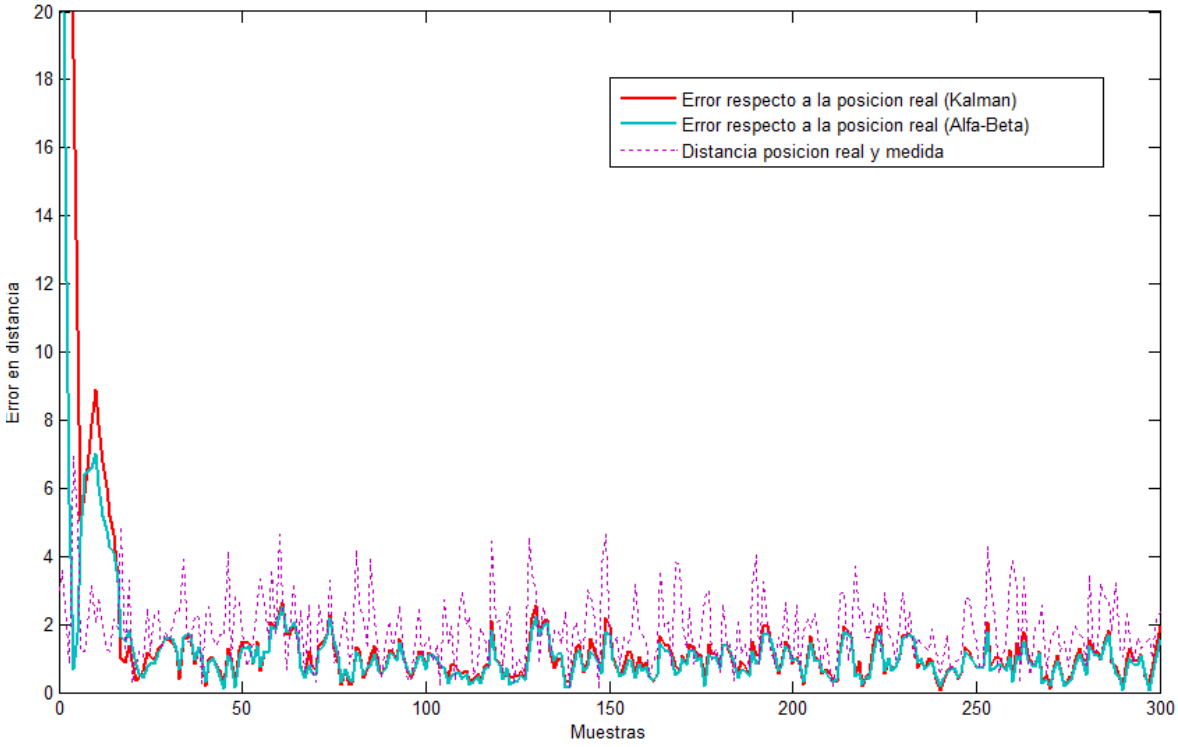


Figura 6.11. Error en el filtrado: Ruido bajo y alfa=0.3

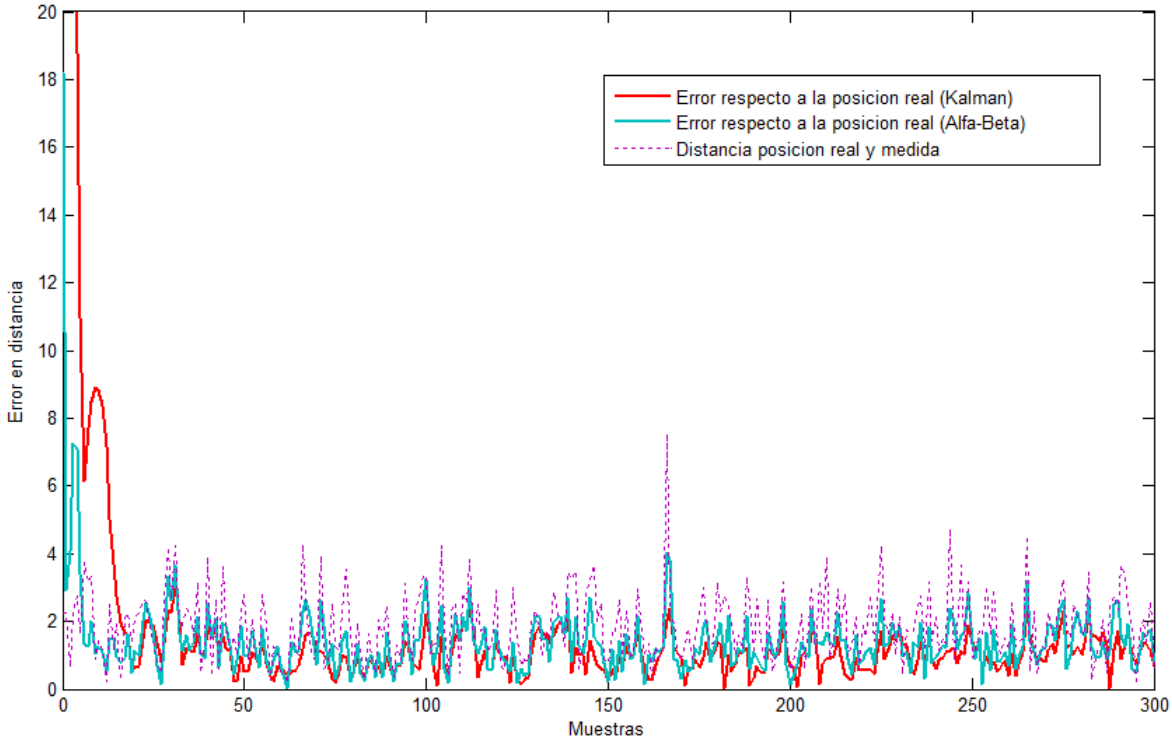


Figura 6.12. Error en el filtrado: Ruido bajo y alfa=0.6

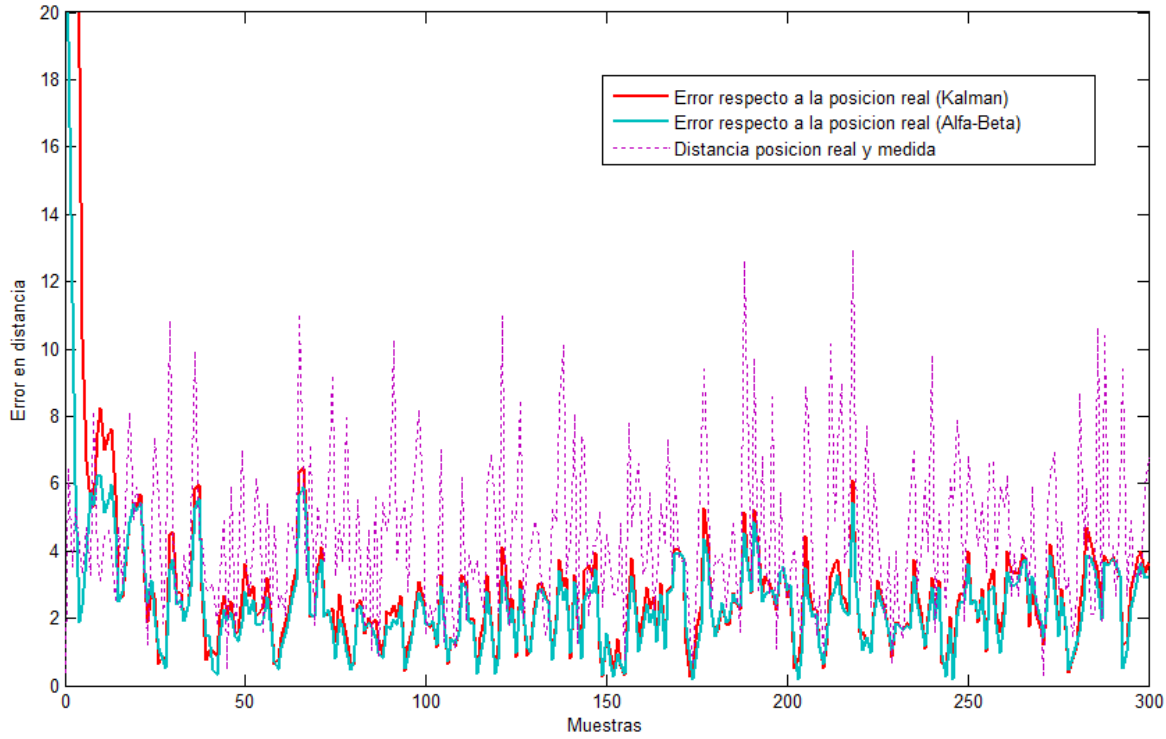


Figura 6.13. Error en el filtrado: Ruido alto y alfa=0.3

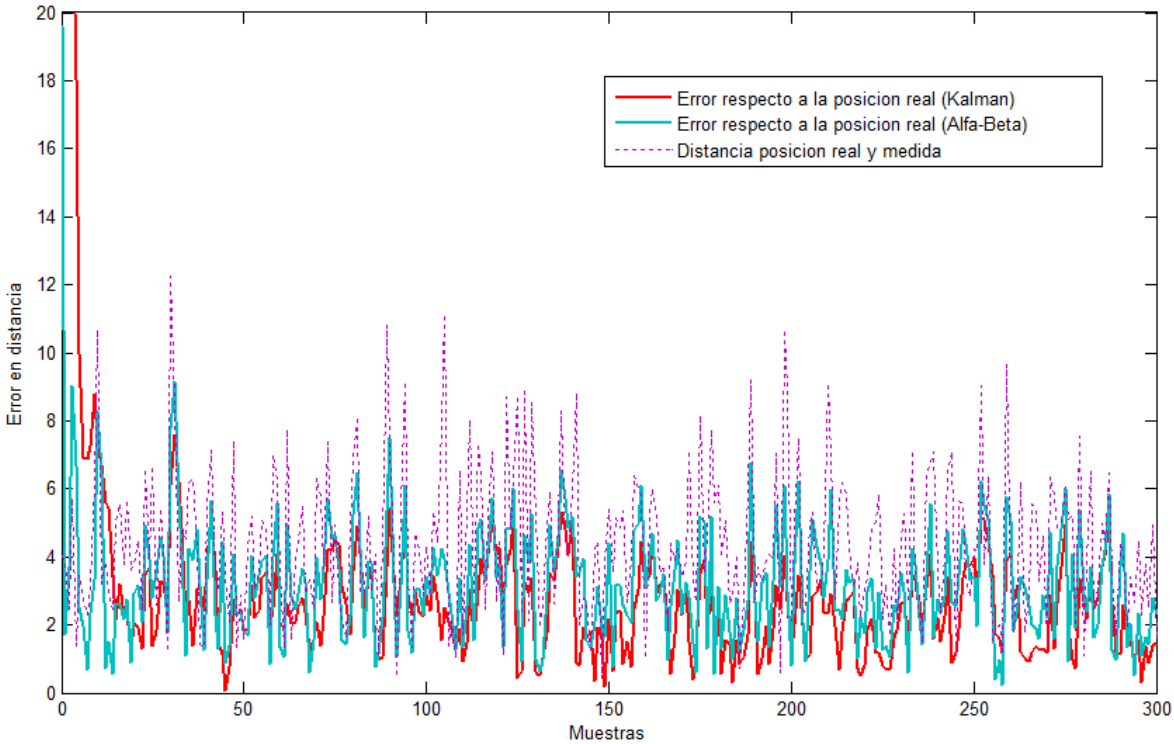


Figura 6.14 Error en el filtrado: Ruido alto y alfa=0.6

6.3 Análisis de funcionamiento del sistema en video

- Detección SSIM y seguimiento

Una vez que se ha experimentado con los filtros en un entorno controlado y se conoce su funcionamiento, es hora de aplicar los algoritmos de seguimiento a videos reales. Las primeras pruebas se realizan empleando el algoritmo de detección a todo el *frame* de video completo, con el fin de comprobar cómo responde el sistema a la adquisición de datos provenientes de un video, a diferencia de las pruebas anteriores donde los datos eran ficticios. Se quiere comprobar cómo el algoritmo de detección SSIM y el de seguimiento funcionan a la vez sobre un mismo video. En esta parte no se considera que la cámara se vaya a mover siguiendo al objetivo, sino que tomando la imagen capturada fija se pretende indicar sobre dicha imagen la trayectoria filtrada.

En este ejemplo se utiliza un video donde un pequeño objeto se mueve libremente de manera aleatoria a alta velocidad. El video esta obtenido de internet en la página [38]. También se incorporan a la vez el filtro de Kalman y el alfa beta para compararlos entre sí.

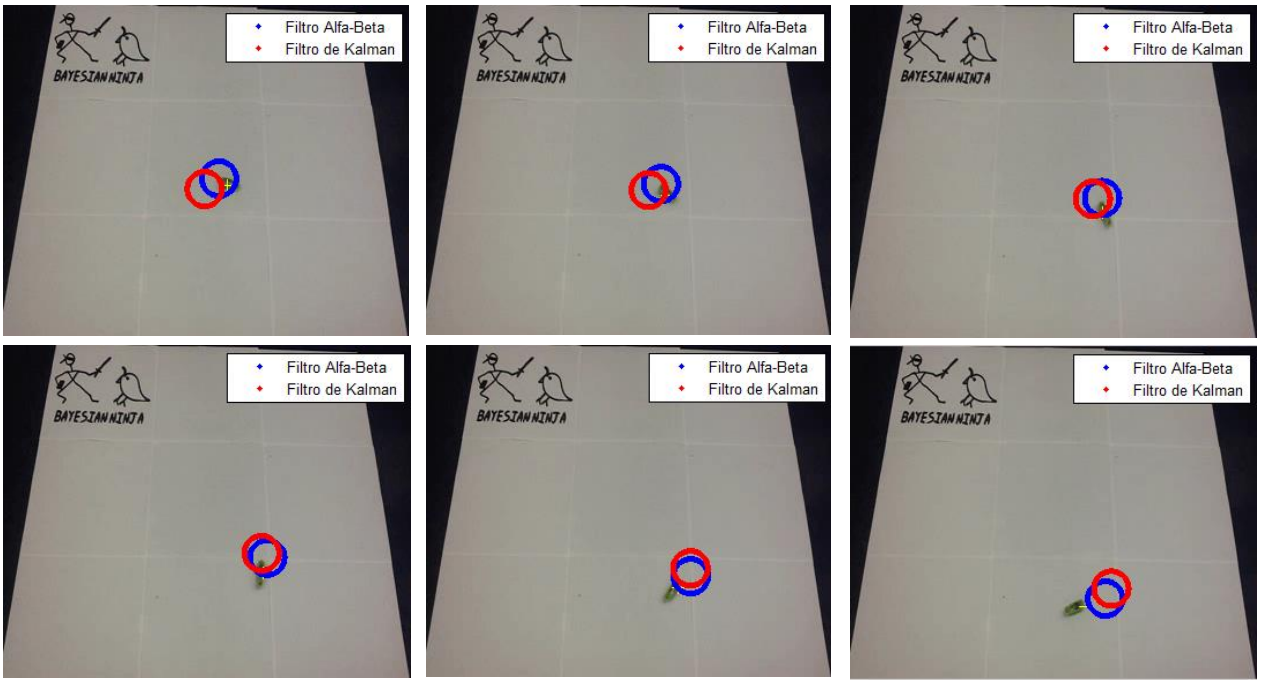


Figura 6.15. Secuencia 1

Las figuras 6.15 y 6.16 muestran una secuencia de fotogramas correspondientes a varios instantes del vídeo, cada *frame* está separado entre si 5 fotogramas. En ambos casos se supone que en la primera imagen los dos filtros ya llevan un intervalo de tiempo funcionando, lo suficiente para haber superado la fase de ajuste inicial. Como se comentaba anteriormente, se aplica el algoritmo SSIM sobre los fotogramas del video y del mapa SSIM resultante se genera un punto en el centro del objeto en movimiento, que se representa como una pequeña

cruz amarilla en las figuras. Este punto es la medida de la posición que se introduce en los filtros.

En referencia al funcionamiento, hay que señalar que el objeto se mueve a una velocidad elevada, lo que supone que cuando este describe una curva prolongada los dos filtros tienen algunos problemas para seguir el movimiento con precisión. Se observa que la posición de los filtros queda en ocasiones por detrás del punto medido. Cuando el movimiento es aproximadamente rectilíneo el seguimiento se ajusta con precisión al blanco, aunque se siga manteniendo la misma velocidad elevada. Hay que tener en cuenta que el algoritmo de detección genera el punto medido en la parte posterior del objeto contraria a su movimiento, especialmente cuando en los dos *frames* que se comparan el desplazamiento de los objetos entre ambos es considerable (por ejemplo, en cambios de dirección), lo que puede introducir errores en el seguimiento.

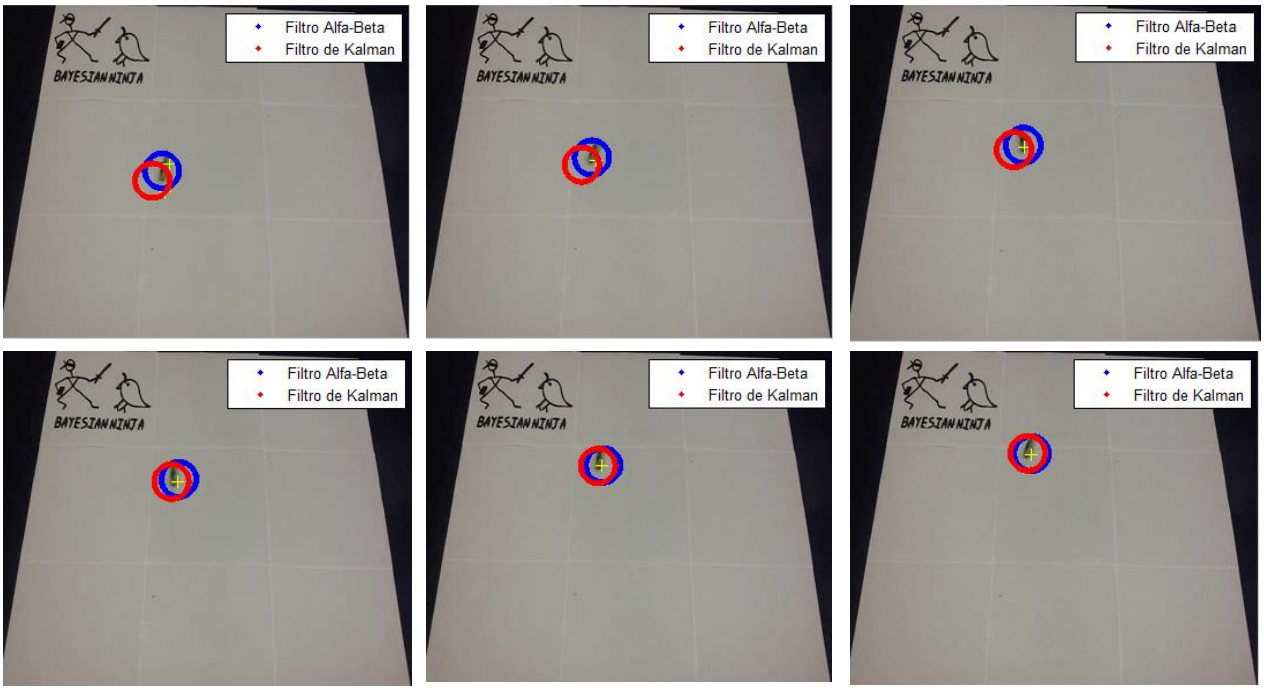


Figura 6.16. Secuencia 2

- **Detección SSIM y movimiento de cámara**

En este nuevo conjunto de pruebas se pretende evaluar el funcionamiento del algoritmo de detección cuando tomamos fragmentos del *frame* de video total y se comparan entre si al desplazarse (simulando el movimiento de la cámara, que se corresponde con el movimiento de una parte de la imagen sobre otra que esta fija). Para ello debe calcularse el desplazamiento, deslizando una imagen sobre otra calculando el índice SSIM en cada iteración hasta encontrar el máximo, que se corresponde con el desplazamiento ocurrido entre ambos *frames*. Como en esta ocasión no utilizamos ningún algoritmo de seguimiento, las imágenes desplazadas donde aparece un objeto moviéndose se obtienen de forma artificial, conociendo de antemano cómo

se va a producir el movimiento en ellas, únicamente con el objetivo concreto que hemos mencionado en este apartado.

En el ejemplo de la figura 6.17, se parte de los *frames* completos y se toma la zona dentro del cuadro verde como las imágenes donde se realiza la detección y se genera el mapa SSIM. Se aplica el mecanismo mencionado anteriormente para calcular el desplazamiento producido entre el *frame* anterior y actual, y poder calcular el mapa SSIM correcto añadiendo ese desplazamiento.

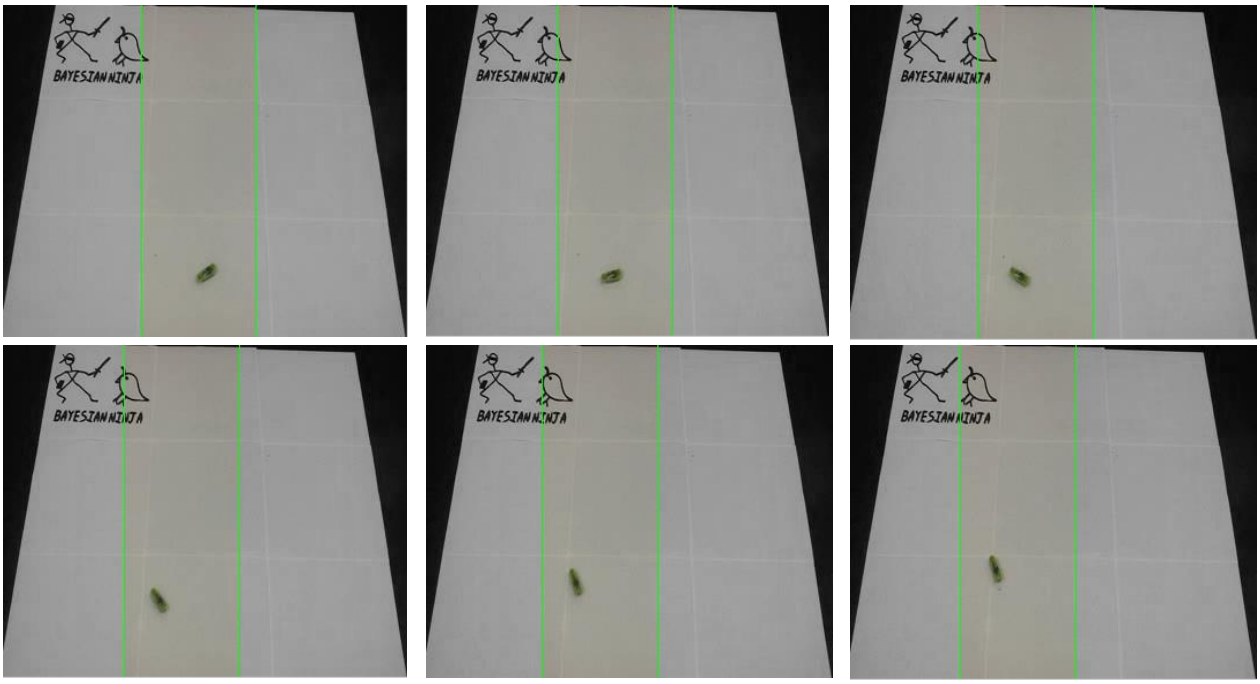


Figura 6.17 Secuencia de detección con movimiento de cámara

En la figura 6.18 se muestran los mapas correspondientes a cada instante de tiempo. Se observa que el algoritmo detecta correctamente el movimiento del objeto y que los elementos del fondo, como las letras y dibujos, no se consideran como movimiento pese a que la porción de los mismos que aparece dentro del recuadro varía entre los distintos *frames*. Es necesario señalar que los *frames* utilizados en el ejemplo (al igual que en el resto de simulaciones realizadas) no son exactamente consecutivos sino que están espaciados 3 fotogramas, lo que provoca una mayor diferencia entre la posición del objeto móvil entre dos *frames* observados y que en el mapa SSIM se observe una especie de “desdoblamiento” de la posición detectada.

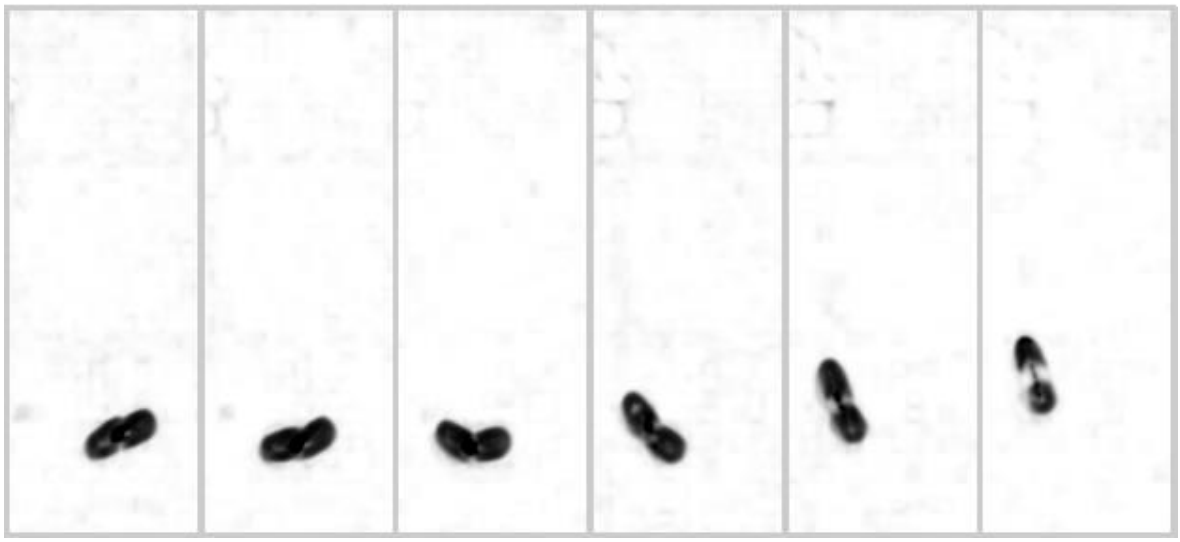


Figura 6.18 Mapas SSIM correspondientes a la secuencia de video

- **Sistema final**

Las últimas pruebas consisten en ejecutar el sistema con todas las funcionalidades completas, que unifica las características probadas en los dos apartados anteriores: detección de movimiento con el algoritmo SSIM en presencia de movimiento de la cámara y aplicación de un algoritmo de seguimiento. Como se explicó en el capítulo de diseño de la aplicación, la detección se realiza dentro de la ventana que se va desplazando por todo el área de vigilancia buscando objetivos en movimiento. En las siguientes figuras se desarrolla un caso de ejemplo en el que se puede comprobar cómo se aplica la detección y seguimiento a una secuencia de vídeo.

En la figura 6.19 se presenta una secuencia donde se detecta y se sigue el movimiento de una moto que se desplaza de izquierda a derecha por el área de vigilancia, y que ha sido detectado por la ventana al realizar un barrido. En la primera detección del objeto móvil, del área generada en el mapa SSIM se obtiene un punto que se pasa al filtro de seguimiento y que se utiliza para obtener la estimación de la posición del blanco. Esta posición estimada se representa como un punto azul en las imágenes, y se utiliza además como centro de la ventana de detección para poder seguir el movimiento del objeto y detectarlo en los siguientes *frames* del video.

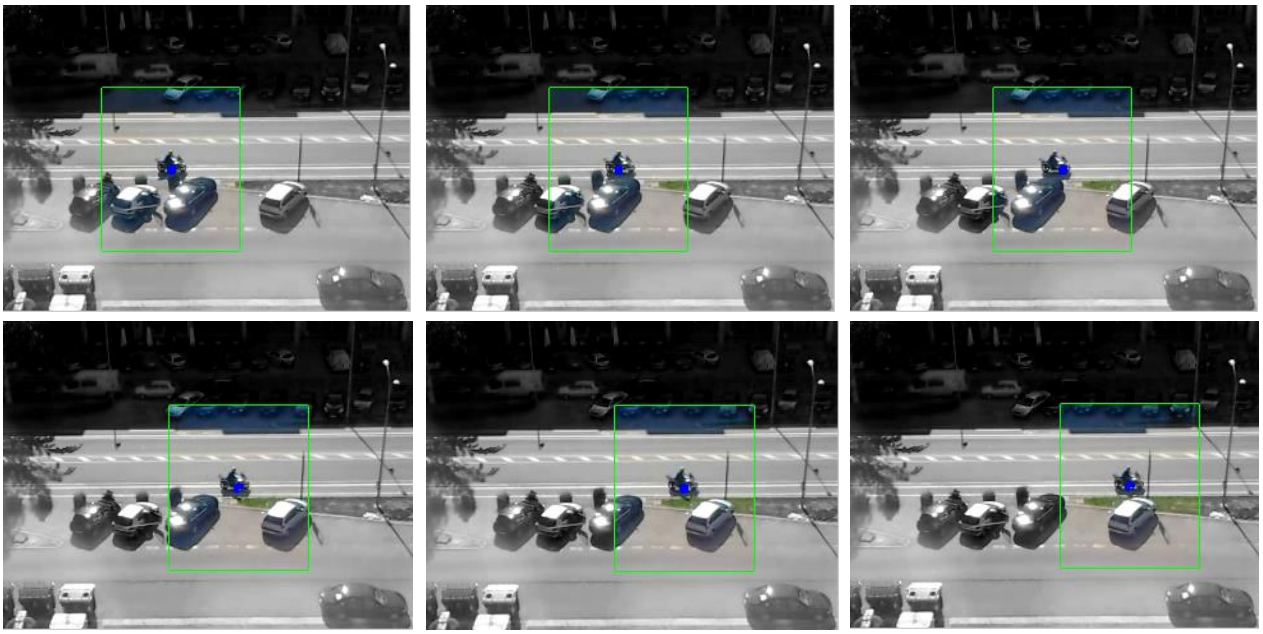


Figura 6.19. Secuencia de seguimiento

Para ilustrar cómo se parte del mapa SSIM de dos fotogramas de vídeo consecutivos comparados, la figura 6.20 muestra la relación entre el mapa SSIM generado en un determinado instante, con la posición estimada indicada por el filtro de seguimiento para el *frame* correspondiente. Al mapa SSIM obtenido se ha aplicado un umbral de 0.5 que lo convierte en una imagen binaria la cual se ha representado sobre el *frame* actual del vídeo. También se muestra como una cruz amarilla el punto que se pasa al filtro como posición medida.

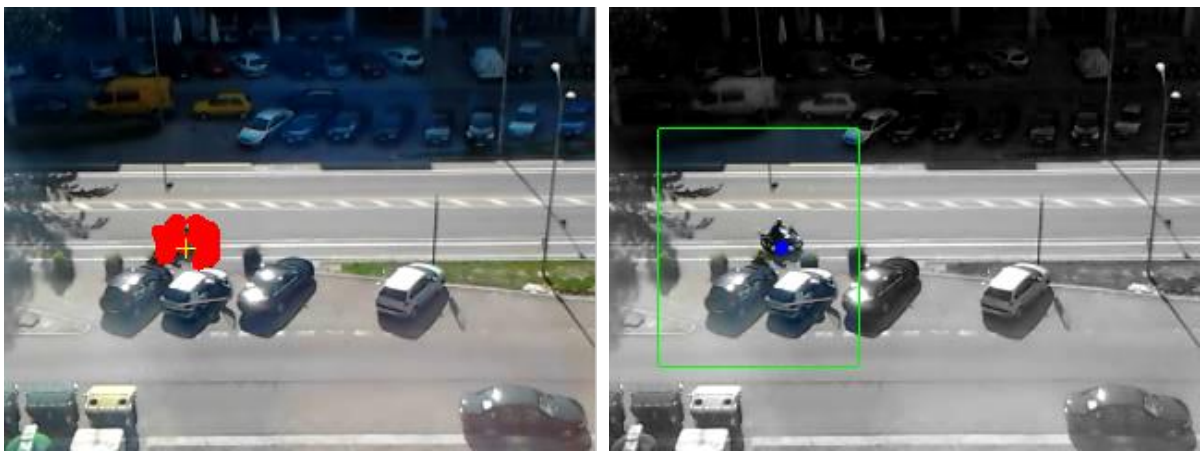


Figura 6.20 Mapa SSIM y posición estimada por el filtro

La ventana puede moverse con libertad por toda el área de vigilancia, pero cuando está siguiendo a un móvil que se desplaza cerca de los bordes de la imagen, la ventana no puede sobrepasar el límite. En casos como estos la ventana no puede centrarse exactamente sobre el

objetivo, sino que debe adoptar la posición más cercana ajustándose a los limites adecuadamente y manteniendo al objeto en movimiento dentro de la ventana. En el ejemplo de la figura 6.21 se observa un coche moviéndose por la parte inferior de la imagen, como la ventana no pude colocarse exactamente encima del blanco, se muestra la posición que tiene el centro de la ventana con un punto azul y la posición real del filtro de seguimiento si este pudiera moverse libremente con un círculo amarillo. En las últimas imágenes de la secuencia se observa cómo el coche sale del área de vigilancia y cuando el círculo amarillo lo sigue ya no está en la vertical del punto azul.

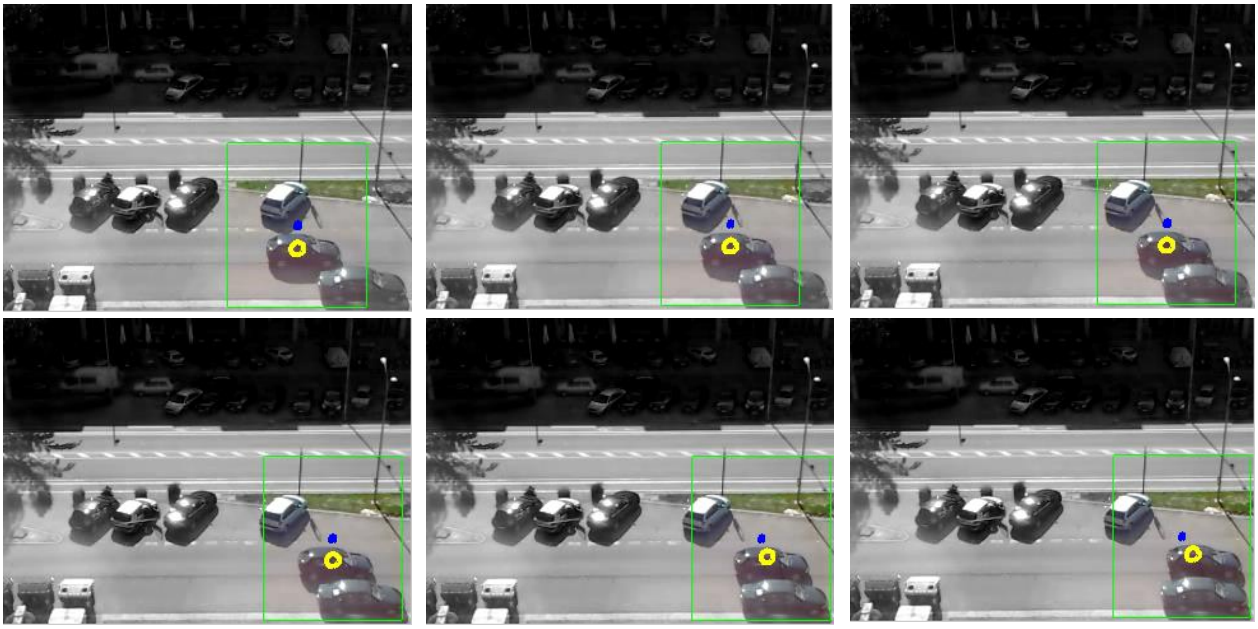


Figura 6.21 Secuencia de ejemplo

CAPÍTULO 7

PRUEBAS Y RESULTADOS

Después de conocer las posibilidades que permite el sistema diseñado, en este capítulo se analizará el funcionamiento del sistema sobre un conjunto de secuencias de video de prueba, que comprende algunos de los casos que se han grabado para testear y evaluar la aplicación.

Buena parte de la efectividad y del correcto funcionamiento de la aplicación viene dado por el algoritmo de detección, cuya importancia es capital dentro del sistema global. Merece la pena repasar de manera aislada el comportamiento del algoritmo SSIM como mecanismo de detección de objetos en movimiento, sus aciertos y sus problemas. En el anexo A se recuperan un conjunto amplio de pruebas, donde se analiza su rendimiento en detalle.

Volviendo a nuestro sistema final, dentro del funcionamiento general como sistema de vigilancia, a continuación se seleccionarán algunos videos que ilustren situaciones concretas de importancia y que requieran alguna explicación adicional. Para organizar el banco de pruebas, las secuencias se clasifican en grupos según su complejidad: Los casos más simples son aquellos en los que un blanco entra por un extremo del área de vigilancia y sale por el otro siguiendo una línea más o menos recta a velocidad constante. Otros casos incluyen más de un elemento móvil simultáneamente (el sistema tendrá que seguir sólo uno), desplazamientos con velocidad variable o con trayectoria algo más compleja.

- **Primer grupo**

En este primer conjunto de secuencias de ejemplo, tenemos unas personas que cruzan de izquierda a derecha (y viceversa) a una velocidad baja. Dependiendo de la posición inicial del filtro tardará más o menos en encontrar el blanco, y el proceso de engancharse será más o menos brusco; pero una vez conseguido el sistema debe funcionar sin mayores problemas.

Video: *viña3.avi*

Se debe ajustar el tamaño del bucle de barrido en la corrección del desplazamiento para que cuando el sistema detecte la presencia de un blanco, y la ventana de detección de mueva rápidamente sobre dicho blanco, este movimiento quede dentro de los límites del barrido. De otra forma, el sistema da error en el instante a continuación que detecta el blanco, pero

funciona correctamente en los siguientes, por lo que el funcionamiento global del sistema no se ve afectado. (nota: deja de detectar cualquier movimiento durante el instante con error). Un barrido de +/- 30 pixeles en horizontal y vertical, suele ser suficiente para todos los casos. Como se indicó en el capítulo 5, un aumento del tamaño del barrido supone una ralentización considerable del tiempo de procesado.

El siguiente grupo de fotogramas (figura 7.1) ilustra el proceso de inicio del seguimiento de un blanco, cuando el bucle de corrección del desplazamiento tiene suficiente tamaño, en este caso [-30 x +30] en las dos direcciones. Se observa que entre dos fotogramas en los que la ventana de desplaza considerablemente el mapa SSIM generado es más pequeño pues solo se corresponde a las zonas con movimiento visualizadas anteriormente.

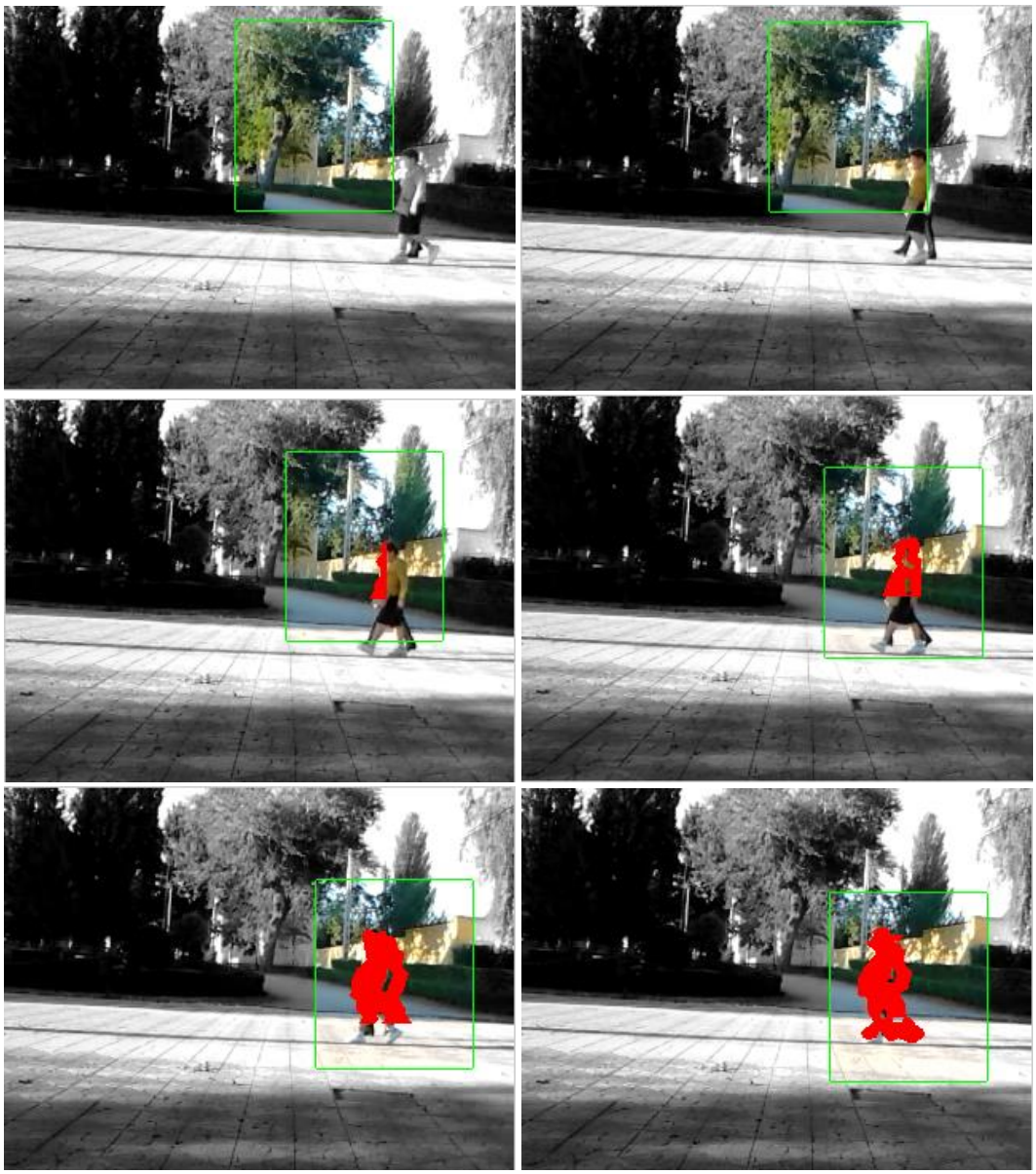


Figura 7.1. Inicio de seguimiento 1

En la siguiente secuencia (figura 7.2) se representa el inicio del seguimiento de un blanco cuando la corrección de posición no funciona adecuadamente (el tamaño del bucle es [-20 x +20]). Se observa que cuando la ventana se desplaza un número de píxeles superior a las dimensiones del bucle se produce un error al generar el mapa SSIM, después el sistema funciona con normalidad pues la ventana ya está totalmente colocada sobre el blanco a seguir.

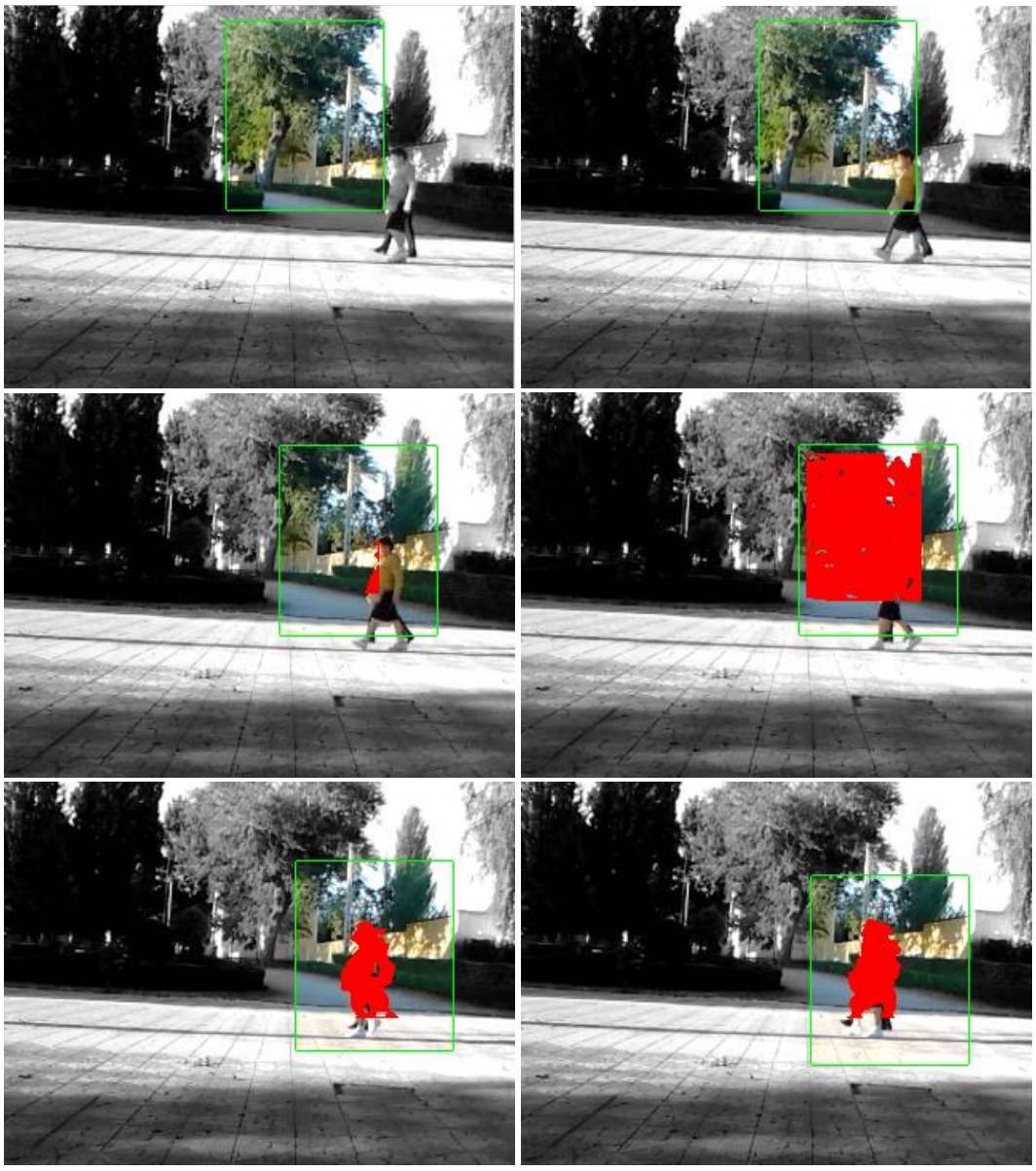


Figura 7.2. Inicio de seguimiento 2

-Segundo grupo

En el segundo grupo de secuencias de ejemplo, tomamos videos que tengan como característica principal la presencia un único blanco que describa una trayectoria compleja: movimientos zigzagueantes, movimientos circulares, cambios de dirección y de velocidad, etc.

Video: *Vid_BugTrack.avi*

En este video, tomado de Internet [38], se coloca un mini-robot (HexBug) sobre una superficie lisa y se le deja moverse libremente por el espacio, consiguiendo una trayectoria aleatoria que se adapta a nuestras necesidades, para evaluar el funcionamiento de nuestro sistema en diferentes situaciones.

Figura 7.3 Imagen del mini-robot y trayectoria seguida en el video

En este caso es especialmente significativo hacer una distinción entre el funcionamiento de las dos opciones de corrección del desplazamiento propuestas. Utilizando la opción en la que se conoce el desplazamiento, el funcionamiento es óptimo (de la misma manera que para el resto de videos usando esta opción), pero con el segundo procedimiento el resultado es muy diferente. El principal problema viene dado por el fondo de toda la secuencia, el cual es muy plano (totalmente de color blanco a excepción de unas letras en la esquina superior) y provoca que a la hora de hacer el bucle de solapamiento de imágenes el índice SSIM se calcule de forma incorrecta al entender que las zonas de máxima coincidencia sean cuando el objeto en movimiento se superpone entre las dos imágenes y no cuando se tienen dos fondos idénticos (ante la imposibilidad de distinguir detalles identificadores que demuestren la existencia de un movimiento entre un instante y otro).

En la figura 7.4 cuando la ventana de detección va salir de la zona de la imagen que tiene un dibujo de fondo, en el instante anterior el dibujo sigue estando dentro de la ventana mientras

que en el instante actual el dibujo queda fuera por la izquierda. Al comparar las dos imágenes, este dibujo hace una diferencia significativa entre las dos imágenes y, al ser un fondo plano sin detalles identificadores, el algoritmo lo detecta como un movimiento, lo que supone un resultado erróneo.

La figura 7.5 muestra un caso en el que se aplica compensación de desplazamiento conociendo el movimiento entre fotogramas. Como puede observarse, el seguimiento del objeto es perfecto al desplazar siempre las imágenes que se comparan la distancia recorrida por la posición del filtro entre dos instante consecutivos.

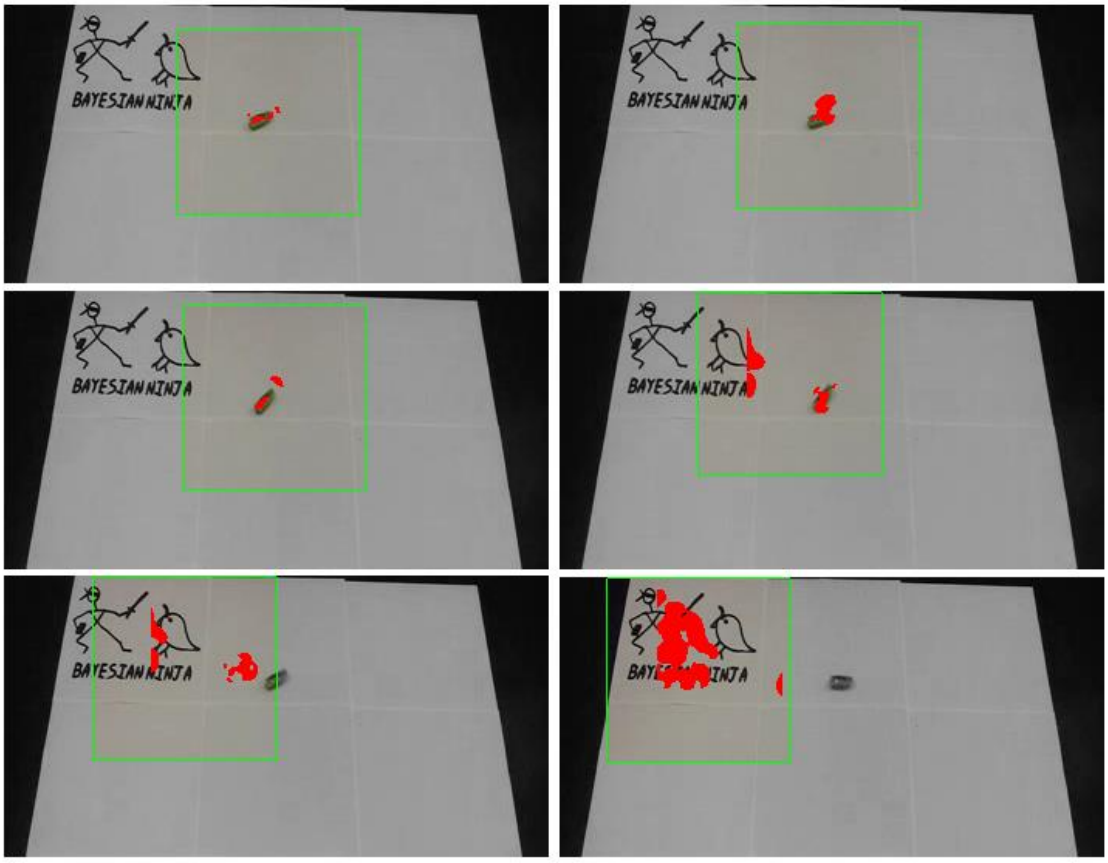


Figura 7.4. Resultado si el desplazamiento entre fotogramas no se conoce

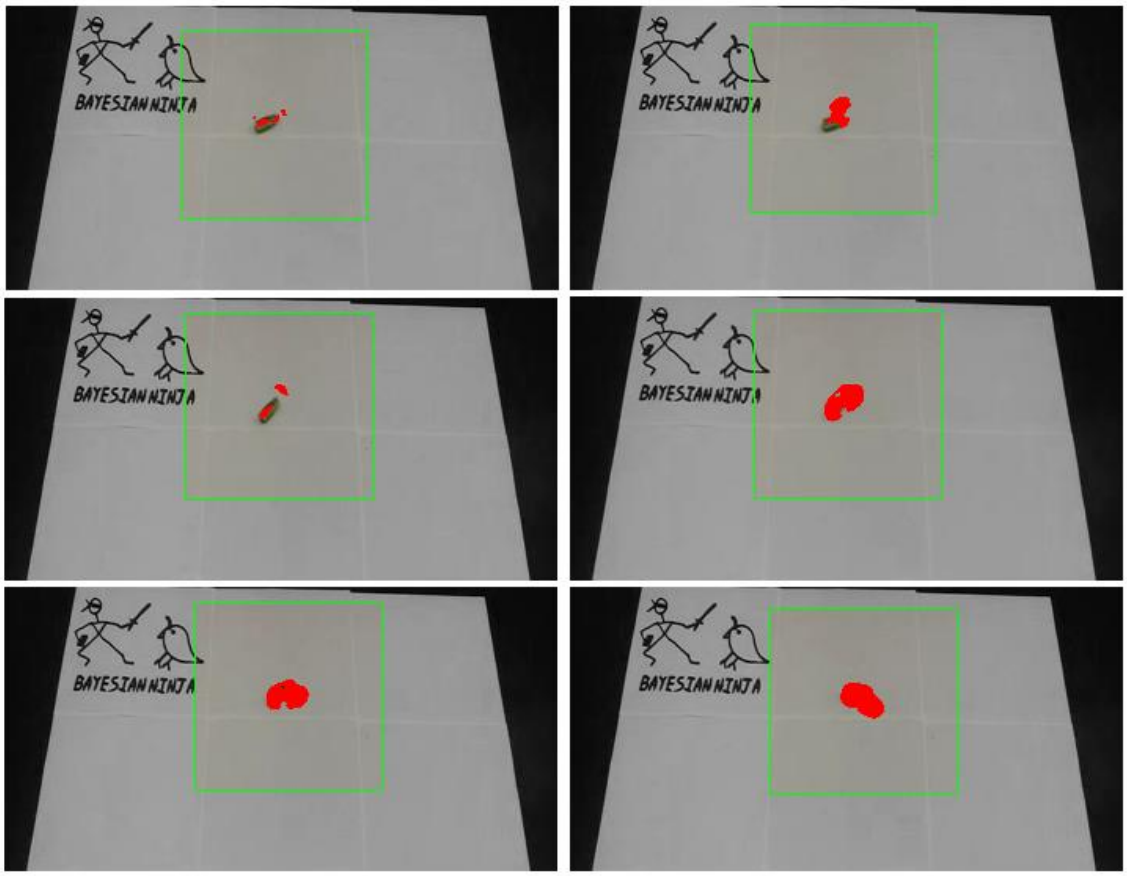


Figura 7.5. Resultado si el desplazamiento entre fotogramas es conocido

- **Tercer grupo**

En el tercer grupo de secuencias de prueba se encuentran los videos que reflejan situaciones que se acercan más a escenarios reales. En estos casos tenemos varios elementos en movimiento dentro del área de vigilancia que se desplazan de un lugar a otro siguiendo principalmente trayectorias rectilíneas.

Video: *Vid_exterior3.avi*

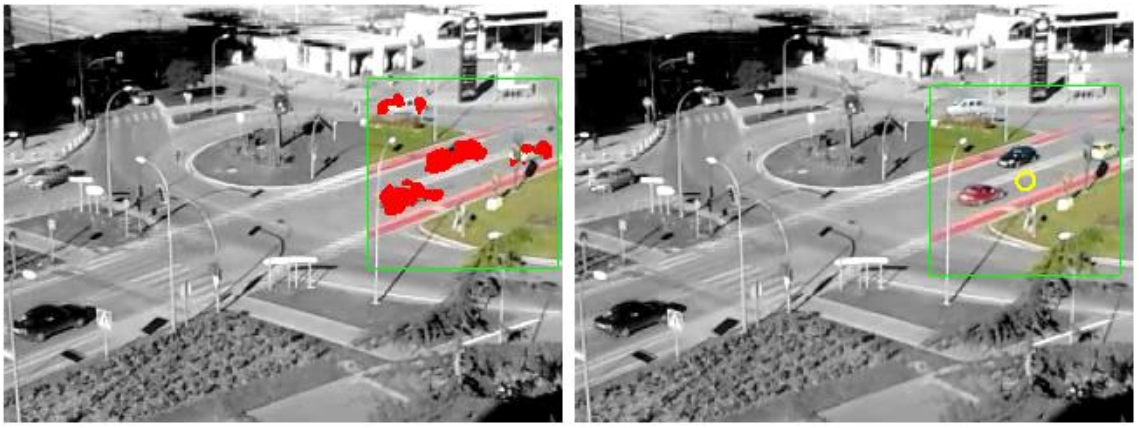
La escena capturada en este ejemplo consiste en un tramo de calle con varios carriles en ambos sentidos, donde se pretende detectar y seguir los vehículos que cruzan de un lado a otro. Los principales retos del sistema en este video son que debe detectar igualmente vehículos de distinto tamaño (coches, camiones, motos…) y que debe engancharse a vehículos en movimiento con diferentes velocidades.

Otro detalle de interés es que al tener videos en los que la cámara se mueve (vibra) levemente sobre la que debería ser su posición ideal (ruido en la posición provocado por una mala sujeción de la cámara), el método de corrección de posición sin conocer el desplazamiento es más preciso. Con este método al hacer el bucle de solapamiento podemos corregir mejor estas pequeñas (o grandes) desviaciones en la posición de la imagen por fallos en la cámara;

mientras que con el método de corrección de posición conociendo el desplazamiento, éste viene dado únicamente por la diferencia entre la posición de los filtros y no tiene en cuenta otros factores externos, por lo que estos fallos estarán más presentes.

Video: *Vid_exterior10.avi*

En este ejemplo, tenemos varios vehículos simultáneamente en el área de vigilancia, cada uno con su propia velocidad, dirección de movimiento, sentido, etc. Si dentro de la ventana de seguimiento hay varios coches, el sistema detectará el movimiento de todos ellos pero sólo será capaz de seguir a uno. Ante la existencia de más de un blanco, el algoritmo de seguimiento actúa sobre el que presente una mayor área en movimiento detectada, y en la práctica esto depende especialmente del tamaño del vehículo, del contraste de su color y el color del fondo (es decir, es competencia directa del algoritmo de detección). En los ejemplos de la figura 7.6 (a) y (b) tenemos dentro de la ventana de detección 4 y 5 coches en movimiento, respectivamente. El algoritmo SSIM detecta su posición en la imagen de la izquierda y en la derecha se muestra la posición del filtro que se consigue adoptar sobre los blancos, en cada caso.



(a)

(b)

Figura 7.6. Detección múltiple de blancos y asignación de posición del filtro

En el caso de tener dos vehículos en movimiento muy próximos, las zonas detectadas pueden solaparse situándose la posición del filtro en un punto intermedio entre ambos, situación que se mantendría hasta que estén suficientemente separados y el sistema se centre automáticamente sobre uno de ellos.

Figura 7.7. Seguimiento en blancos que se cruzan

En la figura 7.7 se puede ver una secuencia en la que dos vehículos cruzan su trayectoria, después de unos momentos donde ambos casi se solapan y apenas se puede diferenciar entre ambos coches, continúa siguiendo al blanco original (el coche que se mueve de izquierda a derecha).

CAPÍTULO 8

CONCLUSIONES Y LÍNEAS FUTURAS

En este último capítulo se expondrán las conclusiones del trabajo realizado en este proyecto, y se propondrán diferentes pasos a seguir para mejorar el trabajo en un futuro. A semejanza de un resumen final, se hará una recopilación del trabajo y de la investigación llevada a cabo durante la realización del proyecto, y se realizará una revisión de los resultados globales obtenidos tras las pruebas del capítulo anterior.

8.1 CONCLUSIONES

Al comienzo de este proyecto nos planteamos el objetivo de diseñar un sistema de detección y seguimiento de movimiento en secuencias de vídeo, que simulara el funcionamiento de una cámara de vigilancia que se mueve siguiendo a un único elemento en movimiento. Igualmente, otro de los requisitos es que el sistema desarrollado fuera sencillo y ligero para que pueda ser implementado a posteriori en dispositivos móviles con cámara integrada como teléfonos, tabletas, etc. El entorno que se eligió para el desarrollo del prototipo del sistema fue Matlab, por su elevada versatilidad en el tratamiento de imágenes y vídeos.

Previamente al inicio del desarrollo del sistema, se hizo un repaso a las etapas que conforman un sistema típico de video-vigilancia (siendo las más importantes la detección y seguimiento, las cuales se pretende implementar en este proyecto) y a las técnicas empleadas en la literatura especializada para resolver los problemas de campo de la vigilancia. Respecto al algoritmo de seguimiento, se consideró implementar desde un principio los filtros alfa-beta y de Kalman, ampliamente conocidos en aplicaciones de seguimiento. Después de describirlos en detalle, el estudio se amplió a otros algoritmos de seguimiento de interés, como el filtro de partículas. Para comprobar el buen funcionamiento de los dos filtros y que sus ecuaciones son correctas, se efectúan varias pruebas sobre ambos algoritmos con el fin de evaluar la importancia y finalidad de sus parámetros, la capacidad de reacción ante cambios, etc.

En cuanto al algoritmo de detección, a pesar de la gran diversidad de posibilidades revisadas, para realizar este proyecto nos decantamos por una opción que, al parecer, no ha sido demasiado explorada, la similitud estructural. Por tanto, hemos utilizado un método de

evaluación de la calidad de imagen para establecer las diferencias entre fotogramas consecutivos y determinar los elementos que se están moviendo entre ambos. Adicionalmente, se realizó un repaso de distintas técnicas de medida de evaluación de la calidad de imagen. La validez del método SSIM como algoritmo de detección se comprobó en un trabajo anterior, mediante un conjunto de pruebas en entornos variados, que se recuperan en el anexo. Con ello se determinó que su rendimiento como algoritmo de detección era satisfactorio y que puede usarse en diferentes entornos de igual manera que otros algoritmos típicos de detección.

El proceso de unificar en un mismo sistema los dos bloques de detección y seguimiento se realiza en varias fases, para poder evaluar cómo funcionan de forma separada al enfrentarse a distintas situaciones: en la fase I se utilizan vídeos fijos (sin pretender que se mueve la cámara) y se aplican los algoritmos de detección y el SSIM, para comprobar que es posible detectar correctamente el objeto en movimiento, pasar la posición al filtro de seguimiento y tener localizado al blanco en todo momento. En la fase II se utilizan partes de fotogramas de un video consecutivos, conseguidos de forma artificial, que contienen un objeto en movimiento para verificar que el algoritmo de detección puede identificar un blanco cuando el fondo de la imagen varía entre los fotogramas comparados. Finalmente en la fase III se unifican ambos conceptos, detección con fondos variables y seguimiento.

Una vez desarrollado el prototipo completo que realiza todas las funciones previstas, se procede a evaluar su comportamiento mediante un conjunto de vídeos que reflejan situaciones más complejas para comprobar la respuesta ante distintos escenarios reales y encontrar las limitaciones del sistema. Estos son los casos referidos en el capítulo 7. Las pruebas realizadas comprenden principalmente la detección y seguimiento de personas y vehículos en espacios abiertos exteriores. En las mejores condiciones, la efectividad del sistema es elevada y no se detectan grandes errores. Sin embargo, a continuación vamos a hacer referencia a situaciones que pueden provocar una inferior efectividad del sistema.

Los puntos más conflictivos de todo el sistema son los procesos de adquisición de las imágenes a comparar y de detección de movimiento. En cuanto al primero de ellos, calcular de manera eficaz las dos imágenes que se van a comparar es vital para el funcionamiento del sistema: es la parte que más tiempo consume un nuestra aplicación y una compensación del desplazamiento errónea puede imposibilitar acciones posteriores. Respecto a la detección usando SSIM, los problemas más frecuentes que se han encontrado (según el anexo A) se deben a la aparición de falsos positivos por la presencia de sombras, reflejos… que el sistema considera como objetos reales en movimiento, además de movimientos involuntarios de la cámara no corregidos que pueden provocar falsas detección en esquinas, bordes, etc. En los casos más extremos, el filtro de seguimiento puede acoplarse a estos fallos y el error se incrementa; sin embargo, en la mayoría de casos, el filtro es capaz de obviar estos fallos al seguir datos correctos de instantes anteriores. La última parte referente al algoritmo de seguimiento plantea menos problemas. Aunque se indicaron las diferencias entre los filtros de Kalman y alfa-beta, en la práctica, el uso de uno u otro no influye de manera determinante en

el resultado final, más allá de situaciones excepcionales, por lo que se ha optado por utilizar principalmente el alfa-beta.

Después de haber analizado en profundidad el funcionamiento del sistema, estamos en disposición de hacer un balance de los puntos positivos y negativos, y recapitular las ventajas e inconvenientes que hemos encontrado en el sistema diseñado:

- Ventajas: Entre las ventajas más destacables, 1) en primer lugar requiere de un código sencillo de implementar y de poco peso, lo cual es positivo de cara a una posible adaptación a otro lenguaje para desarrollar una aplicación. 2) Hablando propiamente del sistema, se ha comprobado que tiene capacidad de detectar con exactitud elementos en movimiento de muy variada índole y en diferentes condiciones, además de realizar un seguimiento preciso utilizando algoritmos sencillos, lo que mejora el rendimiento global del programa sin complicarlo en exceso. 3) Gracias al algoritmo de seguimiento y a la estabilidad que proporciona su uso, el sistema es resistente a problemas como oclusiones, falta de información momentánea, errores ocurridos en el vídeo, alteraciones en la grabación, etc. 4) La implementación de un método que simula el uso de una cámara móvil aumenta la versatilidad del sistema, permitiendo que el mecanismo de detección y seguimiento pueda aplicarse a un mayor número de situaciones, más allá de la típica vigilancia fija.
- Inconvenientes: Entre los puntos a mejorar se encuentran el 1) elevado tiempo de procesado requerido en el caso de realizar la corrección de desplazamiento sin conocer dicho movimiento, que puede ralentizar el funcionamiento del sistema conjunto. 2) Un error en el cálculo del desplazamiento entre instantes puede provocar que la comparación dos *frames* consecutivos no sea correcta y la detección posterior sea equivocada. 3) En ocasiones se detecta que en vídeos con fondos con pocos detalles aparecen problemas a la hora de distinguir los elementos fijos de los elementos en movimiento. 4) Pueden aparecer fallos ocasionales producidos por algún movimiento brusco de cámara, y falsas detecciones momentáneas de difícil previsión, si bien no alteran el buen funcionamiento del sistema.

Por último, en vista a las pruebas realizadas, podemos concluir que un sistema de vigilancia basado conjuntamente en SSIM y en un algoritmo de seguimiento, es un método eficaz para la detección y seguimiento de movimiento con cámara móvil y que puede emplearse de igual forma que otros métodos más típicos en aplicaciones de vigilancia.

8.2 LÍNEAS FUTURAS

A partir del funcionamiento básico del prototipo presentado, existen puntos que pueden mejorarse y desarrollarse con mayor profundidad:

- Algoritmos de seguimiento: En el sistema diseñado sólo se han empleado dos esquemas de seguimiento. En futuras ampliaciones de este trabajo, podría ser

interesante incorporar nuevos algoritmos, tales como el filtro de partículas, que mejoren el rendimiento de los que se han utilizado y que sean capaces de adaptase mejor a diferentes tipos de movimientos y dinámicas.

- Sistema de giro: Sería interesante desarrollar un dispositivo mecánico que, montado junto a una cámara de vídeo, permitiera a ésta girar sobre su eje horizontal y vertical para seguir el movimiento de personas. De alguna manera consistiría en un servo que funcionara de manera conjunta con la información de posición indicada por el filtro de seguimiento, que indicaría la posición que debe tener en todo momento.
- Compensación de movimiento: Debería mejorarse el sistema de compensación de movimiento, es decir, ser capaces de detectar un desplazamiento cuando la cámara se mueve sin tener en cuenta los cambios que se producen en el fondo de la imagen. Especialmente cuando no se conozca el movimiento concreto, en cuyo caso sería conveniente perfeccionar el tratamiento del bucle para reducir los tiempos de procesado y mejorar el rendimiento del sistema.
- Desarrollo de una aplicación para plataformas móviles: Una vez se ha completado el desarrollo del prototipo, el objetivo final sería implementar una aplicación basada en el mismo que sea capaz de funcionar en dispositivos móviles. El mecanismo empleado en este trabajo es lo suficientemente ligero y robusto como para que pueda ser adaptado para su uso en teléfonos móviles, tabletas, etc.
- Eliminación de sombras: A pesar de que el funcionamiento del sistema sea correcto en una situación concreta y que todos los blancos se detecten, las sombras que estos proyectan sobre el suelo también se perciben. El hecho de que se detecten las sombras es una prueba de que el sistema funciona, pero de cara a la vigilancia de personas estaríamos considerando que tenemos más blancos de los que realmente hay. Sería conveniente buscar algún tipo de solución que permita delimitar las sombras de los objetos en movimiento y eliminarlas.
- Eliminación del movimiento de la cámara: Aunque el sistema desarrollado puede funcionar con pequeñas vibraciones de la cámara, grandes alteraciones pueden hacerlo ineficiente. Por tanto, es conveniente asegurar que no se producen ningún tipo de movimientos de la cámara. Para solucionarlo podemos elegir desde incorporar a la cámara mecanismos estabilizadores de imagen o añadir algún tipo de procesado para alinear imágenes.

REFERENCIAS

[1] M. Valera, S.A. Velastin, "Intelligent distributed surveillance systems: a review", *IEE Proc.-Vis. Image Signal Process., Vol. 152, No 152, April 2005*.

[2] A. Jazayeri, H. Cai, J. Y. Zheng, M. Tuceyan, "Vehicle Detection and Tracking in Car Video Based motion model", *IEEE Transactions on Intelligent Transportation Systems, Vol. 12, No. 2, June 2011.*

[3] D. Beymer, P. McLauchlan, B. Coifman, J. Malik, "A real-time computer vision system for measuring traffic parameters".

[4] S. Srivastava, K. K. Ng, E. J. Delp, "Crowd flow estimation using multiple visual features for scenes with changing crowd densities", *8th IEEE Conference on Advanced Video and Signal-Based Surveillance 2011*.

[5] H. Yang, L. Shao, F. Zheng, L. Wang, Z. Song "Recent advances and trends in visual tracking: A review*", Neurocomputing 74 (2011) 3823–3831*.

[6] Z. Zhang, M. Piccardi "A Review of Tracking Methods under Occlusions", *Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, Japan.*

[7]A. Yilmaz, O. Javed, M. Shah, "Object tracking. A review", *ACM Computing Surveys, Vol. 38, No. 4, Article 13, December 2006.*

[8] R. Ormaechea, "Desarrollo de una Aplicación de Detección de Movimiento Basada en Comparativa Estructural de Imágenes", Univerisdad de Valladolid, Septiembre 2012.

[9] Z. Wang, E. P. Simoncelli, A. C. Bovik, "Multi-scale structural similarity for image quality assessment", *Proceedings of 37th IEEE conference of Signals, Systems and Computers 2003*.

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE Transactions on Image Processing, Vol. 13, No. 4, April 2004.*

[11] P. R. Kalata, "Alpha-Beta Target Tracking: A Survey", *American Control Conference, 1992*.

[12] P. R. Kalata, “The Tracking Index: A Generalized Parameter for Alpha-Beta and Alpha-Beta-Gamma Target Trackers”, *IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-20, No.2, March 1984.*

[13] W. D. Blair, “Fixed-Gain, Two-Stage Estimators for Tracking Maneuvering Targets”, July 1992.

[14] M. S. Grewal, A. P. Andrews, “Kalman Filtering: Theory and Practice Using MATLAB”, Third Edition, 2008, John Wiley & Sons.

[15] B. Ristic, S. Arulampalam, N. Gordon, “Beyond the Kalman Filter. Particle Filters for Tracking Applications”, Artech House Radar Library, 2004.

[16] S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”, *IEEE Transactions on Signal Processing, Vol. 50, No. 2, February 2002.*

[17] R. C. Gonzalez, R. E. Brooks, “Digital Image Processing”, Addison-Wesley, Reading, Massachusetts, 2008.

[18] R. C. Gonzalez, R. E. Brooks, “Digital Image Processing using Matlab”, Upper Saddle River, NJ, Pearson, 2004.

[19] “Image processing toolbox. User´s guide”, MathWorks, 2011.

[20] T. Ko, “A survey on behavior analysis in video surveillance applications”, *Raytheon Company, USA.*

[21] D. Comaniciu, P. Meer, “Mean Shift: A Robust Approach toward Feature Space Analysis”.

[22] D. Comaniciu, V. Ramesh, P.Meer, “Kernel-Based Object Tracking”.

[23] F. Gustafsson, “Adaptive Filtering and Change Detection”, John Wiley & Sons, 2000.

[24] G. Welch, G. Bishop, “Introduction to Kalman Filter”, Department of computer Science, University of North Carolina, 2006.

[25] G. Aubree, P. Longepe, N. Puorrain, L. Fourdan, “Implementing a Digital Tracker for Monopulse Radar Using the TMS320C40 DSP”, Texas Instruments, EFRIE, France, 1995.

[26] A. Doucet, A. M. Johansen, “A Tutorial on Particle Filtering and Smoothing: Fifteen years later”, March 2012.

[27] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, L. Van Gool, “Robust Tracking-by-Detection using a Detector Confidence Particle Filter”, *IEEE 12th International Conference on Computer Vision (ICCV), 2009.*

[28] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Klarsson, P. Nordlund, "Particle filters for positioning, navigation and tracking", Department of Electrical Engineering, Linköping University, Sweden.

[29] Z. Wang, Q. Li, "Information Content Weighting for Perceptual Image Quality Assessment", *IEEE Transactions on Image Processing, Vol. 20, No. 5, May 2011.*

[30] Z. Wang, A. C. Bovik, "Mean squared error: Love it or leave it? – A new look at signal fidelity measures", *IEEE Signal Processing Magazine, Vol. 26, No.1, pages: 98-117, January 2009.*

[31] S. Aja Fernández, R. San José Estepar, C. Alberola López, C. F. Westin, "Image Quality Assessment Based on Local Variance", *Proceedings of the 28th IEEE EMBS Annual International Conference, New York City, USA, Aug 30-Sept 3, 2006.*

[32] H. R. Sheikh, A. C. Bovik, "Image Information and Visual Quality", *IEEE Transactions on Image Processing, Vol. 15, No.2, February 2006.*

[33] L. A. Zadeh, "Fuzzy Sets", *Information and Control 8, p. 338-353, 1965.*

[34] D. V. Weken, M. Nachtegael, E. Kerre, "Some New Similarity Measures for Histograms", *Proceedings of ICVGIP, 441-446, 2004.*

[35] D. V. Weken, M. Nachtegael, V. de Witte, S. Schulte, E. Kerre, "Constructing similarity measures for color images", *Fuzzy logic, soft computing and computational intelligence: eleventh international fuzzy systems association world congress, 2005.*

[36] H. R. Sheikh, A. C. Bovik, "A Visual Information Fidelity Approach to Video Quality Assessment", the first International Workshop on Video Processing and Quality Metrics for Consumer Electronics, January 23-25, 2005, Scottsdale, AZ.

[37] https://ece.uwaterloo.ca/~z70wang/research/ssim/ fecha de última visita: 8 de enero de 2015. Electrical and Computer Engineering, University of Waterloo, Canada.

[38] http://studentdavestutorials.weebly.com/ fecha de última visita: 8 de enero de 2015.

ANEXO A

ESTUDIO DE DETECCIÓN

El objetivo de este anexo consiste en evaluar la capacidad del algoritmo de detección de funcionar en diferentes entornos, mediante un conjunto de pruebas orientadas a determinar si el algoritmo detecta movimiento real y si es capaz de discriminar falsas detecciones en diferentes entornos. Las pruebas realizadas en este anexo únicamente se aplican sobre el algoritmo SSIM y no sobre los algoritmos de seguimiento, por lo que en ningún caso se tendrá en cuenta la posterior incorporación de mecanismos de seguimiento. Inicialmente se expondrán algunos casos y ejemplos específicos para ilustrar las situaciones a las que nos enfrentamos y que se producen durante las pruebas. Por último, se recogerán todos los experimentos realizados para presentarlos de manera global y evaluar el rendimiento del algoritmo.

En primer lugar describimos el banco de datos que se ha empleado para validar el correcto funcionamiento de la aplicación. Para realizar las distintas pruebas se ha confeccionado una base de datos formada por vídeos grabados desde una cámara de un teléfono móvil. Los vídeos son de duración variable, llegando hasta el minuto de duración, tienen una tasa de 25 frames por segundo y una resolución de 320x240 píxeles. Los vídeos se han dividido en distintas categorías, dependiendo de la situación en la que se produzcan y las características presentes en ellos, como por ejemplo:

- Interior
- Exterior
- Poco movimiento
- Mucho movimiento
- Iluminación

A.1. RUIDO EN EL MAPA SSIM

El primer comentario que vamos a realizar consiste en analizar el comportamiento del sistema sin ningún tipo de movimiento. Cuando estamos grabando un escenario de interior como una habitación sin que ocurra ningún acontecimiento y la cámara está inmóvil, lo esperado sería que el algoritmo SSIM no detectara ningún tipo de movimiento, que el índice para todos los pixeles fuera 1, o lo que es lo mismo, que el mapa SSIM fuera totalmente blanco. Sin embargo, esto no ocurre así, ya que el comportamiento real se aleja levemente de la predicción anterior.

Durante la grabación de la secuencia de vídeo aunque no haya ningún movimiento de objetos o de la cámara, se introduce cierta cantidad de ruido provocado por la compresión, por cambios de formato, etc. que van a distorsionar fotogramas de la secuencia que en principio deberían ser idénticos. Estos fenómenos son captados por el algoritmo SSIM y en consecuencia pueden apreciarse en el mapa SSIM.

Figura A.1. Ruido en el mapa SSIM sin movimiento

Si comprobamos algunos valores de los píxeles sobre las imágenes obtenidas podemos observar que los píxeles más oscuros tienen un valor alrededor de 0.9. Por otro lado, al representar el histograma de uno de los mapas SSIM anteriores vemos que los valores de los píxeles se sitúan cerca de la unidad, por lo que su influencia sobre el rango en el que trabaja SSIM (0 – 1) es limitado. Por este motivo, podemos diferenciar claramente el ruido en una escena inmóvil de los objetos en movimiento, que, como veremos más adelante, tienen unos valores de SSIM más cercanos a 0.

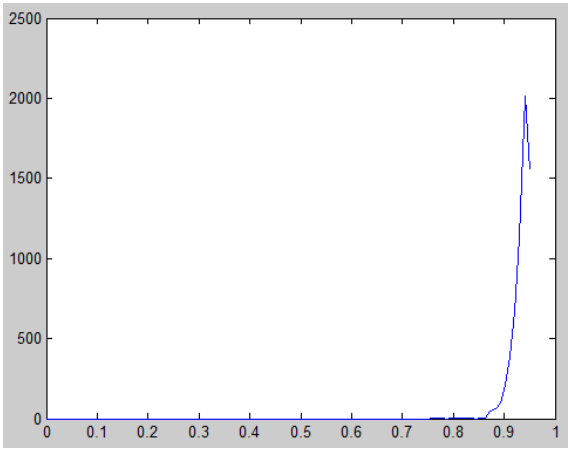


Figura A.2. Histograma de mapa SSIM sin movimiento

A.2. ELECCIÓN DEL UMBRAL

Uno de los puntos más importantes de la aplicación consiste en determinar el umbral adecuado con el cual decidir si hay o no movimiento en una imagen. De la elección del umbral depende la correcta detección y el correcto funcionamiento del sistema. Si elegimos un umbral demasiado bajo estaremos descartando zonas en las que se produce movimiento, aunque sea menos intenso, lo que nos provoca falsos negativos. En cambio, si el umbral es demasiado alto estamos clasificando como movimiento de un objeto algún otro tipo de variación que provoca una distorsión en el mapa SSIM, y que genera falsos positivos o falsas detecciones.

A continuación veremos algunos ejemplos en los que nos plantearemos elegir el umbral más apropiado. De partida, vamos a asumir que el umbral va a ser inferior a 0.8 y en los histogramas siguientes no vamos a representar todo el intervalo [0 – 1] para no tener en cuenta los píxeles que corresponden al fondo inmóvil, con valor cercano a 1 y que son los más numerosos, y a alteraciones puntuales.

El ejemplo que se muestra en la figura A.3 es un caso muy favorable, en el cual tenemos un movimiento claro y definido. En el mapa SSIM asociado, los píxeles de movimiento destacan mucho sobre el resto porque tienen un color negro muy intenso. En el histograma se puede ver que la mayoría de los valores de estos píxeles están cerca de cero. Por tanto, podemos bajar mucho el umbral sin que se pierda prácticamente información de las zonas de movimiento.

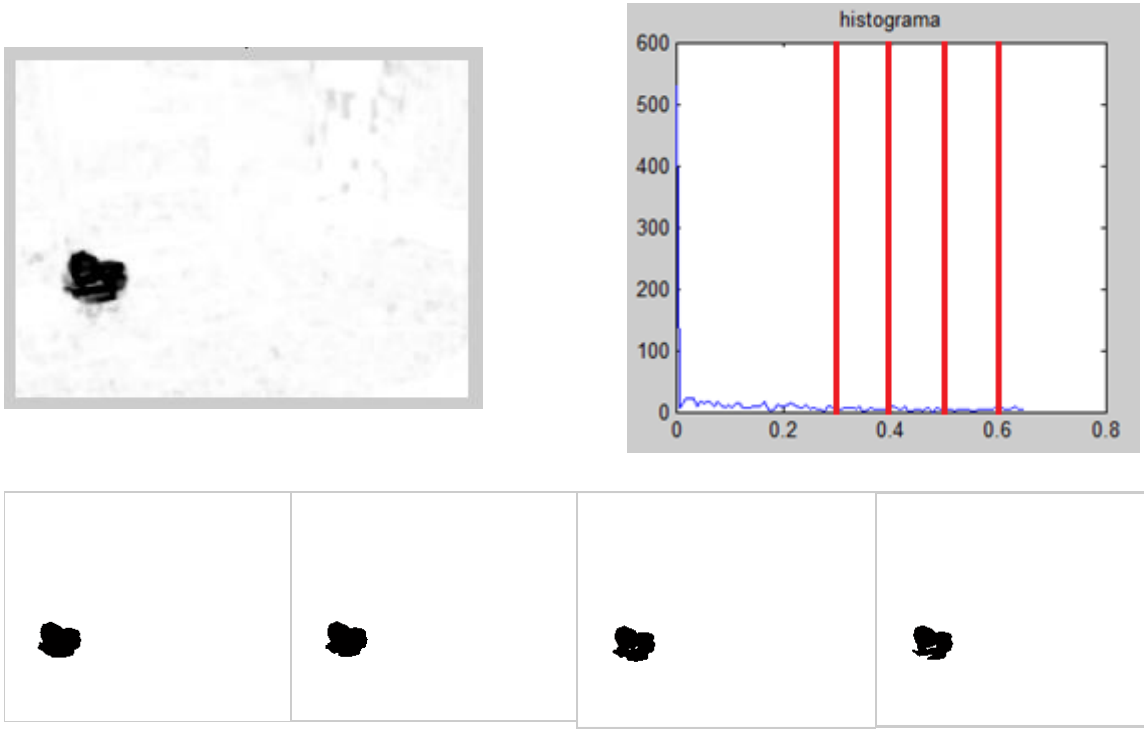


Figura A.3. Ejemplo de elección de umbral 1

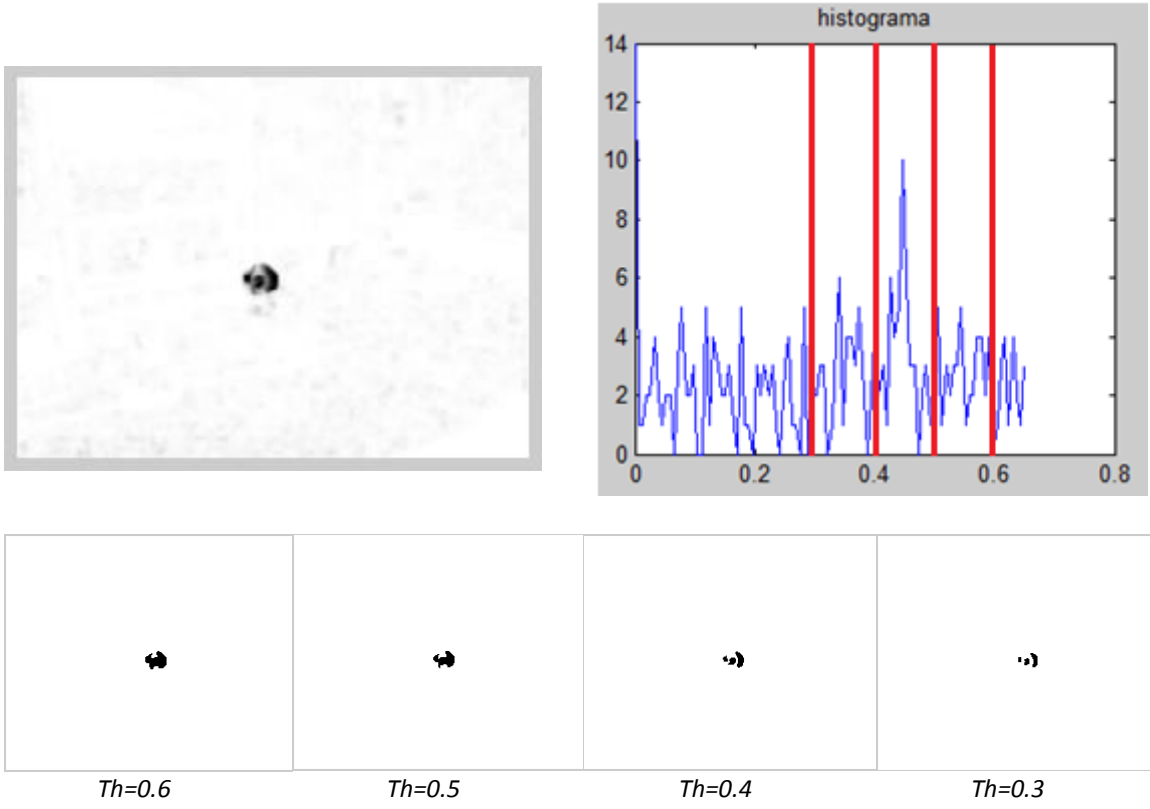


Figura A.4. Ejemplo de elección de umbral 2

En el caso de la figura A.4, vemos que los píxeles de la zona de movimiento se encuentran más repartidos por los distintos niveles de gris del histograma. Si variamos el umbral disminuyéndolo progresivamente, entonces estamos eliminando grupos de píxeles que son relevantes para la representación del objeto. A diferencia del ejemplo anterior, ya no podemos elegir un umbral con tanta libertad sino que estamos obligados a afinar más nuestra decisión.

En situaciones como la que se muestra en la figura A.5, tenemos que el desplazamiento del objeto representado es muy reducido. Como se puede ver, el mapa SSIM lo identifica de forma muy difusa y no existen píxeles que pertenezcan a los niveles de gris del histograma más próximos a cero, como ocurría en casos anteriores. Para que nuestro sistema funcione de forma eficaz, debería ser capaz de detectar este tipo de movimientos más sutiles. En el ejemplo se muestra que únicamente a partir de un umbral de 0.5 comenzamos a percibir los píxeles del objeto. En una situación como esta, el comportamiento más natural consiste en subir el umbral para permitir aumentar el número de píxeles que se analicen, con el riesgo de que cualquier comportamiento anómalo nos produzca una falsa detección.

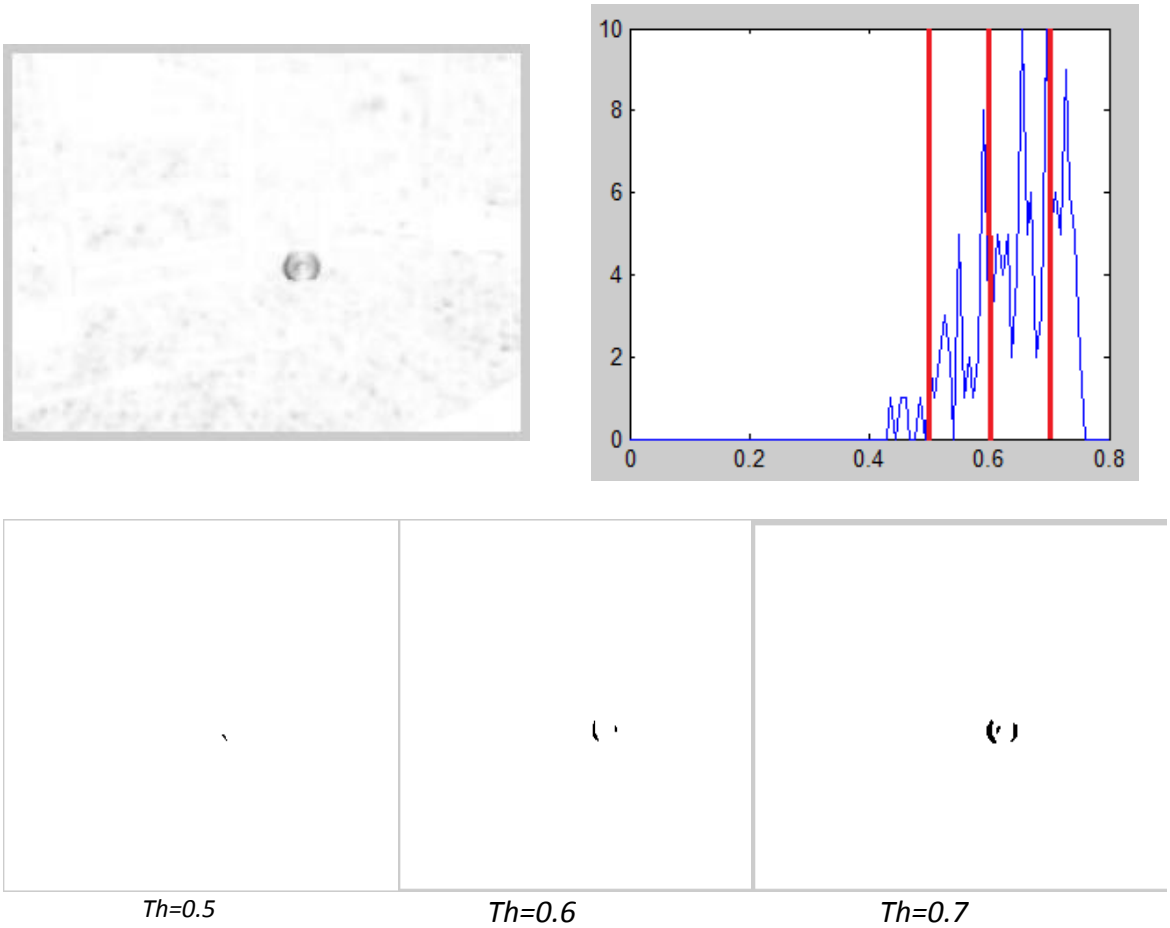


Figura A.5. Ejemplo de elección de umbral 3

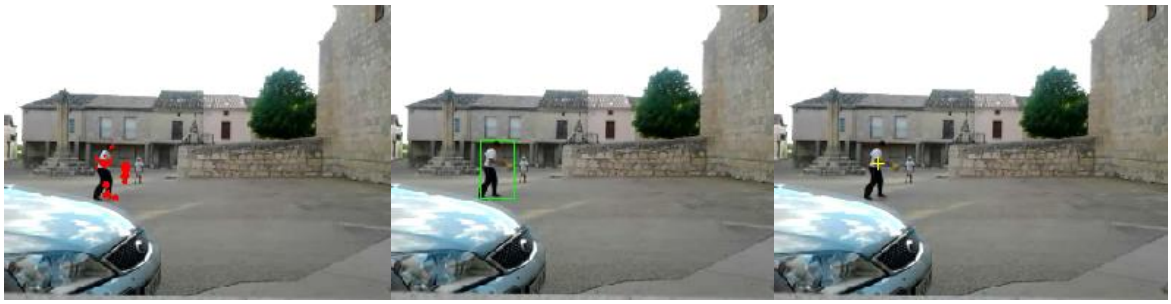
Finalmente hemos considerado que el mejor umbral para hacer nuestros experimentos es tomar un valor entre 0.5 y 0.6, ya que para estos umbrales obtenemos un número suficiente

de píxeles para detectar la presencia de un objeto. En algunos casos, este valor incluso se puede superar si tenemos certeza de que no hay ninguna alteración ni movimiento de la cámara.

A.3. REPRESENTACIÓN DE OBJETOS

Como indicamos en el capítulo sobre las técnicas de detección de movimiento y seguimiento de objetos, los elementos presentes en una escena pueden representarse de diferente forma, mediante por ejemplo: siluetas, puntos, formas geométricas… En este trabajo vamos a optar por la representación utilizando las zonas generadas por el mapa SSIM, puntos y rectángulos.

Los píxeles que representan un movimiento, conseguidos a partir del mapa SSIM, los hemos marcado en rojo sobre el fotograma en cuestión; y aunque esta representación se puede emplear en todo tipo de situaciones, es especialmente más útil cuando el objeto o persona ocupa un elevado espacio del total del fotograma. El uso de un rectángulo que envuelve los puntos detectados por el mapa SSIM y que se adapta a la forma del objeto, y de un punto que marca su centro es más apropiado cuando el tamaño es menor.



(a) Mapa SSIM *(b) Rectángulo* *(c) Punto*

Figura A.6. Distintas formas de representación

A.4. TAMAÑO DE LOS ELEMENTOS

Los elementos en movimiento que nos encontremos dependerán en buena media del tipo de vídeo que estemos tratando. En vídeos de interior consideramos que el espacio que se está analizando es menor, y los elementos más frecuentes serán personas y objetos. Dentro de una habitación o un entorno cerrado, una persona puede ocupar un porcentaje considerable de la escena. En vídeos de exterior, si por ejemplo realizamos la vigilancia de una calle, tendremos la cámara colocada en una posición desde donde tengamos una perspectiva amplia de la calle y los objetivos serán principalmente vehículos y personas. Es razonable pensar que el tamaño relativo de los objetos difiere dependiendo del escenario en el que nos encontremos. Así el tamaño de una persona dentro de una habitación será distinto su tamaño en la calle.

En las siguientes imágenes se pueden observar algunos ejemplos del porcentaje del fotograma que ocupan diferentes elementos en movimiento.

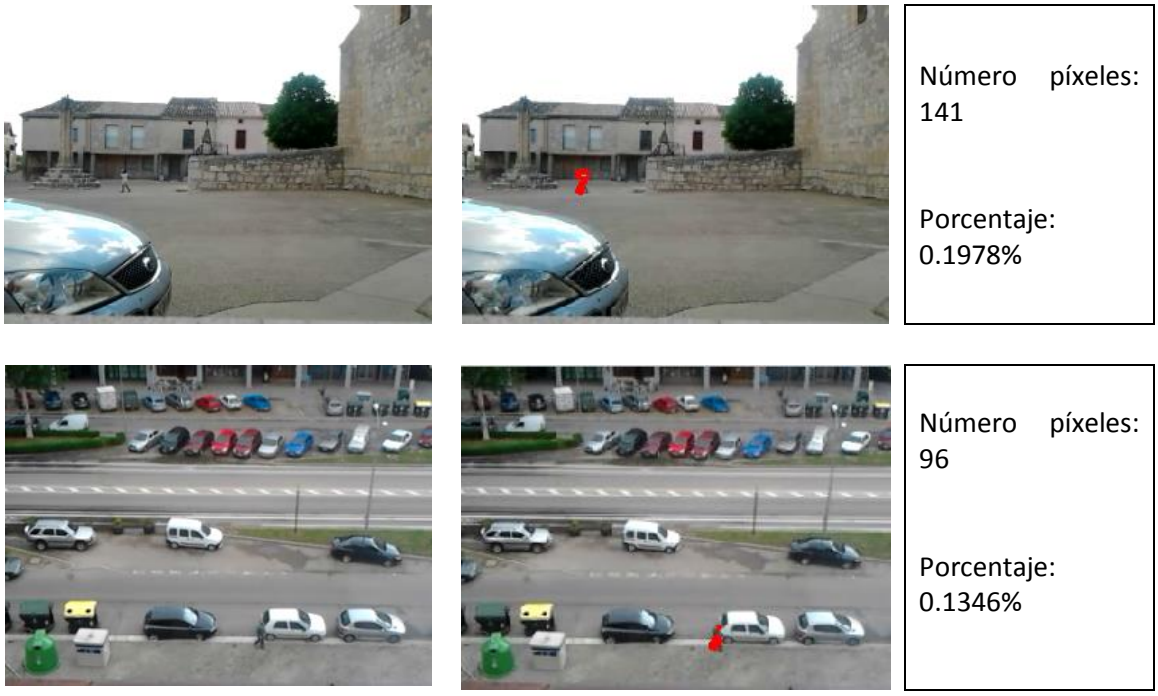


Figura A.7. Tamaño relativo de personas en exterior

En situaciones como las de la figura anterior, en las que el tamaño de los objetos es muy pequeño con respecto a la totalidad de la imagen, cobra mayor importancia el color de la persona respecto al del fondo. Si el objeto resalta sobre el fondo será más fácil para el algoritmo detectar los cambios, pero si ambos tienen colores parecidos pueden confundirse y la detección no se realiza de forma precisa.

En la figura A.8 tenemos la presencia de una persona en una habitación interior, en el que vemos principalmente el movimiento del brazo y parte del cuerpo. En este tipo de situaciones en las que aparece un cuerpo de mayor tamaño, en algunas ocasiones todo ello tendrá movimiento y en otras solo algunas partes se moverán, por lo que el tamaño detectado en diferentes situaciones variará.

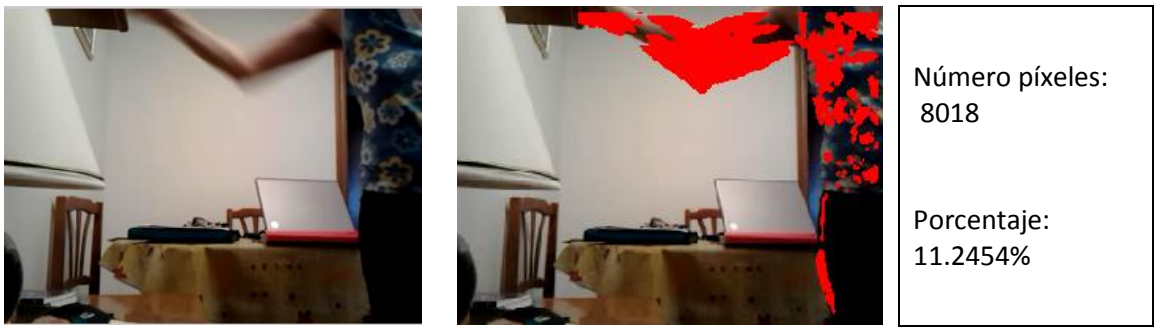


Figura A.8. Tamaño relativo de personas en interior

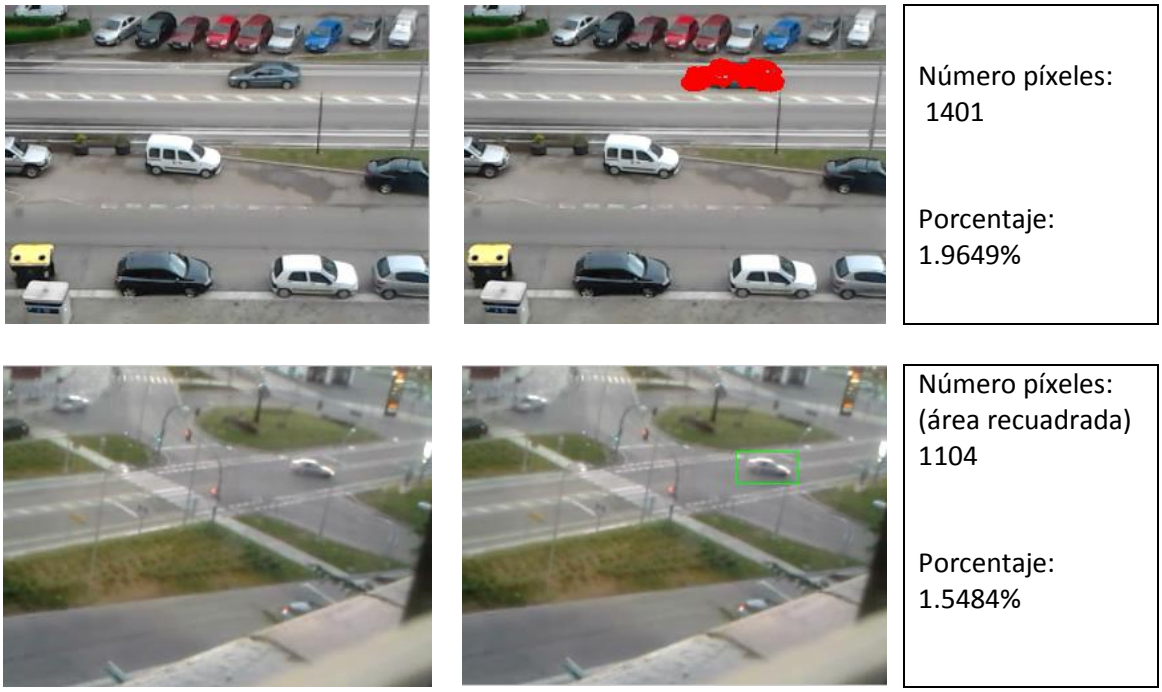


Figura A.9. Tamaño relativo de coches

En situaciones más difíciles donde el algoritmo no detecta la totalidad de los puntos del objeto que se está moviendo, es más recomendable encerrar el objeto dentro de un rectángulo. Aplicando esta variación calcular el tamaño mediante una aproximación por exceso.

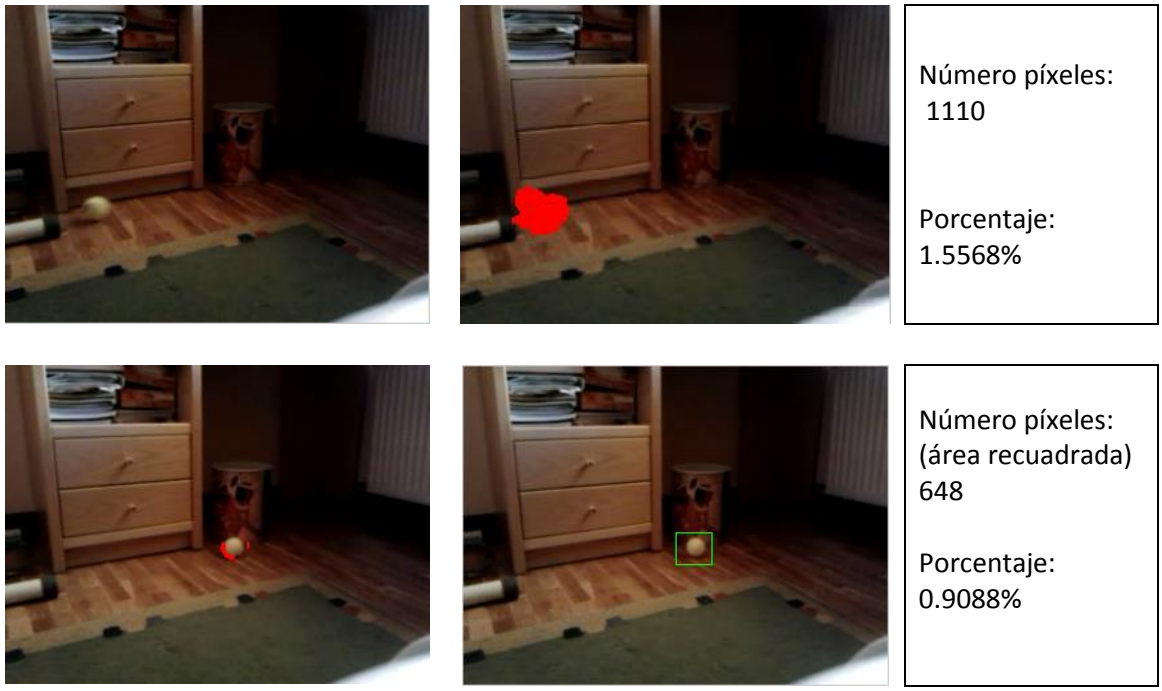


Figura A.10. Tamaño relativo de objetos en interior (pelota de tenis)

Como conclusión a este apartado, podemos decir que conocer el tamaño de objetos puede aplicarse para distinguir entre falsas detecciones o elementos que no interesa evaluar. Si caracterizamos el tamaño de los elementos recurrentes que son importantes en un escenario concreto, la detección de un número de píxeles sustancialmente menor al objeto más pequeño puede considerarse como poco relevante y despreciarse.

A.5. SOLAPAMIENTO DE OBJETOS

Cuando estamos representando los distintos elementos en movimiento de una escena utilizando rectángulos, el área encuadrada se estima mediante los puntos obtenidos en el mapa SSIM que tienen cierta proximidad. En el caso de tener dos objetos cercanos entre sí, llegará un punto en el cual el algoritmo no pueda distinguir entre ambos, los considerará como un único elemento y se representará utilizando un solo rectángulo que englobe a los dos.

(a) *(b)*

Figura A.11. Superposición de objetos

En la figura A.11(a), el algoritmo distingue con claridad entre el niño y la pelota porque están suficientemente distanciados. Sin embargo, en A.11(b) ya no es posible determinar qué puntos del mapa SSIM corresponden con el niño y cuáles con la pelota y el sistema nos los presenta como si hubiera solamente un elemento de mayor tamaño.

A.6. DIVISIÓN DE OBJETOS

El proceso de creación del mapa SSIM se realiza comparando dos fotogramas consecutivos de la secuencia de vídeo (para mayor agilidad, la separación entre dos fotogramas que se van a comparar es de tres). Al observar el desplazamiento de un objeto uniforme entre dos fotogramas, podemos ver en el mapa SSIM que las diferencias entre las dos imágenes y los

lugares donde detecta movimiento se corresponden con los extremos del objeto, mientras que el centro del mismo se mantiene similar. Esto puede provocar que el mapa SSIM determine dos zonas de movimiento y se detecte la presencia de dos objetos pequeños y próximos en lugar de uno solo de mayor tamaño.

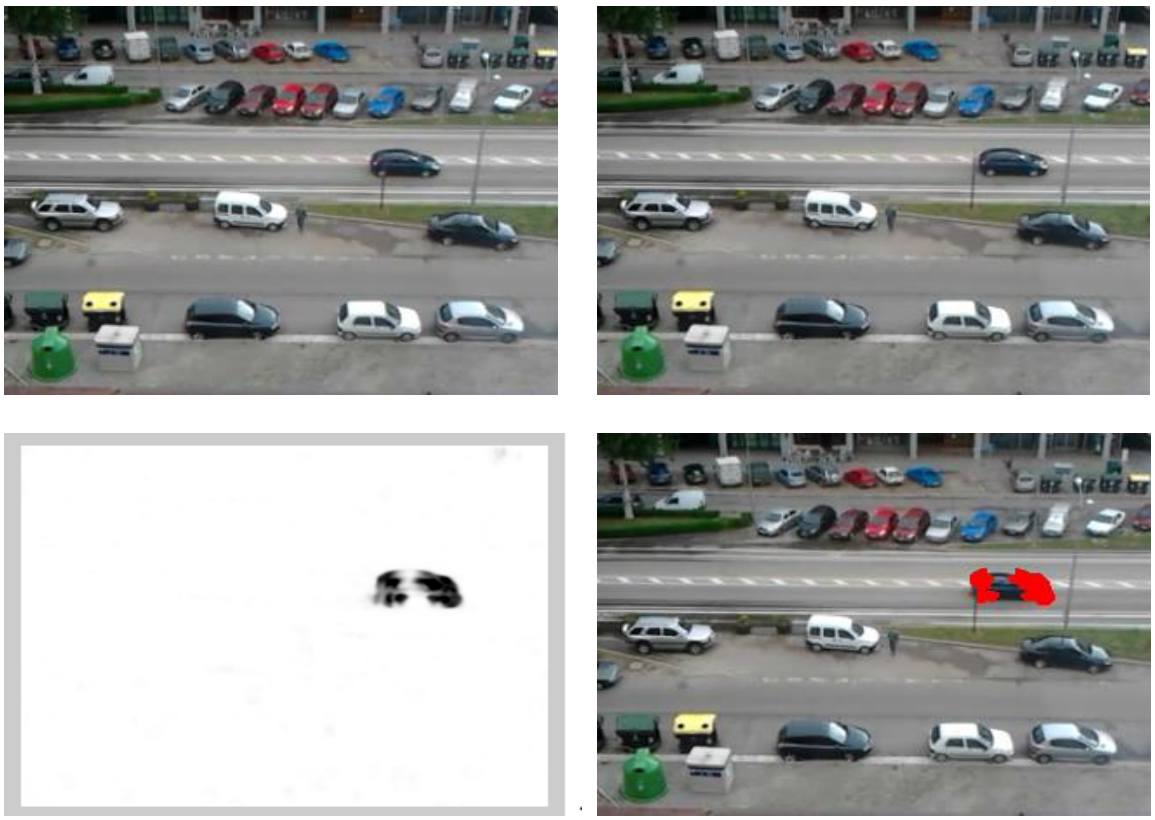


Figura A.12. División de objetos

En las imágenes anteriores, los píxeles oscuros del mapa SSIM corresponden con los extremos delantero y trasero del coche, considerando la parte central como si no se estuviera moviendo, ya que en las dos imágenes que se comparan esta zona es similar. Este problema puede solucionarse expandiendo y/o rellenando el espacio correspondiente a los coches posteriormente a la obtención del mapa SSIM para que las dos zonas separadas se unan y se puedan considerar como una sola, como se ilustra en la siguiente imagen.

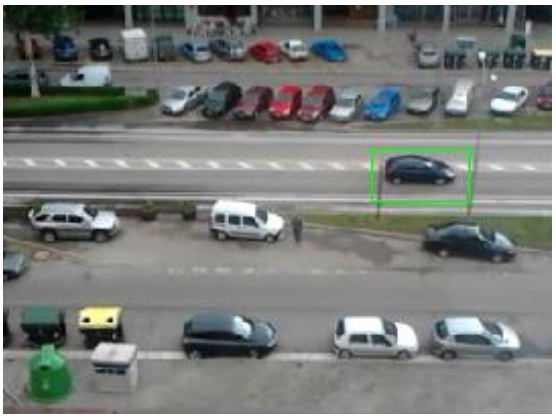


Figura A.13. Solución a la división de objetos

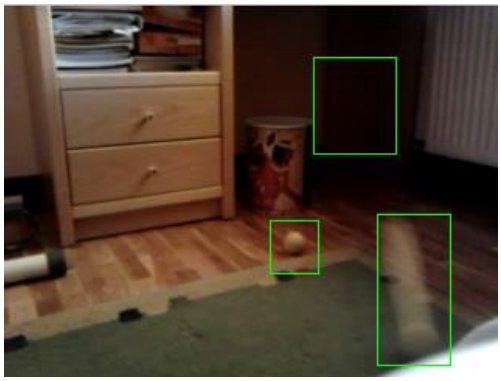
A.7. VELOCIDAD DE LOS OBJETOS

La velocidad a la que se desplazan los objetos tiene influencia en la percepción que tenemos del movimiento entre los sucesivos fotogramas. Hay que tener en cuenta que para confeccionar el mapa SSIM se están comparando dos fotogramas separados tres posiciones. En algunas ocasiones, si la velocidad del objeto es demasiado elevada, la diferencia entre las dos imágenes es mayor de lo esperada. En estos casos el resultado que obtenemos al realizar la detección de objetos puede alejarse de la realidad. A continuación se muestra un caso extremo.

En la figura siguiente A.14 tenemos una situación con dos objetos móviles con distintas velocidades de desplazamiento. La pelota situada en el centro de la escena tiene una velocidad que podemos considerar normal y en (c) se produce una detección correcta. Sin embargo, la pelota a la derecha tiene una velocidad mucho mayor y en los fotogramas que se comparan, (a) y (b), se comprueba que recorre mucha distancia. Esto se traduce en (c) como la detección de dos zonas de movimiento en lugar de detectar un único objeto. También aparece la “estela” del movimiento de la pelota, que se produce al capturar la imagen por la cámara, y que provoca que el tamaño del objeto detectado sea mayor que sus dimensiones reales.

(a) Instante K

(b) Instante K+1



(c) Detección

Figura A.14. Detección de un objeto con velocidad demasiado alta

A.8. MOVIMIENTO DE LA CÁMARA

Hasta ahora hemos considerado que la grabación de los vídeos se realizaba de manera estática, con la cámara totalmente inmóvil. Sin embargo, pueden darse situaciones en las que la cámara sufra pequeñas oscilaciones, ya sea por el efecto del viento si está situada en exteriores, por vibraciones del soporte al que se sujeta, etc. A continuación vamos a comprobar qué efectos tienen pequeños movimientos de la cámara sobre la detección de movimiento.

Al comparar dos imágenes ligeramente desplazadas una respecto a otra, el resultado obtenido es que el mapa SSIM lo interpreta como un movimiento de los contornos de la escena. Un ejemplo de esta situación puede verse en la siguiente figura, donde la imagen de la izquierda ha sufrido una oscilación más suave y la imagen de la derecha un movimiento más brusco.

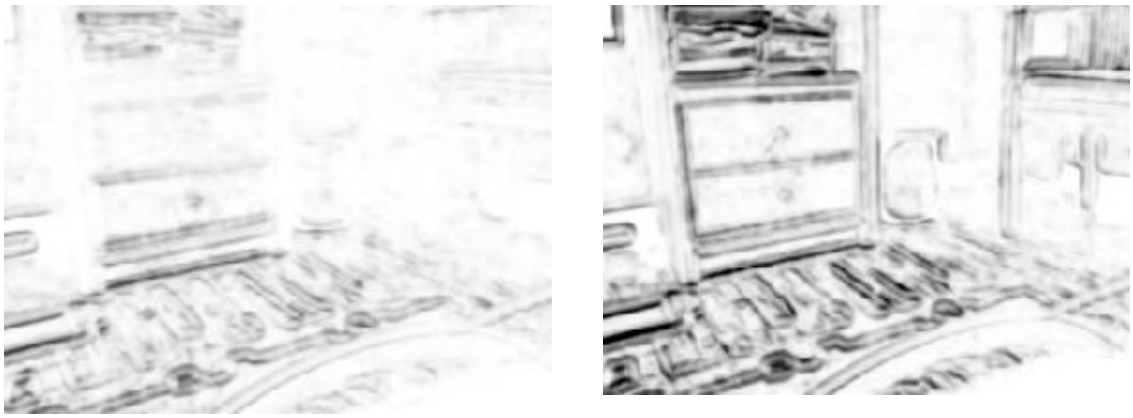


Figura A.15. Mapa SSIM con movimiento de cámara

El problema que esto nos genera en la detección de movimiento es la posibilidad de que estas alteraciones enmascaren sobre el mapa SSIM el movimiento real de objetos. Dependiendo de la intensidad de la oscilación de la cámara podremos realizar una correcta detección o no.

Figura A.16. Detección satisfactoria con movimiento de cámara

En este caso, puede separarse sin problemas los píxeles del objeto en movimiento de los provocados por una ligera oscilación, utilizando un umbral de 0.5. En otras ocasiones no va a ser posible elegir un umbral en el intervalo 0.5 – 0.6, el cual se eligió como el más adecuado en apartados anteriores, para evitar las distorsiones, lo que nos provocará falsas detecciones.

En la figura 5.22 la cámara sufre un movimiento de mayor envergadura. Con el umbral situado en 0.5 vemos que el movimiento del objeto apenas puede distinguirse de las perturbaciones producidas por las oscilaciones. En este caso no es posible realizar una buena detección y el sistema funcionaría de manera errónea.

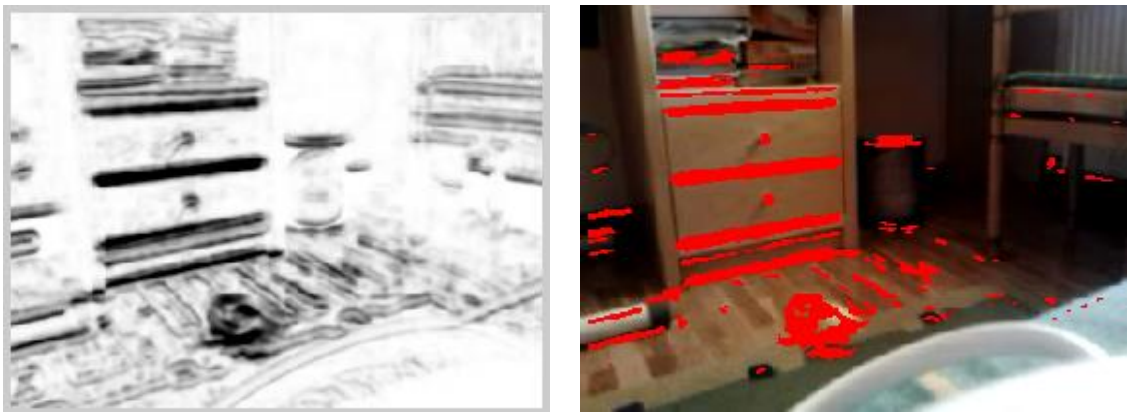


Figura A.17. Detección fallida con movimiento de cámara

A.9. ILUMINACIÓN

Un aspecto importante en detección de movimiento es el efecto de cambios de iluminación en la escena, ya que modificaciones sustanciales en este aspecto pueden provocar falsas detecciones.

Para estudiar el efecto de la iluminación, se consideran escenarios de interior donde se hace variar la intensidad de la luz, oscureciendo e iluminando la estancia. Según las pruebas que hemos realizado parece que los cambios progresivos en la iluminación de la escena no generan grandes alteraciones en el índice SSIM, salvo en puntuales circunstancias; por lo que consideramos que de forma inicial no es necesario incluir ningún tipo de corrección sobre la iluminación. Sin embargo, en el caso de algún tratamiento futuro podría considerarse añadir alguna mejora al respecto.

También se ha probado la detección de movimiento en diferentes condiciones de iluminación dentro una habitación. Para evaluar la respuesta del programa se han utilizado seis vídeos en los que se lanzaban once pelotas y el resultado obtenido ha sido:

- En los vídeos que tenían dos regiones de iluminación, una con más luz y otra de más sombra, todas las pelotas se han detectado en la zona de luz pero no ha sido posible en la de oscuridad.
- Los otros vídeos tenían una distribución de luz más uniforme, aunque también oscura y se ha detectado movimiento en cinco de seis casos de forma más o menos regular durante el movimiento de la pelota, especialmente al principio cuando se desplazaban a mayor velocidad y el movimiento entre fotogramas era más apreciable.

A.10. CONDICIONES NOCTURNAS

Como una prueba extraordinaria, se ha estudiado el comportamiento de la aplicación cuando se aplica a la detección de vehículos en exteriores por la noche. En este tipo de entornos las

características más frecuentes son la presencia de zonas oscuras y otras iluminadas por los focos de los coches o por farolas de la calle. Percibimos que todos los coches pueden ser detectados, en parte gracias a los focos cuando no hay otro tipo de iluminación. Sin embargo, estas luces móviles también provocan alteraciones en la escena, ya sea por el propio haz de luz o porque aparezcan reflejos y brillos puntuales en elementos estacionados en la calle, lo que aumenta el número de falsas detecciones. Esto es especialmente notorio si nos guiamos por la comparación del fotograma actual con el primer fotograma donde se detectó movimiento. También es posible que no se produzca la detección del volumen total del vehículo, sino que sólo aparezca cierta zona o que esté dividido en varios trozos.

En las siguientes imágenes puede comprobarse que el algoritmo detecta la presencia correctamente de dos vehículos, un coche y un autobús, pero comete algunos fallos: Aparecen reflejos de las luces en los coches estacionados, los vehículos se muestran divididos, etc.

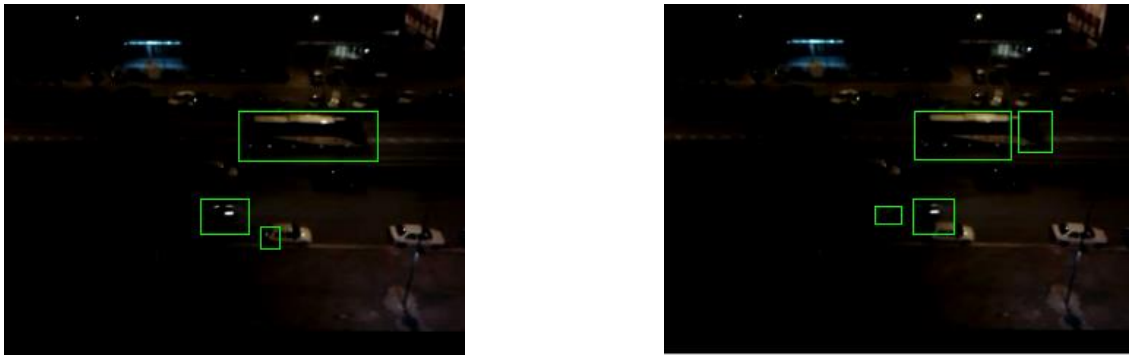


Figura A.18. Errores de detección en condiciones nocturnas

Una situación parecida la podemos encontrar en habitaciones con poca iluminación. Si en una habitación con poca luz tenemos, por ejemplo, un televisor; los cambios de escena o de secuencia en la pantalla pueden provocar variaciones fuertes e instantáneas de la iluminación. Estas situaciones generan serias alteraciones en el mapa SSIM.

A.11. RESULTADOS

En el siguiente apartado vamos a presentar de manera global los resultados obtenidos al realizar los experimentos que se han comentado anteriormente. Los resultados se agruparán según las características que presenten los vídeos y los comportamientos que se pretendan evaluar.

Con el fin de establecer una comparación entre un conjunto de datos de prueba podemos crear un *modelo de comparación*, el cual consiste en una función que divida y clasifique las distintas situaciones que se obtienen después de llevar a cabo unos experimentos entre dos clases bien diferenciadas.

En nuestro caso podemos definir dos clases diferentes: vídeos con movimiento y vídeos sin movimiento. De esta manera el clasificador se convierte en un sistema con dos entradas y dos salidas, el cual se representa mediante la llamada matriz de confusión o tabla de contingencia.

Según la tabla tenemos dos posibles valores en la realidad: Hay movimiento (*p*) o no hay movimiento (*n*). El sistema hace una predicción de la cual se obtienen dos circunstancias: se detecta movimiento (*p´*) o no se detecta movimiento (*n´*). En la figura A.19 puede verse la representación gráfica de una tabla de contingencia.

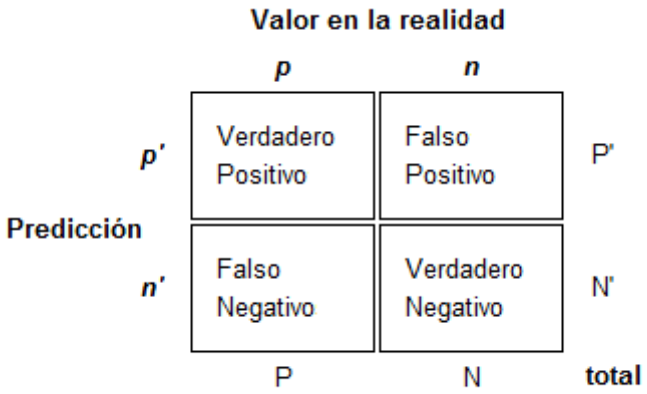


Figura A.19. Tabla de contingencia

De esta tabla se derivan cuatro situaciones diferentes:

- Verdadero Positivo (VP): Si existe movimiento y éste es detectado.
- Falso Positivo (FP): Si no hay ningún movimiento pero el algoritmo indica presencia.
- Falso Negativo (FN): Si hay un objeto en movimiento pero no se detecta.
- Verdadero Negativo (VN): Si no hay movimiento y no se detecta nada.

A partir de estos cuatro términos se pueden definir otros como:

- Razón de Verdaderos Positivos (VPR) o Sensibilidad: $VPR = \frac{VP}{VP+FN}$
- Razón de Falsos Negativos (FPR) o 1-Especificidad: $FPR = \frac{FP}{VN+FP}$

Pasando a continuación a recoger los resultados, en primer lugar se han relazado pruebas del comportamiento del sistema sin que se produzcan movimientos, únicamente evaluando si con un umbral de 0.5 el sistema genera positivos o no.

Se ha realizado una prueba en la cual la cámara se mueve de manera circular y cuando sufre pequeñas vibraciones. Se pretende comprobar cómo afecta este movimiento a la respuesta del

sistema. Se estudia el número de fotogramas que presentan alteraciones y los que se presentan inmóviles:

Vídeo	Nº de fotogramas	Bien	Mal
1	83	26 (31.32%)	57 (68.67%)
2	78	20 (25.64%)	58 (74.35%)
Total	161	46 (28.57%)	115 (71.43%)

Tabla A.1. Efecto de movimientos de cámara circulares

Vídeo	Nº de fotogramas	Bien	Mal
1	38	35 (92.10%)	3 (7.89%)
2	54	35 (64.81%)	19 (35.18%)
Total	92	70 (76.08%)	22 (23.91%)

Tabla A.2. Efecto de vibraciones de cámara

Para comprobar el efecto de los cambios de iluminación se han realizado varios experimentos en una habitación cambiando progresiva y suavemente la iluminación de la sala mediante un interruptor regulable.

Vídeo	Nº de fotogramas	Bien	Mal
1	82	81 (98.78%)	1 (1.39%)
2	70	69 (98.57%)	1 (1.43%)
3	64	61 (95.31%)	3 (4.68%)
Total	216	211 (97.68%)	5 (2.21%)

Tabla A.3. Efecto de cambios de intensidad en la iluminación de una habitación

En las siguientes tablas vamos a caracterizar cada vídeo de movimiento en función de los parámetros: verdadero positivo (VP), falso positivo (FP), verdadero negativo (VN) y falso negativo (FN). Después, a partir de estos valores vamos a calcular el porcentaje de tiempo del vídeo en el que el sistema se comporta de manera correcta, es decir, detecta un movimiento cuando lo hay y no lo detecta cuando no lo hay, y el porcentaje de detecciones en los fotogramas que presentan un movimiento frente al porcentaje de falsos negativos. También se calcula la relación entre los positivos verdaderos y el número total de fotogramas en los que ocurre algún movimiento, que se obtiene como:

Precisión/Sensibilidad: $\frac{VP}{VP+FN}$

Por último se incluyen comentarios y los errores más frecuentes y característicos que se producen en cada vídeo.

Tipo de vídeo	Nº frames	VP	FP	VN	FN	Acierto	Error	Sensibilidad
1. Interior – bola Poca luz	58	0.3275	0.0344	0.6034	0.0344	93.09%	6.88%	**90.47%**
2. Interior – bola Poca luz	74	0.3978	0	0.4054	0.2027	79.72%	20.27%	**65.90%**
Total	132	0.3636	0.0151	0.4924	0.128	85.6%	14.4%	**73.39%**
	Errores	- La oscuridad aumenta del primero al segundo. - Umbral 0.5 se considera que se ha detectado un movimiento cuando aparece reflejado en la comparación actual, en la relativa o en ambas.						
1-a. Interior bola	124	0.4596	0	0.4838	0.056	94.34%	5.65%	**89.13%**
1-b. Interior bola	124	0.4838	0	0.4838	0.0323	96.76%	3.23%	**93.74%**
2. Interior bola	97	0.5361	0	0.4123	0.0515	94.83%	5.15%	**91.23%**
Total [1-b ; 2]	221	0.5067	0	0.4524	0.0408	95.91%	4.08%	**92.25%**
	Errores	- El primer caso se refiere a la detección actual y el segundo a la doble detección. - Cuando la pelota está cerca de detenerse, el algoritmo con el umbral 0.5 no es capaz de distinguir desplazamientos tan débiles. - En ocasiones, si el color o la textura del objeto es muy parecida a la del fondo en el que se encuentra en cierto momento, el algoritmo tiene dificultad.						
1. Interior bola vibración	82	0.5000	0.0244	0.2683	0.2073	76.83%	23.17%	**70.69%**
2. Interior bola vibración	61	0.2786	0.3114	0.3278	0.0819	60.64%	39.33%	**77.27%**
Total	143	0.4056	0.1468	0.2937	0.1538	69.93%	30.06%	**72.50%**
	Errores	- En el segundo caso la vibración es más fuerte que la primera, por lo que aumentan las falsas detecciones. - Sin embargo, en el primero los objetos se mueven más despacio, en ocasiones no se detectan y aparecen más falsos negativos, por lo que la sensibilidad disminuye.						

Tabla A.4. Vídeos en interiores I

Tipo de vídeo	Nº frames	VP	FP	VN	FN	Acierto	Error	Sensibilidad
1. Interior-personas	26	0.5	0.2307	0.2307	0.0384	73.07%	26.92%	**92.85%**
2. Interior-personas	39	0.1795	0.1025	0.7179	0	89.74%	10.25%	**100%**
Total	65	0.3076	0.1538	0.5231	0.0153	83.08%	16.91%	**95.23%**
	Errores	- El porcentaje de detección (sensibilidad) es muy alto, y sólo falla en las entradas o salidas de personas de la escena cuando una pequeña parte de ellas puede apreciarse. - Los falsos positivos que se producen por las sombras son los más frecuentes dentro del porcentaje de fallos.						
1. Interior calibrado &pelota	195	0.0205	0.1743	0.8000	0.0051	82.05%	17.94%	**80%**
2 Interior calibrado	230	0	0.2174	0.7826	0	78.26%	21.74%	**-**
Total	425	0.0094	0.1976	0.7905	0.0023	80.00%	20.00%	**80%**
	Errores	- El escenario se corresponde con una habitación con baja iluminación y un televisor donde se aplica el calibrado - Los errores se deben a falsos positivos provocados por cambios bruscos de iluminación. - La segunda comparación no es útil porque se añaden continuamente errores.						

Tabla A.5. Vídeos en interiores II

Tipo de vídeo	Nº frames	VP	FP	VN	FN	Acierto	Error	Sensibilidad
1(a). Calle– Coche- Día	137	0.7737	0.073	0	0.2189	77.37%	22.62%	**77.94%**
1(b).Calle– Coche- Día	137	0.8102	0.0073	0	0.1825	81.02%	18.98%	**81.61%**
2. Calle – Coche- Día	141	0.7943	0.071	0.1134	0.085	90.77%	9.23%	**90.33%**
Total	278	0.8021	0.0072	0.0575	0.1331	85.95%	14.04%	**85.77%**
	Errores	- El primer caso se refiere a la detección actual y el segundo a la doble detección. Si se emplea la doble detección, considerando el mapa SSIM actual y el relativo, el porcentaje de acierto aumenta, ya que es capaz de seguir objetos que dejan de detectarse un fotograma.						
1(a). Calle – Coche – Tarde	120	0.6926	0	0	0.3083	69.16%	30.83%	**69.16%**
1(b). Calle – Coche – Tarde	120	0.7608	0.1014	0	0.1376	76.08%	23.90%	**84.67%**
2 Calle – Coche – Tarde	113	0.7079	0.1151	0	0.1769	70.79%	29.20%	**80%**
	Errores	- Este escenario tiene peor visibilidad y está tomado desde más lejos, lo que puede explicar las menores prestaciones que en el caso anterior. - De nuevo se comprueba que la segunda comprobación, 1(b), aumenta la efectividad del sistema, aunque también se incremente el número de falsos positivos. - Los errores se deben a la división de los objetivos en varias partes y a no detecciones momentáneas.						
Calle – Coche Noche	142	0.5986	0.2887	0.1126	0	71.12%	28.87%	**100%**
	Errores	- Al estar parcialmente iluminado los objetos vehículos más grandes están divididos en varias partes, por lo que parece que hay mayor número de objetos pequeños, tenemos falsos positivos. - La iluminación de faros causa brillos y reflejos puntuales que generan falsas detecciones (FP). - Por el contrario, siempre detecta la presencia de un vehículo.						

Tabla A.6. Vídeos en exteriores I

Tipo de vídeo	Nº frames	VP	FP	VN	FN	Acierto	Error	Sensibilidad
1(a). Calle-Persona	118	0.6949	0	0.1186	0.1864	81.35%	18.64%	**78.85%**
1(b). Calle-Persona	118	0.3474	0	0.1335	0.5169	48.09%	51.69%	**40.19%**
	Errores	- Cámara a gran altura y objetivos pequeños. - El primer caso se refiere a la doble detección y el segundo a la detección actual. - Fallos al confundir la textura de la persona con la calle						
1. Exterior-personas	56	0.6607	0	0.2678	0.071	92.85%	7.14%	**90.24%**
2. Exterior – personas	83	0.7952	0.0963	0.0361	0.0723	83.13%	16.86%	**91.66%**
Total	139	0.6376	0.0412	0.2614	0.0596	88.9%	10.1%	**91.45%**
	Errores	- Cuando las personas o los objetos están muy juntos el sistema los considera como uno solo mediante un mismo rectángulo. Se consideran que son falsos negativos - Aparecen reflejos y sombras que se caracterizan como falsos positivos.						
1. Exterior-personas	110	0.7454	0.1091	0.1363	0.0091	88.17%	11.82%	**98.79%**
2. Exterior – personas	83	0.7952	0.0963	0.0361	0.0723	83.13%	16.86%	**91.66%**
3. Exterior – personas	81	0.8642	0.0741	0.0123	0.0493	87.65%	12.34%	**94.59%**
Total	274	0.7956	0.0948	0.0693	0.0401	86.49%	13.49%	**95.19%**
	Errores	- El porcentaje de detección (sensibilidad) es muy alto, y sólo falla en las entradas o salidas de personas de la escena cuando una pequeña parte de ellas puede apreciarse. - Los falsos positivos que se producen por las sombras son los más frecuentes dentro del porcentaje de fallos.						

Tabla A.7. Vídeos en exteriores II

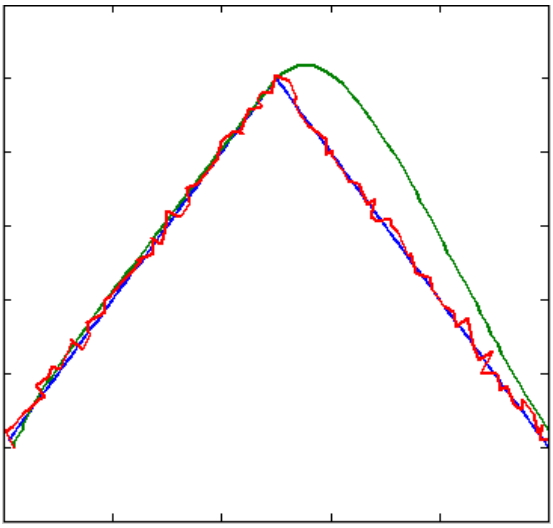
ANEXO B

ANÁLISIS DE LOS PARÁMETROS DE LOS FILTROS

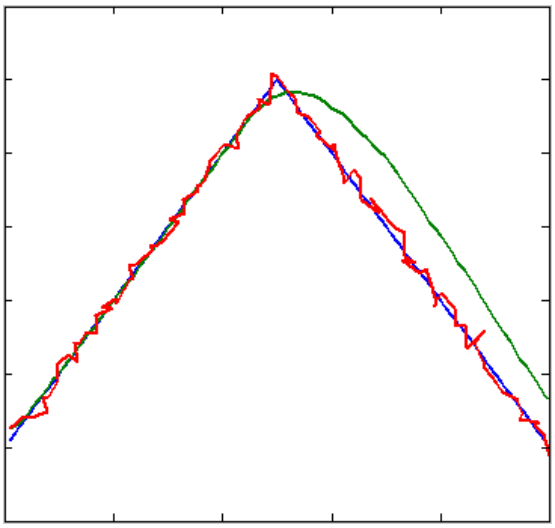
En este anexo se recogen la totalidad de las pruebas realizadas en referencia al análisis de los parámetros de los filtros de seguimiento empleados en el sistema: alfa-beta y Kalman. El objetivo de las pruebas consiste en conocer cómo evoluciona el comportamiento del filtro en cuestión al ir incrementando gradualmente el valor de sus parámetros en un intervalo 0-1, para una trayectoria rectilínea con un cambio brusco de dirección con y sin oclusión. En el caso del filtro de Kalman el parámetro modificado es el intervalo de tiempo de actualización, ya que la variable ganancia de Kalman *K* no permite ser ajustada. Para el filtro alfa-beta, se incrementa el parámetro *alfa* mientras que el parámetro *beta* varía en función de las fórmulas siguientes:

$$\beta_1 = 2(2-\alpha)4\sqrt{1-\alpha} \qquad \beta_2 = 0.8\left(\frac{2-\alpha^2-2\sqrt{1-\alpha^2}}{\alpha^2}\right)$$

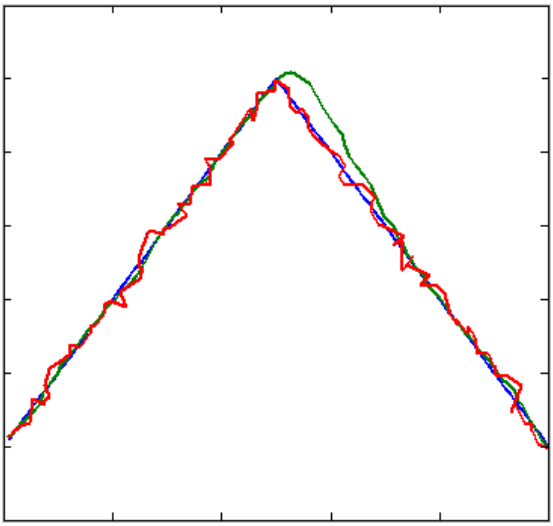
En todas la representaciones realizadas, la línea azul se corresponde con la trayectoria real sin ruido, la línea roja es la trayectoria medida con ruido (en ocasiones desaparece debido a la oclusión) y la línea verde se corresponde con la trayectoria producida por el filtro correspondiente. Debajo de cada gráfica se indica el valor de los parámetros empleado. Los resultados se agrupan en las cuatro tablas que se presentan a continuación, para el filtro alfa beta con y sin oclusión, y para el filtro de Kalman con y sin oclusión.



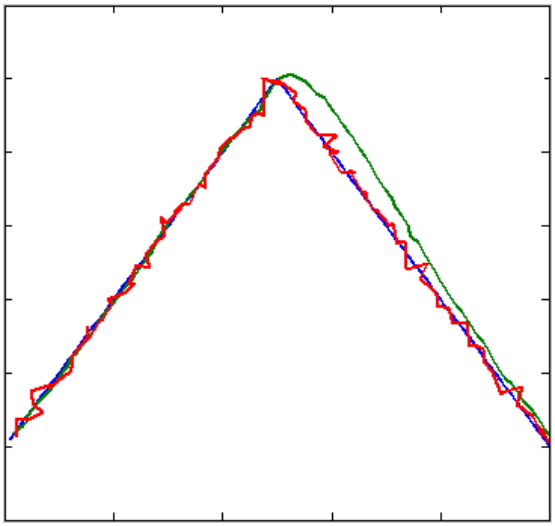
alfa=0.1 beta=0.0053



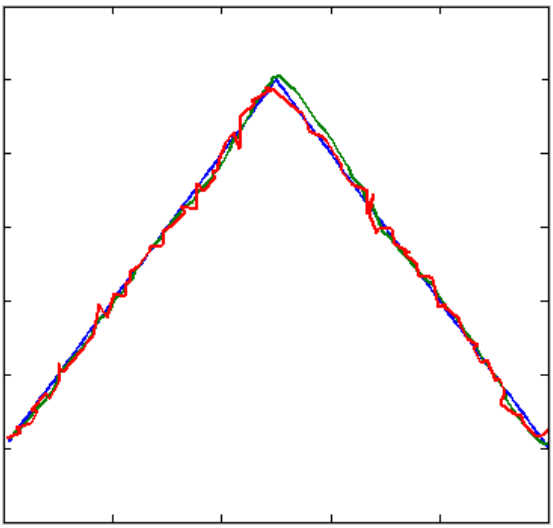
alfa=0.1 beta=0.002



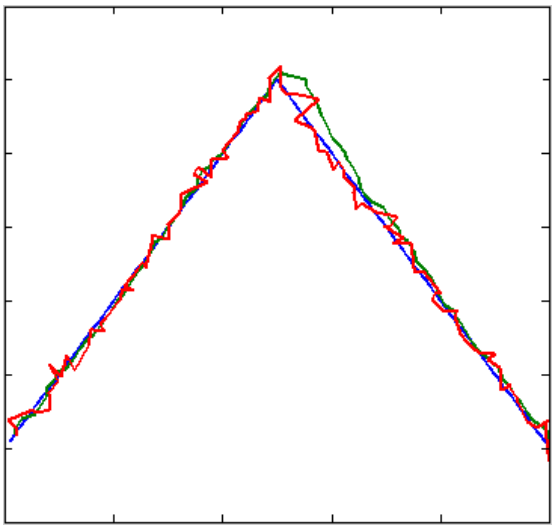
alfa= 0.2 beta= 0.0222



alfa= 0.2 beta= 0.0082



alfa= 0.3 beta= 0.0529



alfa= 0.3 beta= 0.0189

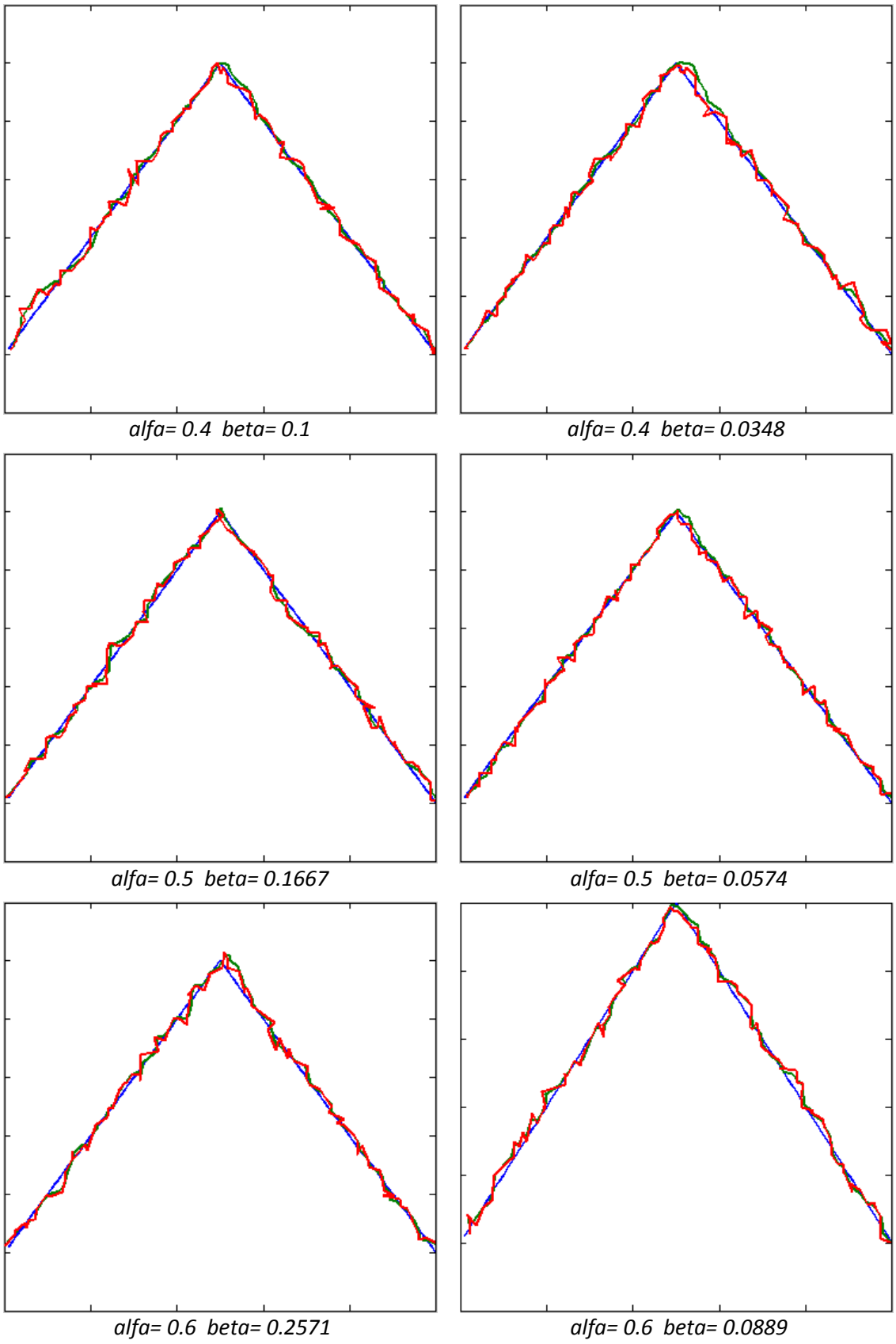
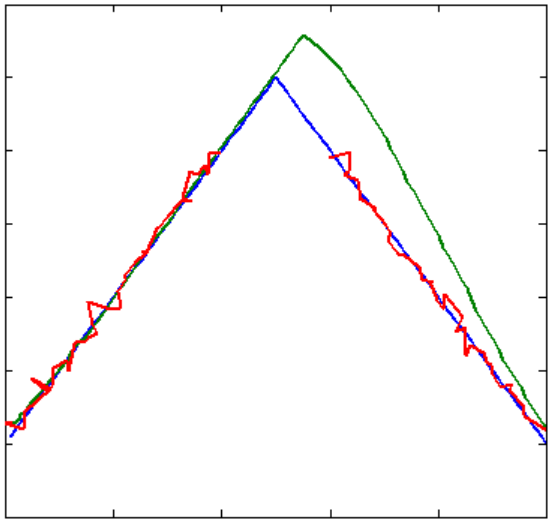
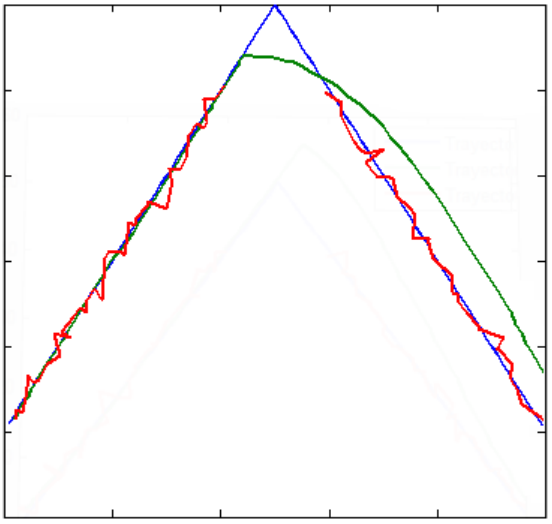


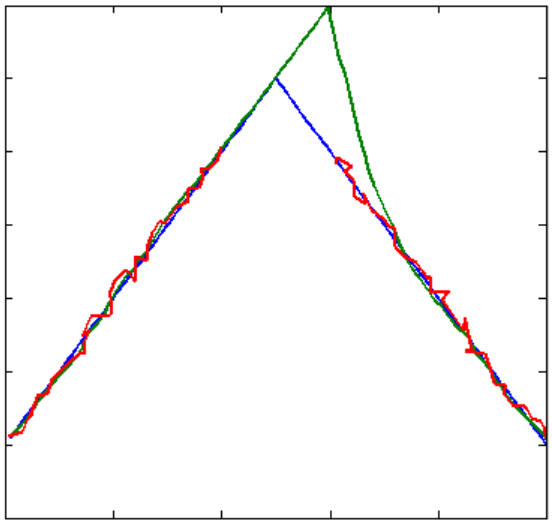
Figura B.1. Filtro alfa beta (sin oclusión)



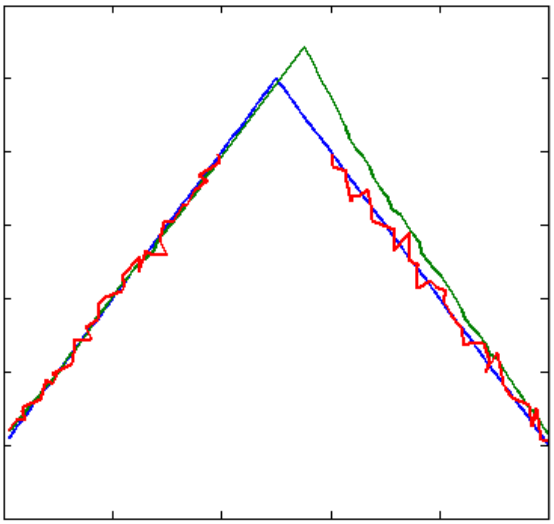
alfa=0.1 beta=0.0053



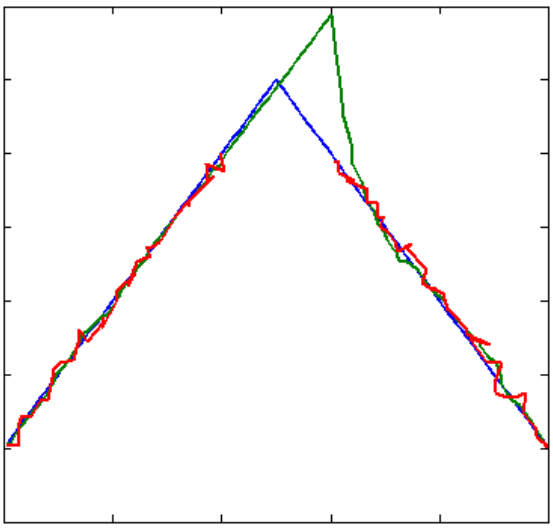
alfa=0.1 beta=0.002



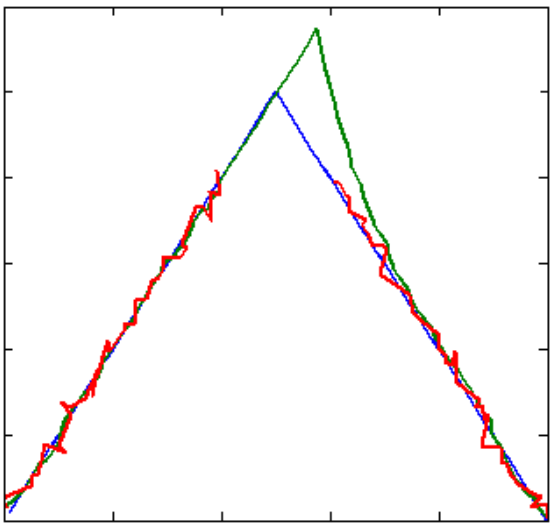
alfa= 0.2 beta= 0.0222



alfa= 0.2 beta= 0.0082



alfa= 0.3 beta= 0.0529



alfa= 0.3 beta= 0.0189

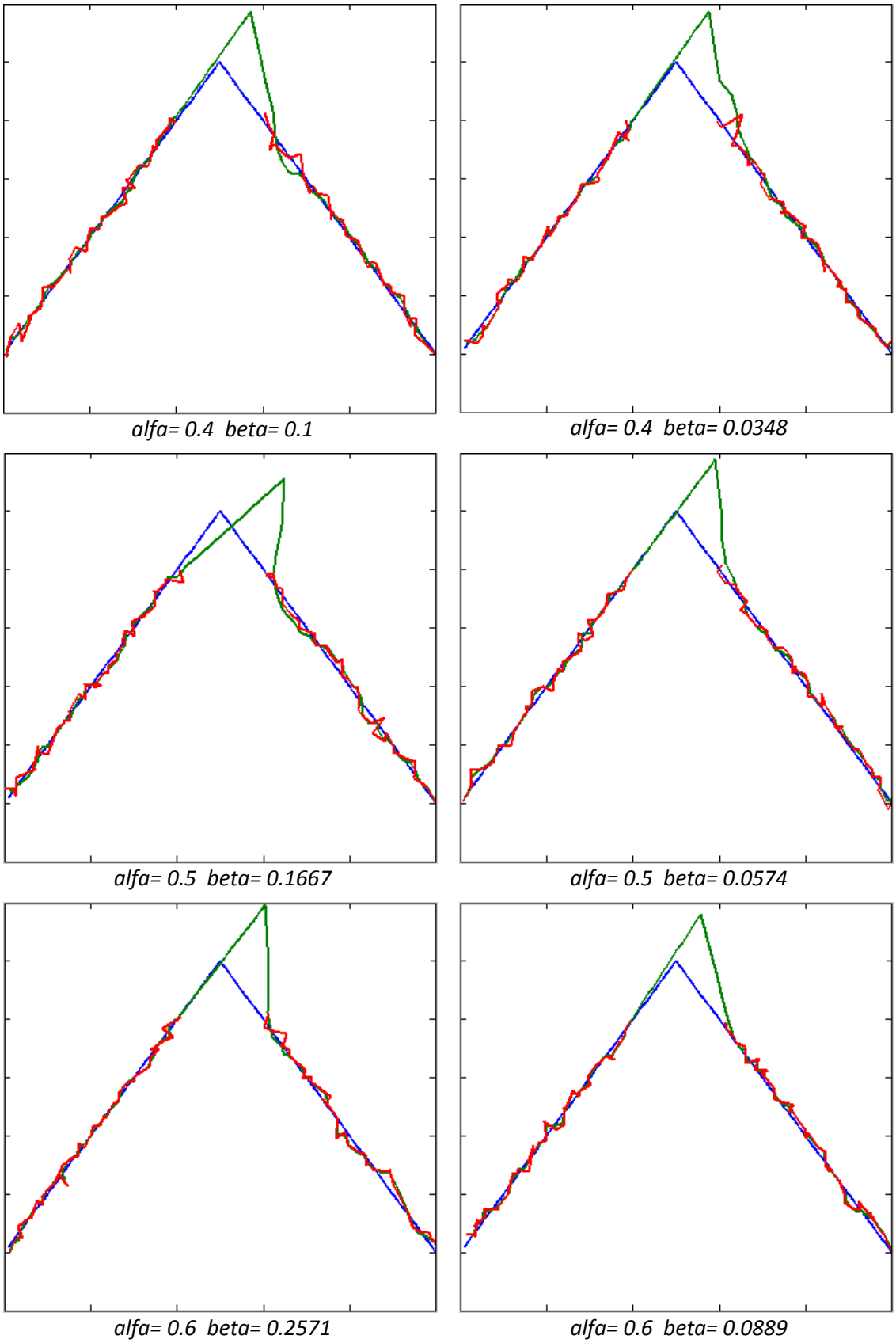
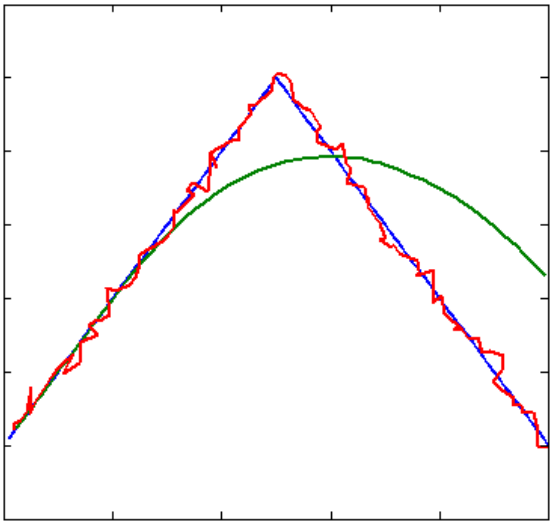
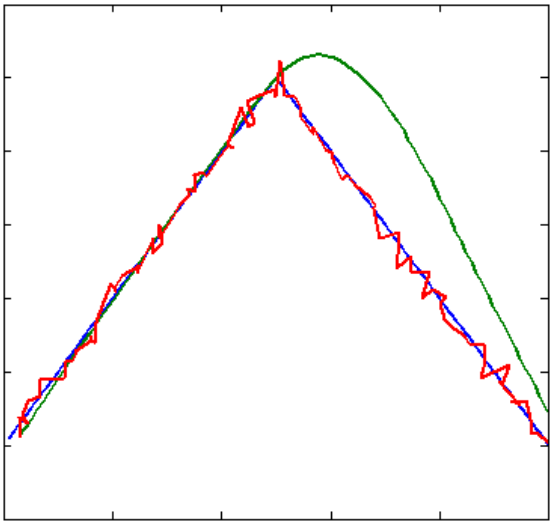


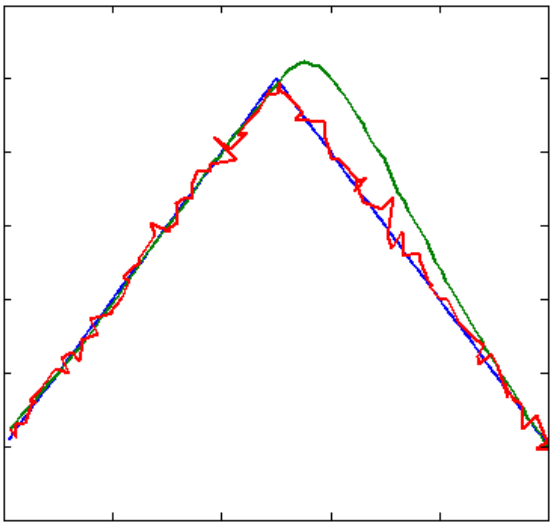
Figura B.2. Filtro alfa beta (con oclusión)



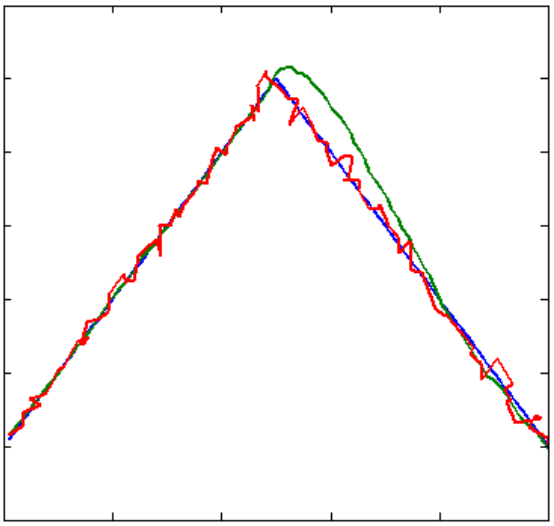
dt=0.1



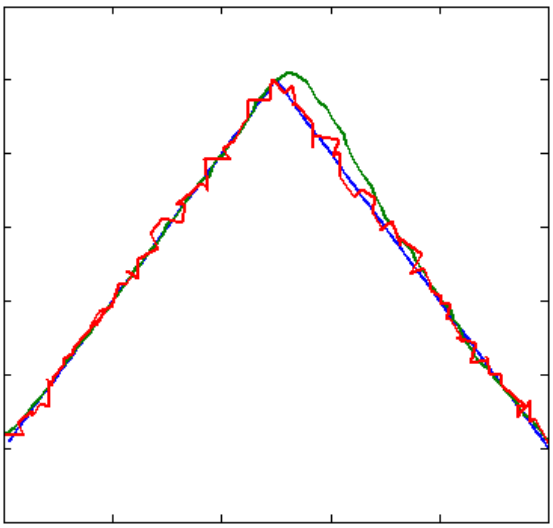
dt=0.2



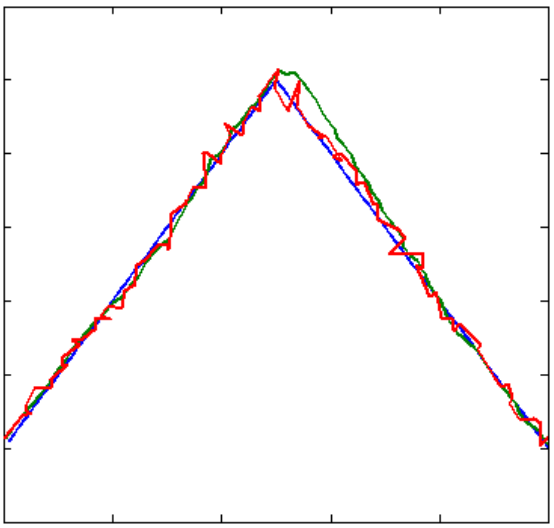
dt=0.3



dt=0.4



dt=0.5



dt=0.6

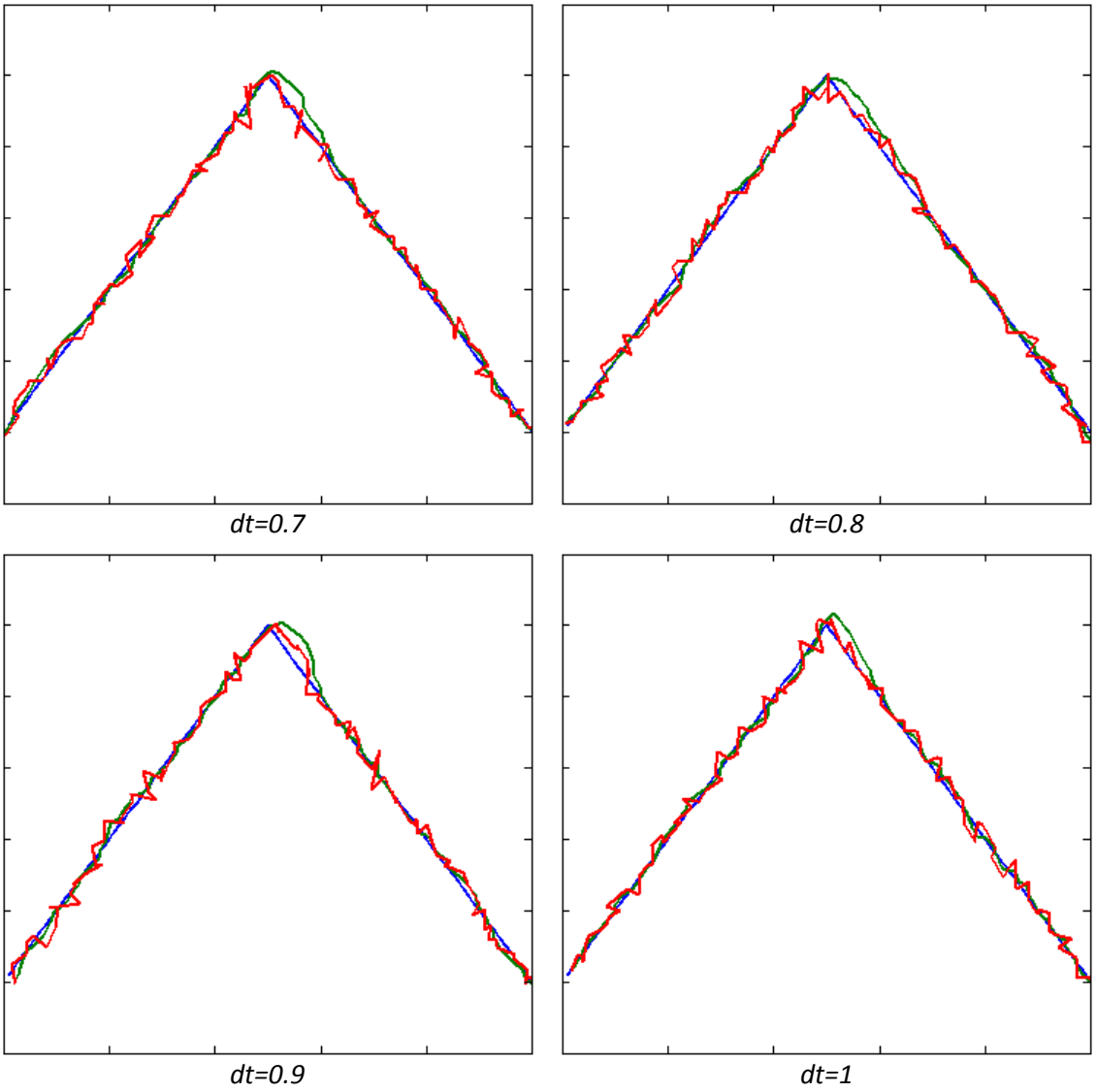
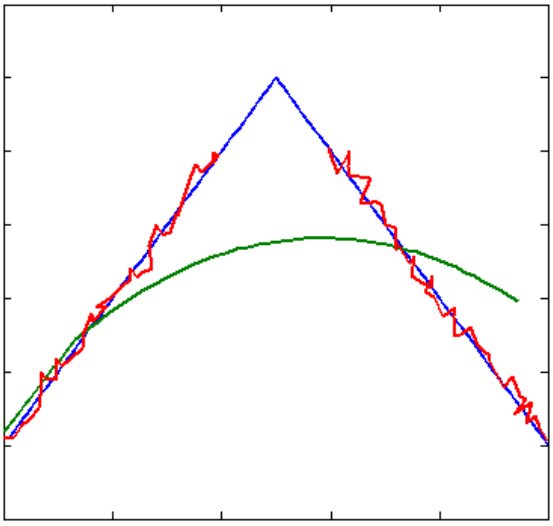
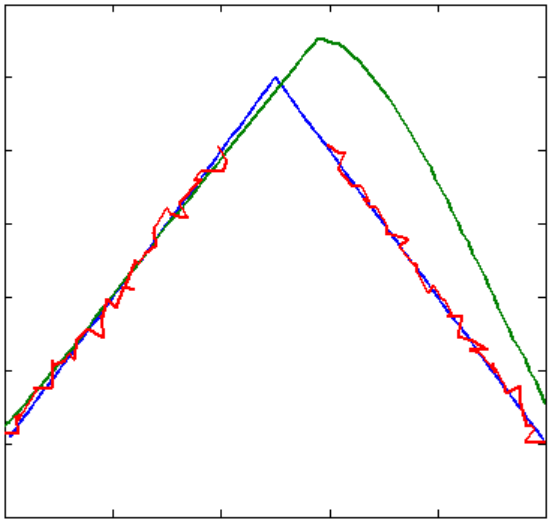


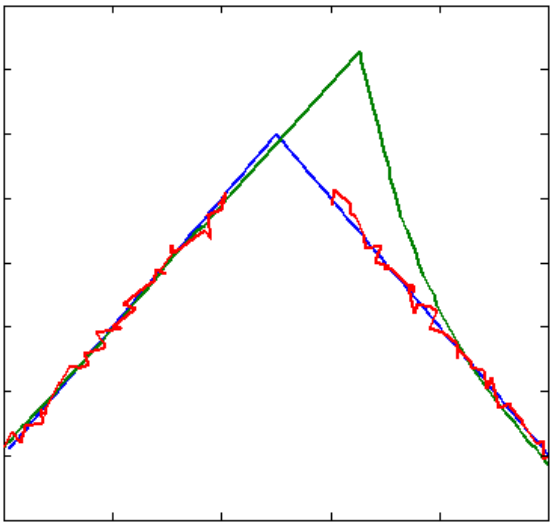
Figura B.3. Filtro de Kalman (sin oclusión)



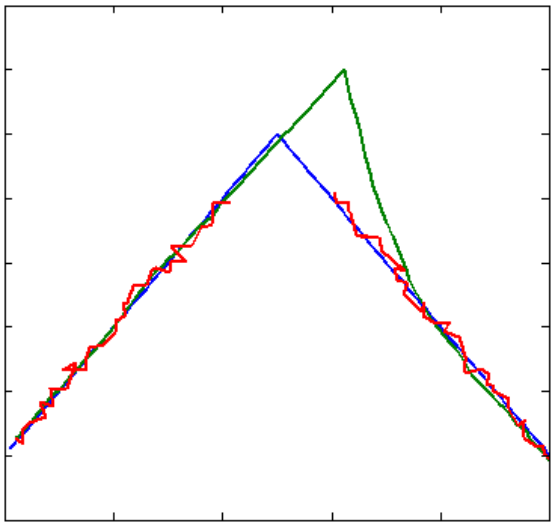
dt=0.1



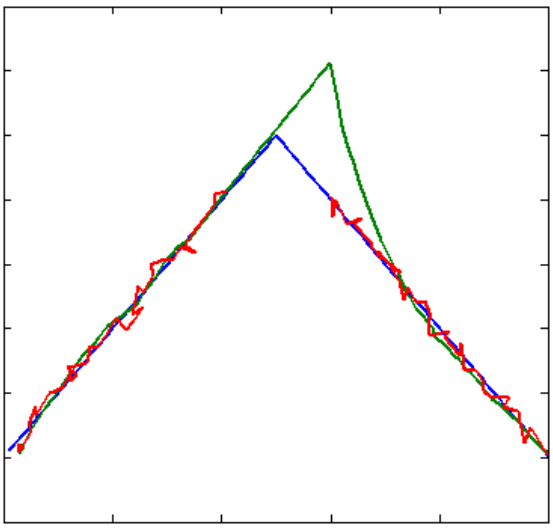
dt=0.2



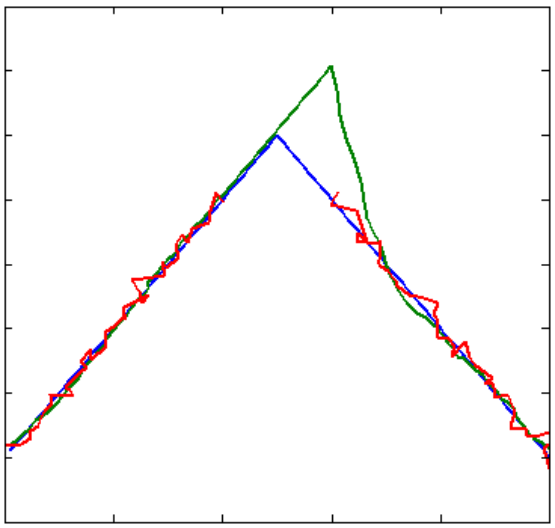
dt=0.3



dt=0.4



dt=0.5



dt=0.6

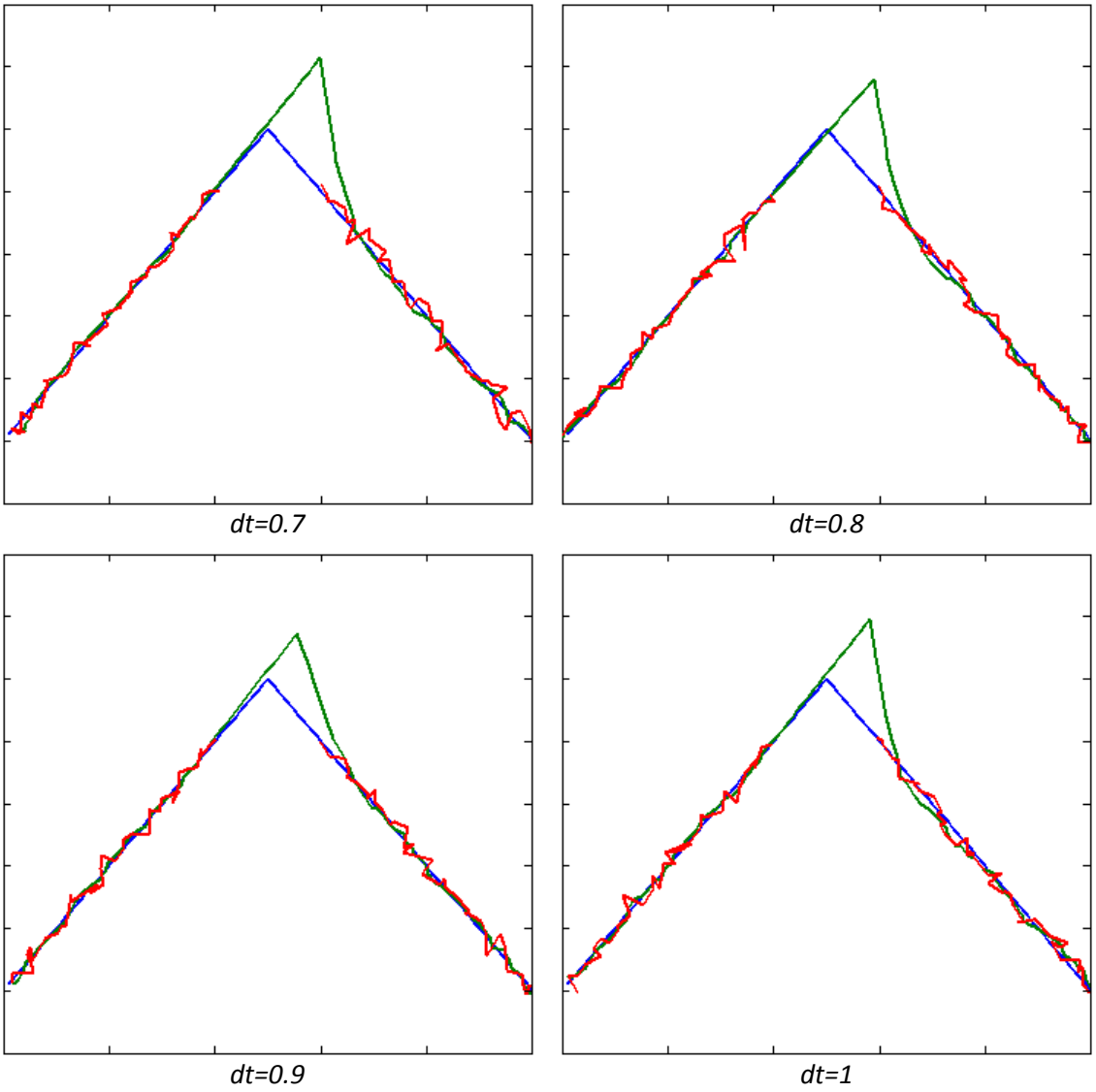


Figura B.4. Filtro de Kalman (con oclusión)

ANEXO C

CÓDIGOS EN MATLAB

A continuación se incluyen los códigos de Matlab utilizados para generar la versión final del sistema de detección y seguimiento de objetos móviles.

- **Función Principal**

```
function
Principal(nombre_video,opcion_filtrado,visualizar,frame_inicio)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%  VERSION FINAL  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Este código corresponde con la fase final del proceso del proyecto:
%Comprobar que el algoritmo de detección SSIM y de seguimiento (Alfa-
%Beta o Kalman) funcionan de forma conjunta sobre el mismo video.
%
%En la versión final del sistema se considera que una "cámara"
%ficticia se está moviendo siguiendo al objetivo, de forma que el
%frame de video completo es el área que la cámara puede barrer (área
%de vigilancia) y la detección solo se realiza dentro de una zona
%determinada de la imagen (ventana de detección).La zona de detección
%se va a centrar sobre la posición del filtro que se estará moviendo
%continuamente, barriendo el área de vigilancia buscando blancos
%móviles o siguiendo a un objetivo detectado. Debido a esta
%circunstancia, las imágenes consecutivas que se comparan están
%desplazadas una respecto a la otra por lo que es necesario
%determinar esa diferencia para aplicar el algoritmo SSIM
%correctamente (separar el movimiento de la cámara del movimiento de
%los objetos).
%
%El código se divide en dos partes: la inicialización de las variables
%y parámetros de los filtros, y el bucle de procesado. El bucle se
%divide en varias fases: 1)Adquisición de los frames de video y su
%manipulación para prepararlos para las siguientes fases. 2)Detección
%del movimiento mediante el algoritmo SSIM. 3)Seguimiento de los
```

```
%objetos en movimiento mediante los filtros, y posicionamiento de la
%ventana de detección. 4)Visualización de los resultados.
%
%
%Funciones dependientes:
%- ssim
%- im_rgb
%- punto
%- Funcion_FiltroAlfaBeta
%- Funcion_FiltroKalman_mod1
%- correccion_desplazamiento
%
%
% Parametros de entrada (ejemplo de funcionamiento):
% Nombre del video
% ->nombre_video='vina3.avi';
% Tipo de filtro: filtro alfa-beta=1, filtro de Kalman=2
% ->opcion_filtrado=1;
% Imagen a representar: posicion filtro=1, mapa SSIM=2, Cruz de
% posicion detectada=3, imagen real=4
% ->visualizar=2;
% Instante de inicio
% ->frame_inicio=45;
%
%=========================================================================

close all

if (nargin==0)
    nombre_video='vina3.avi';
    frame_inicio=45;
    opcion_filtrado=1;
    visualizar=2;
end
if (nargin==1)
    frame_inicio=45;
    opcion_filtrado=1;
    visualizar=2;
end
if (nargin==2)
    frame_inicio=45;
    visualizar=2;
end
if (nargin==3)
    frame_inicio=45;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%% INICIALIZACION %%%%%%%%%%%%%%%%%%%%%%%%%%%

%La variable contador indica si el número de fotogramas consecutivos
%en los que se detecta movimiento.
contador=0;

%Se carga el video
vid=VideoReader(nombre_video);
```

```
%Se obtiene el tamaño de las imágenes con las que se va a trabajar,
%para establecer los límites de exploración y los márgenes para
%corregir la posición del filtro de seguimiento.
img_prueba=read(vid,2);
[sizeY,sizeX,~]=size(img_prueba);
clear img_prueba

%Se define el tamaño de la ventana de detección. Deben ser números
%pares
ventX=110;
ventY=110;

%Se definen los limites hasta donde debe moverse el filtro: la mitad
%del ancho de la ventana por la izquierda (+1) y 54 por la derecha (-
%1), 15 por arriba y abajo.
limites=[ventX/2+1 ventY/2+1 sizeY-ventY/2-1 sizeX-ventX/2-1];


% Inicializar los valores de posición y velocidad estimado
%Posición de inicio aleatoria
inicioX=randi(limites(4)-limites(1),1)+limites(1);
inicioY=randi(limites(3)-limites(2),1)+limites(2);
xk = [inicioX inicioY];
xk2=xk;
vx_inicio=-100*sign(randn);   % Velocidad inicial para la búsqueda en
el eje x
vk = [vx_inicio 0];
vk2 = vk;
inicio=0;   % Indica si se ha detectado un blanco y si se ha iniciado
todo el proceso
contadorNaN=0;   % Cuenta el número de instantes durante los que se
pierde la información de posición del blanco


%Variables generales para el seguimiento y representación
%- Parámetros auxiliares para la representación
r = 5; % r es el radio del circulo de la representación gráfica
j=0:.01:2*pi; %Realiza el circulo
%- Figura para expandir imágenes binarias
H=[0 0 0 1 0 0 0; 0 1 1 1 1 1 0; 0 1 1 1 1 1 0; 1 1 1 1 1 1 1; ...
   0 1 1 1 1 1 0; 0 1 1 1 1 1 0; 0 0 0 1 0 0 0];
%- Ventana para el SSIM simplificado
window=ones(5)./25;

%-------------------------------------------------------------

if(opcion_filtrado==1)
    fprintf('Seleccionado filtro alfa-beta\n');
    leyenda='Filtro alfa-beta';
    %- Definición de los parámetros alfa y beta
    alfa = 0.6;            % valor de alfa
    betta = 2*(2-alfa)-4*sqrt(1-alfa);      % valor de beta
    dt=0.1;
    %- Formulas para relacionar beta con alfa
    %betta_underdamped= (alfa^2)/(2-alfa);
    %betta_criticaldamping= 0.8*((2-alfa^2-2*sqrt(1-alfa^2))
%/(alfa^2));
```

```
    %betta_opt= 2*(2-alfa)-4*sqrt(1-alfa);
    %-------------------------------------------------------------
elseif(opcion_filtrado==2)
    fprintf('Seleccionado filtro de Kalman\n');
    leyenda='Filtro de Kalman';
    %-------------------------------------------------------------
    %Se inicializan los parámetros para el filtro de Kalman:
    %- Definir las variables principales
    acel = 0; % Aceleración
    dt=1.1;
    vx_inicio=vx_inicio/10;
    vk=vk/10;
    vk2=vk;
    %- Definir las matrices. Utilizando las ecuaciones del modelo
    %físico de MRUA: X(t)=Xo+Vo*t+1/2*a*t^2  V(t)=Vo+a*t
    %A: Matriz de transición de estados (predicción del estado)
    %B: Matriz de control de entrada: efecto esperado de la
    %aceleración en el estado
    %C: Matriz de medida: la medida esperada dado el estado predicho
    A = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1];
    B = [(dt^2/2); (dt^2/2); dt; dt];
    C = [1 0 0 0; 0 1 0 0];
    %- Inicialización de los estados:
    %Accel_noise_mag es el ruido del proceso: desviación estándar de
    %la aceleración (metros/seg^2).
    %tkn_x y tkn_y es el ruido de la medida: desviación estándar de la
    %posición (metros) en el eje horizontal.
    %Ez convierte el ruido de medida (stdv) en la matriz de covarianza
    %Ex convierte el ruido del proceso (stdv) en matriz de covarianza
    %P es la estimación de la varianza de posición inicial (matriz
    %de covarianza)
    Accel_noise_mag = .3;
    tkn_x = 1;
    tkn_y = 1;
    Ez = [tkn_x 0; 0 tkn_y];
    Ex = [dt^4/4 0 dt^3/2 0;
        0 dt^4/4 0 dt^3/2; ...
        dt^3/2 0 dt^2 0; ...
        0 dt^3/2 0 dt^2].*Accel_noise_mag^2;
    P = Ex;
    %-------------------------------------------------------------------
else
    fprintf('Opcion de filtrado no valida\n');
    return
end

if(visualizar==1)
    fprintf('Visualizacion: Posicion del filtro\n');
elseif(visualizar==2)
    fprintf('Visualizacion: Mapa SSIM\n');
elseif(visualizar==3)
    fprintf('Visualizacion: Cruz de posicion detectada\n');
elseif(visualizar==4)
    fprintf('Visualizacion: Imagen real\n');
else
    fprintf('Opcion de visualizacion no valida\n');
    return
end
```

```
if(frame_inicio>=vid.NumberOfFrames)
    fprintf('Frame de inicio elegido supera la longitud del video\n');
elseif(frame_inicio<5)
    fprintf('El frame de inicio debe ser al menos 5\n');
    return
end


%%%%%%%%%%%%%%%%%%%%%%%%% BUCLE DE PROCESADO %%%%%%%%%%%%%%%%%%%%%%%%%

for k=frame_inicio:3:vid.NumberOfFrames

    %%%%%%%%%%%%%%%%%%%%%%%% ADQUISICION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Se extraen dos fotogramas del video
    img1_rgb=read(vid,k);
    img2_rgb=read(vid,k-3);

    img1=(rgb2gray(img1_rgb));
    img2=(rgb2gray(img2_rgb));


    % De los fotogramas completos adquiridos nos quedamos con solo una
    % parte sobre la que se va a trabajar para aplicar el algoritmo
    % SSIM, considerando que esta zona del fotograma es lo único que
    % puede captar una cámara móvil ficticia.
    % Las imágenes que se comparan tienen ancho ventX pixeles y un
    % alto ventY pixeles. Se centran sobre la posición del filtro en
    % el instante actual y el anterior, respectivamente, para las dos
    % imágenes capturadas.
    i2=img2((round(xk(2))-(ventY/2)):(round(xk(2))+(ventY/2-1)),...
        (round(xk(1))-(ventX/2)):(round(xk(1))+(ventX/2-1)));
    i1=img1((round(xk2(2))-(ventY/2)):(round(xk2(2))+(ventY/2-1)),...
        (round(xk2(1))-(ventX/2)):(round(xk2(1))+(ventX/2-1)));


    % Se calcula el desplazamiento relativo entre las dos imágenes,
    % aplicando el índice SSIM simplificado: se desplaza la primera
    % imagen sobre la segunda en todas las direcciones y se aplica la
    % aproximación al SSIM. La posición con el índice más alto se
    % corresponde con diferencia de posición entre las dos imágenes
    % capturadas. Con este desplazamiento se "recortan" las dos
    % imágenes para tomar las zonas en que se solapan (donde tienen el
    % mismo fondo), y generar dos imágenes reducidas sobre las que se
    % aplicara el algoritmo de detección SSIM.
    [pos_fila,pos_columna]=correccion_desplazamiento(i1,i2,window);


    % Se crean las imágenes reducidas con el mismo fondo
    if pos_columna<0 && pos_fila<0
        i2_reducida=i2(1-pos_fila:end,1-pos_columna:end);
        i1_reducida=i1(1:end+pos_fila,1:end+pos_columna);
    elseif pos_columna>=0 && pos_fila>=0
        i2_reducida=i2(1:end-pos_fila,1:end-pos_columna);
        i1_reducida=i1(1+pos_fila:end,1+pos_columna:end);
    elseif pos_columna>=0 && pos_fila<0
```

```
    i2_reducida=i2(1-pos_fila:end,1:end-pos_columna);
    i1_reducida=i1(1:end+pos_fila,1+pos_columna:end);
elseif pos_columna<0 && pos_fila>=0
    i2_reducida=i2(1:end-pos_fila,1-pos_columna:end);
    i1_reducida=i1(1+pos_fila:end,1:end+pos_columna);
end


%%%%%%%%%%%%%%%%%%%%%%%%% DETECCION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Se calcula el mapa SSIM con las imágenes reducidas. Se calcula
% su tamaño, las variables f1 y c1 recogen el valor de filas
% (alto) y columnas (ancho) del mapa SSIM, que será menor que el
% de las imágenes i1 e i2.
[~,ssim_map1]=ssim(i2_reducida,i1_reducida);
[f1,c1]=size(ssim_map1);


% Para situar el mapa SSIM obtenido sobre la imagen correctamente,
% hay que concatenar varias matrices de unos al mapa SSIM: primero
% se añade encima del mapa una matriz de alto la distancia desde
% el margen superior al borde superior de la ventana, y ancho el
% ancho del mapa (c1). Segundo se añade por la izquierda una
% matriz de alto la distancia desde el borde superior al margen
% superior de la ventana más el alto del mapa SSIM (f1), de ancho
% la distancia desde el margen izquierdo hasta el borde izquierdo
% del mapa.
suma1=ones(round(xk2(2))-ventY/2,c1);
ssim_map2=[suma1;ssim_map1];
suma2=ones(f1+round(xk2(2))-ventY/2,round(xk2(1))-ventX/2);
ssim_map=[suma2,ssim_map2];

% Se obtienen las dimensiones del mapa SSIM final
[f,c]=size(ssim_map);
fondo=zeros(f,c);


% Con el script dibujar podemos representar sobre la imagen los
% puntos que se corresponden con los objetos presentes.
Cruz=img1_rgb;

bin=im2bw(ssim_map,0.5);
bin=1-bin;
bin=imfill(bin);
bin=imdilate(bin,H);
bin=imfill(bin);
bin=imdilate(bin,H);

estr = bwconncomp(bin, 4);
caja=regionprops(estr,'BoundingBox','Area','Centroid');
area=0;
area_maxima=0;

for u=1:estr.NumObjects
    aa=round(caja(u).BoundingBox(1))+5;
    bb=aa+caja(u).BoundingBox(3)-5;
```

```
    cc=round(caja(u).BoundingBox(2))+5;
    dd=cc+caja(u).BoundingBox(4)-5;

    area=area+caja(u).Area;

    %Las variables pmx y pmx llevan el punto central del objeto
    %detectado. Estas coordenadas deben llevarse al filtro para la
    %etapa de seguimiento.
    if(caja(u).Area>area_maxima)
        area_maxima=caja(u).Area;
        pmx=round((aa+bb)/2)+5;
        pmy=round((cc+dd)/2)+5;
    end

    if visualizar==3
        Cruz=punto(Cruz,aa,bb,cc,dd);
    end
end


% Podemos calcular el porcentaje de la imagen que en el que se
% detecta movimiento mediante el área calculada, y elegir si es
% suficiente para considerar que se ha producido un movimiento

porcen=(area*100)/(c*f);
if(0.08>porcen)
    xm=[NaN NaN];
    contador=0;
else
    %La posición detectada se pasa en la variable xm
    xm=[pmx pmy];
    contador=contador+1;
end


%%%%%%%%%%%%%%%%%%%%%%%%% SEGUIMIENTO %%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Se actualiza el valor del estado anterior (xk y vk) con la
% información actual.
xk=xk2;
vk=vk2;

%----------------------Fase de inicio-------------------
% Cuando no hay datos de ningún objeto moviéndose, el recuadro de
% seguimiento se va moviendo desde la posición central del área de
% vigilancia de derecha a izquierda.
% Cuando el recuadro llega a los límites del área, cambia el
% sentido del movimiento en la otra dirección. Esto se mantiene
% hasta que se detecta el movimiento de un objeto.

if sum(~isnan(xm))
    inicio=1;
end
% Movimiento durante el inicio=0
if(inicio==0 && xk(1)<=limites(1))
    vx_inicio=vx_inicio*-1;
```

```
    xk = [limites(1) xk(2)];
    vk =[vx_inicio 0];
end
if(inicio==0 && xk(1)>=limites(4))
    vx_inicio=vx_inicio*-1;
    xk = [limites(4) xk(2)];
    vk =[vx_inicio 0];
    if(opcion_filtrado==2)
        P=Ex;
    end
end

%--------------------Fase de reseteo---------------------
% Si se pierde la información de posición (xm=[NaN NaN]) durante
% un tiempo determinado (ahora t=4) el recuadro de seguimiento
% vuelve a la posición inicial.

if (sum(isnan(xm)) && inicio==1)
    contadorNaN=contadorNaN+1;
else
    contadorNaN=0;
end
if(contadorNaN>=4)
    inicio=0;
    vk = [vx_inicio 0];
    if(opcion_filtrado==2)
        P=Ex;
    end
end

%-----------------------Fase de Filtrado---------------------
if(opcion_filtrado==1)
    % Se llama a la función que ejecuta el filtro Alfa-Beta
    [xk2,vk2,~]=Funcion_FiltroAlfaBeta(xm,dt,xk,vk,alfa,betta);
    pos_filtro=xk2;
elseif(opcion_filtrado==2)
    % Se llama a la función que ejecuta el filtro de Kalman
    [xk2,vk2,P]=Funcion_FiltroKalman(xk,vk,xm,A,B,C,acel,Ex,Ez,P);
    pos_filtro=xk2;
end

%-----------------------Fase de Corrección-------------------
% Se limita el movimiento del filtro a los bordes de la imagen:
% cuando se supera cierto límite por la izquierda, derecha, arriba
% o abajo se posiciona el filtro en posición correcta.
if(xk2(1)<limites(1))
    xk2(1)=limites(1);
end
if(xk2(1)>limites(4))
    xk2(1)=limites(4);
end
if(xk2(2)<limites(2))
    xk2(2)=limites(2);
end
if(xk2(2)>limites(3))
    xk2(2)=limites(3);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%% VISUALIZACIÓN %%%%%%%%%%%%%%%%%%%%%%%%%

% Para la visualización podemos elegir entre las distintas
% imágenes que hemos generado durante el programa: rgb contiene
% los píxeles en movimiento marcados en rojo, la imagen Cruz que
% utiliza la representación por puntos y la imagen img2_rgb es
% el fotograma capturado por la cámara.

if visualizar==1 || visualizar==4
    im_representar=img2_rgb;
elseif visualizar==2
    %Utilizando la función im_rgb generamos la imagen que se
    %corresponde con la imagen que tiene los píxeles de movimiento
    %marcados en rojo.
    rgb=im_rgb(uint8(img1_rgb),ssim_map,fondo);
    im_representar=rgb;
elseif visualizar==3
    im_representar=Cruz;
end

AA=round(xk2(1))-(ventX/2);
BB=round(xk2(1))+(ventX/2-1);
CC=round(xk2(2))-(ventY/2);
DD=round(xk2(2))+(ventY/2-1);

% En la imagen a representar se dibuja la ventana de detección de
% color verde, y la imagen se modifica para que el interior de
% esta ventana se mantenga en color mientras que el exterior se
% representa  como imagen en blanco y negro.
for jj=1:sizeX
    for kk=1:sizeY
    % Dibuja el exterior de la ventana en blanco y negro
    if(jj<AA || jj>BB || kk<CC || kk>DD)
        im_representar(kk,jj,2)=im_representar(kk,jj,1);
        im_representar(kk,jj,3)=im_representar(kk,jj,1);
    end
    % Dibuja los bordes de la ventana en verde
    if((jj==AA && kk>CC && kk<DD) || (jj==BB && kk>CC && kk<DD)...
            ||(kk==CC && jj>AA && jj<BB) ...
            || (kk==DD && jj>AA && jj<BB))
        im_representar(kk,jj,1)=0;
        im_representar(kk,jj,2)=255;
        im_representar(kk,jj,3)=0;
    end
    end
end

% Se visualiza la imagen con los resultados.
imshow(im_representar)
title(leyenda);
ylabel(sprintf('Video: %s     Frame: %d',nombre_video,k))

% Se incorpora la posición de los filtros
if visualizar==1
    hold on;
```

```
        plot(2*sin(j)+xk2(1),2*cos(j)+xk2(2),'b','LineWidth',3);
        plot(r*sin(j)+pos_filtro(1),r*cos(j)+pos_filtro(2),...
            'y','LineWidth',3);
        xlabel(sprintf('Posicion filtro: Coord.X: %d Coord.Y: %d', ...
            round(pos_filtro(1)),round(pos_filtro(2))));
        hold off
    end

end
```

- **Función ssim**

```
function [mssim, ssim_map] = ssim(img1, img2, K, window, L)

%
========================================================================
==
% SSIM Index with automatic downsampling, Version 1.0
% Copyright(c) 2009 Zhou Wang
% All Rights Reserved.
%
% ----------------------------------------------------------------------
% Permission to use, copy, or modify this software and its
% documentation for educational and research purposes only and without
% fee is hereby granted, provided that this copyright notice and the
% original authors' names appear on all copies and supporting
% documentation. This program shall not be used, rewritten, or adapted
% as the basis of a commercial software or hardware product without
% first obtaining permission of the authors. The authors make no
% representations about the suitability of this software for any
% purpose. It is provided "as is" without express or implied warranty.
%----------------------------------------------------------------------
%
% This is an implementation of the algorithm for calculating the
% Structural SIMilarity (SSIM) index between two images
%
% Please refer to the following paper and the website with suggested
% usage
%
% Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image
% quality assessment: From error visibility to structural similarity,"
% IEEE Transactios on Image Processing, vol. 13, no. 4, pp. 600-612,
% Apr. 2004.
%
% http://www.ece.uwaterloo.ca/~z70wang/research/ssim/
%
% Note: This program is different from ssim_index.m, where no
% automatic downsampling is performed. (downsampling was done in the
% above paper and was described as suggested usage in the above
% website.)
%
% Kindly report any suggestions or corrections to zhouwang@ieee.org
%
%----------------------------------------------------------------------
%
%Input:  (1) img1: the first image being compared
%        (2) img2: the second image being compared
%        (3) K: constants in the SSIM index formula (see the above
%            reference). defualt value: K = [0.01 0.03]
%        (4) window: local window for statistics (see the above
%            reference). default widnow is Gaussian given by
%            window = fspecial('gaussian', 11, 1.5);
%        (5) L: dynamic range of the images. default: L = 255
%
%Output: (1) mssim: the mean SSIM index value between 2 images.
%            If one of the images being compared is regarded as
%            perfect quality, then mssim can be considered as the
%            quality measure of the other image.
```

```
%            If img1 = img2, then mssim = 1.
%        (2) ssim_map: the SSIM index map of the test image. The map
%            has a smaller size than the input images. The actual size
%            depends on the window size and the downsampling factor.
%
%Basic Usage:
%   Given 2 test images img1 and img2, whose dynamic range is 0-255
%
%   [mssim, ssim_map] = ssim(img1, img2);
%
%Advanced Usage:
%   User defined parameters. For example
%
%   K = [0.05 0.05];
%   window = ones(8);
%   L = 100;
%   [mssim, ssim_map] = ssim(img1, img2, K, window, L);
%
%Visualize the results:
%
%   mssim                           %Gives the mssim value
%   imshow(max(0, ssim_map).^4)   %Shows the SSIM index map
%========================================================================

if (nargin < 2 || nargin > 5)
   mssim = -Inf;
   ssim_map = -Inf;
   return;
end

if (size(img1) ~= size(img2))
   mssim = -Inf;
   ssim_map = -Inf;
   return;
end

[M N] = size(img1);

if (nargin == 2)
   if ((M < 11) || (N < 11))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   window = fspecial('gaussian', 11, 1.5);  %
   K(1) = 0.01;                  % default settings
   K(2) = 0.03;                  %
   L = 255;                                  %
end

if (nargin == 3)
   if ((M < 11) || (N < 11))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   window = fspecial('gaussian', 11, 1.5);
```

```
   L = 255;
   if (length(K) == 2)
      if (K(1) < 0 || K(2) < 0)
           mssim = -Inf;
        ssim_map = -Inf;
        return;
      end
   else
       mssim = -Inf;
    ssim_map = -Inf;
       return;
   end
end

if (nargin == 4)
   [H W] = size(window);
   if ((H*W) < 4 || (H > M) || (W > N))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   L = 255;
   if (length(K) == 2)
      if (K(1) < 0 || K(2) < 0)
           mssim = -Inf;
        ssim_map = -Inf;
        return;
      end
   else
       mssim = -Inf;
    ssim_map = -Inf;
       return;
   end
end

if (nargin == 5)
   [H W] = size(window);
   if ((H*W) < 4 || (H > M) || (W > N))
       mssim = -Inf;
       ssim_map = -Inf;
      return
   end
   if (length(K) == 2)
      if (K(1) < 0 || K(2) < 0)
           mssim = -Inf;
        ssim_map = -Inf;
        return;
      end
   else
       mssim = -Inf;
    ssim_map = -Inf;
       return;
   end
end

img1 = double(img1);
img2 = double(img2);
```

```
% automatic downsampling
f = max(1,round(min(M,N)/256));
%downsampling by f
%use a simple low-pass filter
if(f>1)
    lpf = ones(f,f);
    lpf = lpf/sum(lpf(:));
    img1 = imfilter(img1,lpf,'symmetric','same');
    img2 = imfilter(img2,lpf,'symmetric','same');

    img1 = img1(1:f:end,1:f:end);
    img2 = img2(1:f:end,1:f:end);
end

C1 = (K(1)*L)^2;
C2 = (K(2)*L)^2;
window = window/sum(sum(window));

mu1   = filter2(window, img1, 'valid');
mu2   = filter2(window, img2, 'valid');
mu1_sq = mu1.*mu1;
mu2_sq = mu2.*mu2;
mu1_mu2 = mu1.*mu2;
sigma1_sq = filter2(window, img1.*img1, 'valid') - mu1_sq;
sigma2_sq = filter2(window, img2.*img2, 'valid') - mu2_sq;
sigma12 = filter2(window, img1.*img2, 'valid') - mu1_mu2;

if (C1 > 0 && C2 > 0)
   ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./((mu1_sq + mu2_sq
+ C1).*(sigma1_sq + sigma2_sq + C2));
else
   numerator1 = 2*mu1_mu2 + C1;
   numerator2 = 2*sigma12 + C2;
    denominator1 = mu1_sq + mu2_sq + C1;
   denominator2 = sigma1_sq + sigma2_sq + C2;
   ssim_map = ones(size(mu1));
   index = (denominator1.*denominator2 > 0);
   ssim_map(index) =
(numerator1(index).*numerator2(index))./(denominator1(index).*denomina
tor2(index));
   index = (denominator1 ~= 0) & (denominator2 == 0);
   ssim_map(index) = numerator1(index)./denominator1(index);
end

mssim = mean2(ssim_map);

return
```

- **Función punto**

```
%Esta función dibuja una cruz de color amarillo de 10x10 píxeles sobre
%el fotograma a visualizar que indica el centro de los objetos
%detectados anteriormente. La función recibe como parámetros la imagen
%RGB sobre la que se va a dibujar el punto y las coordenadas
%cartesianas de los cuatro vértices del rectángulo que envuelve el
%objeto, calculado en dibujo, para luego obtener el centro. Devuelve
%la imagen BB con la cruz dibujada.

%Se suman 5 a pmx y pmy para intentar compensar que el tamaño del mapa
%SSIM es menor que el de la imagen, para desplazar la cruz (igual que
%para el rectángulo)

function BB=punto(BB,aa,bb,cc,dd)

pmx=round((aa+bb)/2)+5;
pmy=round((cc+dd)/2)+5;


for ii=(pmx-5):(pmx+5)
    if(pmx>5 && pmx<316)
        BB(pmy,ii,1)=255;
        BB(pmy,ii,2)=255;
        BB(pmy,ii,3)=0;
    end
end
for ii=(pmy-5):(pmy+5)
    if(pmy>5 && pmy<226)
        BB(ii,pmx,1)=255;
        BB(ii,pmx,2)=255;
        BB(ii,pmx,3)=0;
    end
end
```

- **Función im_rgb**

```
% Esta función se encarga de marcar en rojo sobre la imagen a color
% los píxeles del mapa SSIM que tienen un valor menor que 0.5, siempre
% que estos no pertenezcan a la zona excluida del análisis por el
% proceso de calibrado. La función recibe la imagen a color del
% instante actual, el mapa SSIM y la imagen fondo que se corresponde
% con la máscara generada en el calibrado. La función devuelve la
% imagen con los píxeles marcados.

function rgb=im_rgb(i2,ssim_map,fondo)

[f,c]=size(fondo);

for ii=1:f
    for jj=1:c
        if(ssim_map(ii,jj)<0.50 && fondo(ii,jj)==0)
            i2(ii+5,jj+5,1)=255;
            i2(ii+5,jj+5,2)=0;
            i2(ii+5,jj+5,3)=0;
        end
    end
end
rgb=i2;
```

- **Funcion_FiltroKalman**

```
function [xk,vk,P]=Funcion_FiltroKalman(xk,vk,xm,A,B,C,u,Ex,Ez,P)

x_medido=[xm(1);xm(2)];
x_estimado=[xk(1);xk(2);vk(1);vk(2)];

% Fase de Predicción------------------
% Predicción del siguiente estado (posición y velocidad) con el último
% estado y movimiento predicho. El estado inicial se corresponde con
% la primera posición medida y velocidad nula.
x_estimado = A * x_estimado + B * u;
% Predicción de la siguiente matriz de covarianza
P = A * P * A' + Ex;

% Fase de Actualización-----------------
% Cálculo de la ganancia de Kalman
K = P*C'/(C*P*C'+Ez);
% Actualizar el estado estimado (solo si se reciben datos)
if ~isnan(x_medido)
    x_estimado = x_estimado + K * (x_medido - C * x_estimado);
end
% Actualizar la covarianza estimada
P =  (eye(4)-K*C)*P;

xk=[x_estimado(1) x_estimado(2)];
vk=[x_estimado(3) x_estimado(4)];
```

- **Funcion_FiltroAlfaBeta**

```
function [xk,vk,rk]=Funcion_FiltroAlfaBeta(xm,dt,xk,vk,alfa,betta)


% Se ejecuta el filtro Alfa-Beta en dos etapas:
%1) Predicción de las estimaciones para posición y velocidad.
%2) Corrección de las estimaciones.

xkp = xk + dt * vk;          % Estimación de posición
vkp = vk;                    % Estimación de velocidad (la misma que en
la iteración anterior)
rk = xm - xkp;               % Error (residuo)

% Si no se reciben datos la estimación no se corrigen las estimaciones
% de posición (xkp) y velocidad (vkp), se siguen utilizando los
% valores de iteraciones anteriores
if ~isnan(xm)
    xkp = xkp + alfa * rk;
    vkp = vkp + betta/dt * rk;
end
% Se pasan los valores a la siguiente iteración del filtro
% 1) La posición estimada (y corregida) se pasa a la siguiente
% iteración en xk.
% 2) La velocidad estimada (y corregida) se pasa a la siguiente
% iteración en vk.
xk = xkp;
vk = vkp;
```

- **Función corrección_desplazamiento**

```
function
[pos_fila,pos_columna]=correccion_desplazamiento(i1,i2,window)

%Esta función recibe las dos imágenes correspondientes a las dos
%ventanas de detección y devuelve el desplazamiento entre ambas, el
%número de filas y columnas. Se realiza una versión simplificada del
%SSIM, utilizando las imágenes media y covarianza de las originales,
%estas nuevas imágenes se deslizan una sobre otra de izquierda a
%derecha y de arriba a abajo y, para cada iteración, se calcula el
%SSIM simplificado sobre la zona que se solapa.
%Al finalizar el bucle se halla la posición en filas y columnas donde
%se encuentra el valor máximo (donde los fondos de ambas imágenes
%coinciden).
%Los valores de columna y fila se devuelven a la función principal
%para poder construir las imágenes reducidas.


i1d=double(i1);
i2d=double(i2);

m1=filter2(window,i1d,'valid');
m2=filter2(window,i2d,'valid');
v1=filter2(window,i1d.^2,'valid')-m1.^2;
v2=filter2(window,i2d.^2,'valid')-m2.^2;

F1=zeros(50,50);

for uu=(-30+2):1:(30-1)
    for ii=(-30+2):1:(30-1)
        if uu<0 && ii<0
            m2c=m2(1-ii:end,1-uu:end);
            m1c=m1(1:end+ii,1:end+uu);
            v2c=v2(1-ii:end,1-uu:end);
            v1c=v1(1:end+ii,1:end+uu);
        elseif uu>=0 && ii>=0
            m2c=m2(1:end-ii,1:end-uu);
            m1c=m1(1+ii:end,1+uu:end);
            v2c=v2(1:end-ii,1:end-uu);
            v1c=v1(1+ii:end,1+uu:end);
        elseif uu>=0 && ii<0
            m2c=m2(1-ii:end,1:end-uu);
            m1c=m1(1:end+ii,1+uu:end);
            v2c=v2(1-ii:end,1:end-uu);
            v1c=v1(1:end+ii,1+uu:end);
        elseif uu<0 && ii>=0
            m2c=m2(1:end-ii,1-uu:end);
            m1c=m1(1+ii:end,1:end+uu);
            v2c=v2(1:end-ii,1-uu:end);
            v1c=v1(1+ii:end,1:end+uu);
        end
        F1(ii+1+30-2,uu+1+30-2)=mean2(((2*(m1c.*m2c)+6.5025).*...
            (2*sqrt(v1c.*v2c)+58.5225))./(...
            (m1c.^2+m2c.^2+6.5025).*(v1c+v2c+58.5225)));
    end
end
```

```
[valor,posicion]=max(F1,[],2);
[~,pos_fila]=max(valor);
pos_columna=posicion(pos_fila);
pos_columna=pos_columna-30+1;
pos_fila=pos_fila-30+1;
```