UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

Seguimiento automático de objetos en sistemas con múltiples cámaras

Carlos Legua Cruz

Mayo 2013

Seguimiento automático de objetos en sistemas con múltiples cámaras

AUTOR: Carlos Legua Cruz

TUTOR: Luis Fernando Lago Fernández

Grupo de Neurocomputación Biológica (GNB)

Dpto. de Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Mayo de 2013

Resumen

En los últimos años, han surgido multitud de algoritmos que tienen como objetivo el seguimiento automático de objetos en secuencias de vídeo. Recientemente, los sistemas de seguimiento de objetos que utilizan múltiples cámaras han tomado una mayor relevancia, puesto que solucionan problemas como la oclusión o el cambio de orientación del objetivo. Estos sistemas se dividen en tres fases: detección, seguimiento y fusión de información. Muchos de los métodos utilizan puntos característicos del objeto para el seguimiento, y emplean relaciones geométricas entre las cámaras para la fusión de información.

En este proyecto se ha implementado un sistema de detección y seguimiento de objetos con dos cámaras, denominado **M**oving **O**bject **DE**tection and **T**racking (**MODET**). Este algoritmo hace foco en la utilización de los puntos de interés de las imágenes que captan las cámaras. Tanto para la detección como para el seguimiento de los objetos se ha utilizado el detector y descriptor de puntos de interés SURF, y se utiliza el algoritmo de alineamiento de puntos RAMOSAC para establecer correspondencias entre los puntos minimizando el error. En la fase de fusión de información de las cámaras se ha empleado la transformación homográfica, que establece relaciones entre las imágenes captadas por las cámaras. En este método no es necesario calibrar las cámaras, ya que todos los parámetros de **MODET**, que se ajustan durante una corta fase de entrenamiento, dependen de la escena.

MODET se ha evaluado en diferentes escenarios y se ha comparado con el algoritmo Mean Shift. La primera prueba se ha realizado sobre videos grabados durante el desarrollo del proyecto en un escenario interior controlado y en los que interviene un único objeto. En la segunda prueba se ha aumentado la dificultad de los vídeos, utilizando el mismo escenario pero con varios objetos en movimiento simultáneamente. En la tercera prueba, el algoritmo se ha evaluado con videos tomados de la base de datos PETS 2001, que constituyen un marco de comparación habitual en el área. Por último, **MODET** se ha testado en videos con una única cámara en movimiento, descargados del portal YouTube. Los resultados demuestran que el algoritmo es capaz de seguir los objetos con gran exactitud en distintos escenarios, alcanzando mayor precisión que Mean Shift en las comparativas realizadas.

Palabras clave

Seguimiento de objetos, Puntos característicos, SURF, RANSAC, RAMOSAC, Homografía, Detección de movimiento, Visión artificial, Video vigilancia.

Abstract

In the last years, many object tracking techniques have been proposed. Recently, tracking systems using multiple cameras have become more relevant, as they are able to solve issues such as occlusions or changes in target orientation. The usual approach is divided into three steps: detection, tracking and information fusion. Many of these systems use the object's feature points for tracking and use geometric relationships between the cameras for information fusion.

In this work a system for object detection and tracking using two cameras, called **M**oving **O**bject **DE**tection and **T**racking (**MODET**), has been developed. The algorithm focuses on interest points extracted from the images taken by the cameras.. A homographic transformation is applied in the information fusion step to establish the relationship between images captured by the cameras. It is not necessary to calibrate the cameras in this method, since all **MODET** parameters are adjusted during a short training stage.

MODET has been tested in different scenarios and has been compared to the Mean Shift algorithm. The first evaluation was performed on videos which were recorded as part of this project, in a controlled environment and with a single object. In a second evaluation the difficulty Detection and description of interest points is performed using the SURF method, and the algorithm RAMOSAC is used to establish point correspondences minimizing the error was increased, using the same scenario but with several moving objects. In the third test, the algorithm has been evaluated on outdoor sequences taken from the PETS 2001 database, which is a common benchmark in the area. Finally, **MODET** has been evaluated on videos with a single moving camera downloaded from YouTube. The results show that the algorithm is able to track objects with a high precision in a variety of scenarios, obtaining higher accuracy than Mean Shift on the comparatives performed.

Key words

Object Tracking, Keypoint, SURF, RANSAC, RAMOSAC, Homography, Motion detection, Computer Vision, Video Surveillance.

Agradecimientos

Llegado el momento de cerrar mi vida universitaria, me gustaría agradecer a todas aquellas personas con las que he compartido estos años.

En primer lugar me gustaría dar las gracias a mi tutor, Luis Lago, que me ha brindado la oportunidad de realizar este proyecto tan ambicioso y apasionante. Por los consejos recibidos y sobre todo, por el tiempo que me ha dedicado. También a todos los profesores que he tenido durante la carrera por todos los conocimientos que he adquirido gracias a ellos.

Agradecer a todos los buenos amigos que he conocido durante todos estos años en la universidad, sin vosotros nada hubiese sido igual. Aunque ya vayamos acabando, siempre quedaran nuestras "quedadas". Chema y Tomás, seguro que seguimos en contacto por Communicator. En especial quiero resaltar a Ángel, gracias a estos años, nos une una grandísima amistad.

Asimismo, agradecer a aquellas personas que he conocido en mi vida laboral. A mi compañera Patricia por todos los consejos y ayuda prestada.

También quiero agradecer a Sandra, por haber estado a mi lado todos estos años, por estar siempre dispuesta a ayudarme en todo y por las experiencias vividas juntos.

No quisiera olvidarme del resto de mis amigos porque, aún sin saber muy bien lo que hacía, siempre estaban interesándose. Por supuesto, gracias a Pablo, por las risas y grandes momentos que hemos pasado juntos.

En definitiva, gracias a toda la gente que he conocido durante todos estos años, tanto en la universidad como fuera de ella.

Por último, gracias a toda mi familia, en especial a mis padres y mi hermano por haber estado a mi lado en todo momento, dándome ánimos y ayuda durante toda mi vida.

Muchas gracias a todos.

Carlos Legua Cruz

Mayo 2013

Índice de contenidos

Índice de figuras

Índice de tablas

1 INTRODUCCIÓN

1.1 Motivación

El seguimiento automático de objetos o tracking es una tarea muy importante de visión artificial, con múltiples aplicaciones en campos como la vigilancia, el reconocimiento de objetos basado en movimiento, la interacción persona-ordenador, la monitorización del tráfico o la navegación automática. El seguimiento de un objeto puede definirse como la estimación, a partir de una secuencia de *frames* obtenida con una cámara de vídeo, de la trayectoria del objeto sobre el plano de la imagen a medida que se mueve por la escena.

En aplicaciones de video vigilancia, el caso más frecuente suele ser el seguimiento de personas y vehículos en la zona vigilada, para detectar posibles actividades sospechosas, las cuales cobran cada vez mayor importancia en los sistemas actuales de vídeo vigilancia debido a la gran cantidad de gente que se concentra en esos lugares y los riesgos potenciales de seguridad que ello implica.

Dado que las cámaras tienen un campo de visión limitado, normalmente es necesario utilizar más de una cámara para cubrir todo el espacio de interés. Típicamente, las aplicaciones de video vigilancia presentan las imágenes de todas las cámaras a un observador para su análisis. No obstante, la capacidad de una persona para concentrarse en varios vídeos de forma simultánea es muy limitada. De aquí que, últimamente haya habido un gran interés en el desarrollo de sistemas de visión artificial, capaces de analizar simultáneamente los datos provenientes de un conjunto de cámaras y, presentar el análisis de una manera más compacta y simbólica a los usuarios finales. Por otra parte, el uso de varias cámaras puede permitir recuperar la trayectoria real de los objetos en 3D.

Para cubrir el área de interés, normalmente se utilizan cámaras cuyos campos de visión están solapados. Esto presenta ventajas e inconvenientes. Por un lado, al contar con varias vistas de los objetos es posible extraer información sobre la estructura tridimensional de la escena. Por otro, se genera cierta ambigüedad a la hora de seguir los objetos, puesto que un mismo objeto puede tener aspecto diferente bajo distintos puntos de vista. Es por tanto necesario identificar las múltiples proyecciones de los objetos como el mismo elemento 3D, y etiquetarlas de manera consistente en todas las cámaras si se quiere utilizar este tipo de sistemas en aplicaciones de seguridad o vigilancia.

1.2 Objetivos

En este trabajo se explorarán sistemas de seguimiento de objetos basados en la detección de puntos característicos. Según este planteamiento, los objetos se representan como una colección de características de bajo nivel, como esquinas o segmentos, y el seguimiento de los mismos se basa en la búsqueda de correspondencias entre estas características en *frames* consecutivos. Existen numerosos detectores de características que se han usado en la literatura para este fin, como MSER, esquinas de tipo Harris-Affine o Hessian-Affine, o los más

populares SIFT y SURF. Estos últimos presentan además invariancia frente a escala, rotación y cambios de iluminación, lo cual los hace especialmente atractivos para sistemas reales. La correspondencia entre los puntos característicos de dos *frames* sucesivos puede utilizarse para estimar el movimiento de los objetos. Partiendo de sistemas con una única cámara, se estudiará su extensión a sistemas con dos o más cámaras que permitan fusionar la información para mejorar el seguimiento.

El objetivo final del proyecto es el diseño e implementación de un sistema completo capaz de realizar detección y seguimiento de objetos en movimiento, utilizando puntos característicos, en sistemas con al menos dos cámaras. Para ello, sus campos de visión deberán estar parcialmente solapados, y en un entorno controlado.

Para alcanzar los objetivos que se plantean, el proyecto se desarrollará de acuerdo a las siguientes tareas:

- **Estudio de literatura y estado del arte:** En primer lugar, revisaremos la literatura en relación al seguimiento automático de objetos y haremos un estudio del estado del arte de las diferentes técnicas, en particular, trabajos cuyo enfoque sea el uso de puntos característicos y sistemas para múltiples cámaras.
- **Elaboración de banco de vídeos:** A continuación, recopilaremos distintas secuencias de vídeo para realizar los análisis y pruebas de evaluación de los sistemas que desarrollemos. Además de utilizar bases de datos que sean un estándar en la literatura, por ejemplo los vídeos de PETS 2001, se grabarán vídeos propios en un entorno más controlado.
- **Diseño e implementación de algoritmos:** Una vez que hemos decidido qué técnicas vamos a utilizar, implementaremos los distintos algoritmos, pudiendo utilizar librerías ya desarrolladas, para la detección y seguimiento de objetos utilizando puntos característicos y cámara simple. Una vez evaluados, se seleccionarán los que ofrezcan mejores soluciones para el problema planteado.
- **Extensión a sistemas multicámara:** Después de implementar el sistema para una cámara haremos una extensión del mismo para establecer una fusión de información entre, al menos, dos cámaras.
- **Diseño del sistema final**: Combinando los resultados de las dos fases anteriores se diseñará e implementará el sistema final.
- **Evaluación de resultados:** Posteriormente evaluaremos el sistema con los distintos vídeos recopilados en el punto 2. También compararemos los resultados con otras implementaciones de tracking estándar en la literatura, como Mean Shift. En la página web clegua.modet.es pueden verse algunos vídeos con ejemplos de aplicación del algoritmo desarrollado.
- **Análisis de la viabilidad y conclusiones:** Analizaremos su posible aplicación en un sistema real, extraeremos las conclusiones finales de este proyecto y propondremos posibles continuaciones del trabajo realizado.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte:** Se establecerá el marco teórico en el que se encuentra este PFC y se particularizará en los métodos de seguimiento por puntos donde se clasifica nuestro sistema.
- **Diseño:** Se presentará el diseño del sistema planteado. Se desglosará cada una de las fases de este sistema y se escogerán los algoritmos utilizados para el desarrollo del mismo.
- **Desarrollo:** Este capítulo se centrará en la implementación del sistema. Se detallará cada módulo implementado y las funciones y parámetros utilizados.
- **Pruebas y resultados:** Se realizarán pruebas del sistema completo en distintos escenarios y con distintas complejidades. Posteriormente se mostrarán los resultados de estas pruebas.
- **Conclusiones y trabajo futuro:** En este apartado se expondrán las conclusiones de este PFC y se presentarán posibles continuaciones del trabajo realizado.
- **Anexos:** Serie de anexos referentes a información complementaria acerca de los capítulos del proyecto.

2 ESTADO DEL ARTE

En este capítulo presentaremos una visión global sobre el estado actual del seguimiento de objetos o tracking en una y en varias cámaras, así como los algoritmos y técnicas más utilizados. En particular, se realizará un estudio más orientado hacia el seguimiento mediante puntos característicos por ser el utilizado en este proyecto.

En la primera parte proporcionaremos una descripción general de los métodos de tracking de objetos en vídeo. Después, nos centraremos en los algoritmos de tracking basados en puntos característicos existentes en la literatura. Finalmente, en la última parte, pasaremos a analizar las técnicas utilizadas en el seguimiento de objetos con múltiples cámaras.

2.1 Tracking en vídeo

El seguimiento de un objeto o tracking es el proceso de estimar, a partir de una secuencia de vídeo, la trayectoria del objeto a medida que se mueve por la escena. Todos los métodos de tracking tienen que localizar primero un objeto en la escena y posteriormente seguirlo en el tiempo. En la literatura, muchos estudios enfocan su investigación solamente en el seguimiento, ya que resuelven la localización del objeto marcándolo manualmente. Comúnmente la obtención de las secuencias se realiza mediante el uso de cámaras de vídeo. El tracking ha suscitado un alto interés en el área del análisis de vídeo por ordenador, gracias al avance en la potencia de los procesadores ligado al menor coste de cámaras de vídeo de alta resolución.

Tiene un gran número de aplicaciones, tales como:

- Identificación humana.
- Detección automática de objetos.
- Automatización de la seguridad y vídeo vigilancia.
- Comunicación y compresión de vídeo.
- Realidad aumentada.
- Control de tráfico.

Uno de los objetivos de este PFC es la estimación de la trayectoria de un objeto mientras se mueve alrededor de la escena. Dependiendo del tipo de objeto, este tipo de seguimiento puede ser muy complejo a causa de:

- Pérdida de información causada por la proyección del mundo 3D a una imagen 2D.
- Baja calidad de las imágenes.
- Oclusiones y formas complejas de los objetos.
- Alta velocidad de movimiento y cambios de orientación.
- Ruido y cambios de iluminación en la escena.

Para evitar gran parte de estos problemas se imponen restricciones en base a la aparición y movimiento de los objetos. Por ejemplo, algunos algoritmos de tracking suponen que el

movimiento de los objetos es uniforme, otros restringen el movimiento del objeto a que la velocidad o aceleración sea constante. También se puede restringir el número y tamaño de los objetos con el fin evitar falsas apariciones.

A continuación mostraremos las principales clasificaciones de los métodos más utilizados en sistemas de detección y seguimiento de objetos.

2.1.1 Detección de objetos

La detección de objetos es la primera fase de todo sistema de seguimiento. Tiene por objetivo reconocer los objetos de interés que aparecen en la escena.

Todos los sistemas de tracking requieren un mecanismo de detección de objetos, ya sea en cada *frame* o sólo cuando el objeto aparece por primera vez en el vídeo. La localización de objetos puede darse manualmente, seleccionando el contorno del objeto, o automáticamente, mediante diferentes algoritmos de detección.

Un enfoque común de la detección de objetos es el uso de información de un único *frame*. Sin embargo, algunos métodos utilizan la información adquirida temporalmente en varias secuencias de *frames* para reducir el número de falsas detecciones como en [1].

El primer paso de la detección consiste en distinguir entre las regiones en movimiento, que corresponden a objetos móviles como personas o vehículos, y el fondo. Una vez que se adquieren las regiones en movimiento hay que extraer las características de cada uno de los objetos. Posteriormente, se define el tipo de representación que se va a utilizar para definir al objeto detectado.

A continuación describiremos las técnicas más habituales en la localización de regiones en movimiento y, en segundo lugar, estudiaremos los distintos tipos de representación que se pueden utilizar para delimitar un objeto y realizar su posterior seguimiento.

2.1.1.1 Localización de regiones en movimiento

El principal objetivo de la localización de regiones en movimiento es la distinción entre el primer plano de los objetos y el fondo estático de la escena. Debido a los cambios dinámicos, que se producen en entornos naturales, es muy complicado alcanzar detecciones de movimiento fiables.

Las técnicas más frecuentes de localización de regiones en movimiento son las siguientes (en la tabla 2.1 mostramos un resumen):

- **Sustracción de fondo:** La sustracción de fondo es una técnica muy utilizada para la localización de movimiento en escenas con fondo estático [2] o para la localización de objetos abandonados [3]. Este método consiste en detectar las regiones de movimiento restando pixel a pixel la imagen actual $I_t(x, y)$ con una imagen referencia

del fondo $B_t(x, y)$. Los píxeles donde la diferencia entre la imagen de referencia y la actual son superiores a un umbral T, se clasifican como primer plano.

$$\left|I_t(x, y) - B_t(x, y)\right| > T \quad (2.1)$$

La imagen de referencia se va actualizando con el tiempo para adaptarse a los cambios de la escena. La ecuación que define la nueva imagen de referencia es la siguiente,

$$B_{t+1}(x, y) = \alpha I_t(x, y) + (1 - \alpha) B_t(x, y) \quad (2.2)$$

donde α es el coeficiente de adaptación.

A pesar de ser una buena técnica para detectar movimiento, puede introducir más ruido que otros métodos frente a cambios dinámicos como sombras, brillos, luces, etc.

- **Métodos estadísticos:** Se han desarrollado métodos estadísticos más avanzados que superan las deficiencias de los métodos de sustracción de fondo, inspirados en éstos. Las estadísticas del fondo se actualizan dinámicamente durante el proceso. En cada fotograma, este método almacena y actualiza las estadísticas correspondientes al fondo. Para identificar los puntos del primer plano, se comparan las estadísticas de cada píxel con las estadísticas de la imagen de referencia. Esta técnica es cada vez más popular debido a su robustez frente a escenas que contengan ruido y cambios de iluminación. Además se caracteriza por su utilización en sistemas de vídeo vigilancia en tiempo real como en el método W4 [4] o el presentado por Stauffer y Grimson [5].
- **Diferencias temporales:** Los métodos de diferenciación temporal hacen uso de la comparación de píxel a píxel entre dos o tres *frames* consecutivos para detectar regiones en movimiento. Una de las virtudes de este método es la adaptabilidad a los cambios dinámicos en la escena, pero por el contrario, no obtiene todos los píxeles correspondientes al primer plano en objetos que tienen una textura uniforme o se mueven lentamente. En esta clase de métodos, la extracción de los píxeles en movimiento es simple y rápida. Lipton [6] presenta un esquema en el que se comparan sólo los píxeles del *frame* actual y los píxeles del *frame* previo.

$$\left|I_t(x, y) - I_{t-1}(x, y)\right| > T \quad (2.3)$$

- **Flujo óptico:** Los algoritmos de flujo óptico detectan las regiones en movimiento a través del estudio de los vectores de flujo de la secuencia. Con este método se pueden detectar objetos en movimiento, incluso en secuencias grabadas con cámaras en movimiento. Sin embargo, la mayoría de los métodos de flujo óptico son computacionalmente complejos, lo que conlleva altos tiempos de ejecución. Deng [7] combina la diferencia entre *frames* y el flujo óptico para detectar objetivos en movimiento. Wedel [8] presenta un enfoque para identificar y segmentar objetos en movimiento a partir de la información de densidad de flujo de la escena.
- **Detección de cambios de luz:** Los algoritmos de detección de movimiento descritos anteriormente funcionan correctamente en escenarios interiores y exteriores. Sin embargo, son susceptibles a cambios locales de iluminación, como sombras y luces, y cambios globales, como variaciones en la iluminación solar, haciendo que estas técnicas sean inexactas. Horprasert [9] presenta un algoritmo de detección de objetos en movimiento en un escenario con cambios de iluminación. Cada píxel se representa

por un modelo de color que separa el brillo de la componente de cromaticidad. Así, los píxeles se clasifican en una de estas cuatro categorías:

- Fondo
- Fondo sombreado
- Fondo resaltado
- Objeto de primer plano en movimiento

Esta clasificación se realiza mediante el cálculo de la distorsión de brillo y cromaticidad entre el fondo y los píxeles de la imagen actual.

En todos los algoritmos de localización de movimiento, las regiones en movimiento detectadas corresponden a diferentes objetivos. Para poder diferenciar estos objetivos, los píxeles de primer plano se agrupan en regiones conectadas. Estas regiones se representan de tal manera que se pueda realizar un posterior seguimiento a lo largo de distintos fotogramas, tal y como veremos en el apartado siguiente.

Tabla 2.1. Métodos de localización de regiones en movimiento.

Método	Referencia	Descripción
Sustracción de fondo	Mathew, Yu y Zhang [2]	Detección de objetos que aparecen en escenas con fondo estático.
	Liao, Chang y Chen [3]	Detección de equipaje abandonado.
Métodos estadísticos	Haritaoglu, Harwood y Davis [4]	Detección y seguimiento de personas en tiempo real.
	Stauffer y Grimson [5]	Segmentación de regiones en movimiento en tiempo real.
Diferencias temporales	Lipton, Fujiyoshi y Patil [6]	Clasificación y seguimiento de objetos en tiempo real.
Flujo óptico	Deng, Xiong y Ou [7]	Detección de objetos en movimiento.
	Wedel, Meißner, Rabe, Franke y Cremers [8]	Detección y seguimiento de objetos a través de la densidad de flujo de la escena.
Detección de cambios de luz	Horprasert, Harwood y Davis [9]	Detección de objetos en movimiento en escenarios con fondo estático.

2.1.1.2 Representación de objetos

Es muy importante representar de manera apropiada el objeto detectado con el fin de realizar un seguimiento fiable. Es por eso que cada tipo de representación se ajusta más a un objetivo que a otro. Por ejemplo, se utilizan distintas tipologías de representación para sistemas de detección de matrículas que para seguimiento de personas o vehículos.

Los diversos patrones que existen se pueden dividir en dos grandes grupos: modelos basados en formas y modelos basados en apariencia [10] (en la tabla 2.2 listamos ejemplos de los distintos grupos).

Los *métodos basados en formas* utilizan las siguientes características referentes a la forma del objeto para representarlo:

- **Puntos:** El objeto se representa por un punto (ver figura 2.1 (a)) [11], centroide, o por un conjunto de puntos (ver figura 2.1 (b)) [12]. Este método es adecuado para objetos que ocupan regiones pequeñas de la escena.
- **Formas geométricas**: El objeto se representa por formas geométricas tales como un rectángulo (ver figura 2.1 (c)) o una elipse (ver figura 2.1 (d)) [13]. Estas formas son adecuadas para la representación de objetos rígidos, aunque también son utilizadas en el seguimiento de objetos flexibles con gran éxito.
- **Silueta y contorno:** El objeto se representa por su contorno (ver figura 2.1 (g, h)), o por la región dentro del contorno, la silueta (ver figura 2.1 (i)). Este tipo de modelo es ideal para el seguimiento de formas flexibles como personas [14].
- **Formas articuladas:** Este método se basa en la representación de formas geométricas con la variación de utilizar una forma geométrica para cada una de las partes rígidas del objeto (ver figura 2.1 (e)) [15].
- **Esqueleto:** Consiste en la utilización del esqueleto del objeto para representarlo (ver figura 2.1 (f)). Esta técnica es muy utilizada en aplicaciones de reconocimiento de objetos [16].

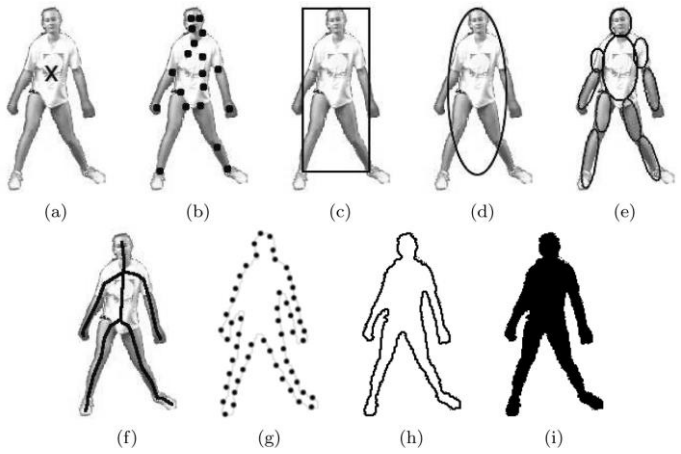

Figura 2.1. Representación de objetos. (a) Centroide. (b) Múltiples puntos. (c) Forma rectangular. (d) Forma elíptica. (e) Formas articuladas. (f) Esqueleto del objeto. (g) Contorno por puntos. (h) Contorno. (i) Silueta. Esta figura ha sido reproducida de [10].

Los *métodos basados en apariencia* utilizan información cromática o de contorno de los píxeles del objeto. Son muy utilizados en combinación con modelos basados en forma ya que, este tipo de métodos ofrece gran cantidad de información, aunque suele resultar insuficiente. Las representaciones de apariencia más comunes en el seguimiento de objeto son:

- **Densidad de probabilidad:** Los objetos pueden definirse por las funciones de densidad de probabilidad de que su color o textura sean localizados en sucesivos fotogramas. Para conseguir estas medidas se pueden utilizar diferentes funciones de densidad como Gaussianas [17] o mezclas de Gaussianas [18]. Las densidades de probabilidad de la apariencia de los objetos, como color o textura, pueden calcularse a partir de

regiones de la imagen especificadas por representaciones de forma, como elipses o rectángulos.

- **Patrones:** Están formados por formas geométricas o siluetas predefinidas. Tiene un gran rendimiento en el seguimiento de cabezas y manos [19]. Una de las ventajas de los patrones es que contienen información espacial e información cromática del objeto. El uso de estas técnicas sólo es adecuado para el seguimiento de objetos que no varían en el tiempo.
- **Modelo de apariencia activo:** Este modelo está basado en la generación de información sobre la forma y apariencia del objeto. La forma se define por un conjunto de puntos de referencia, similar a la representación de contorno por puntos y, para cada uno de estos puntos, se almacena un vector con información de su apariencia como el color, textura o magnitud del gradiente. Ha sido utilizado en métodos para la reconstrucción de caras [20] o detección de movimiento en tiempo real [21].
- **Modelo de apariencia multivista:** Esta representación almacena información de diferentes vistas del objeto para alcanzar un sistema menos susceptible a cambios de forma u orientación del objeto. Moghaddam [22] presenta una técnica de aprendizaje visual en la que utiliza para el modelado de datos dos tipos de funciones de densidad; Gaussianas multivariadas para distribuciones unimodales y mezcla de Gaussianas para distribuciones multimodales. El algoritmo *EigenTracking* [23] utiliza modelos de entrenamiento multivista para seguir y reconocer gestos de movimiento de manos.

Una vez que se han localizado y representado los objetos de interés, se realiza un seguimiento de los mismos en los siguientes fotogramas. Es por ello que la detección de objetos tiene una alta sensibilidad, una detección errónea de un objetivo será arrastrada a lo largo del sistema de tracking.

Tabla 2.2. Métodos de representación de objetos.

Tipo	Método	Referencia	Descripción
Métodos basados en formas	Puntos	Veenman, Reinders y Backer [11]	Se propone un algoritmo que optimiza el conjunto de puntos característicos a utilizar en el seguimiento de objetos.
		Serby, Koller-Meier y Gool [12]	Seguimiento de objetos utilizando características de puntos.
	Formas geométricas	Comaniciu, Ramesh y Andmeer [13]	Seguimiento de objetos representados con elipses.
	Silueta y contorno	Yilmaz, Li y Shah [18]	Seguimiento de objetos en sistemas con cámaras en movimiento.
	Formas articuladas	Sundaresan y Chellappa [15]	Seguimiento del movimiento de articulaciones humanas en sistemas multicámara.
	Esqueleto	Ali y Aggarwal [16]	Segmentación y reconocimiento de actividad humana.
Métodos basados en apariencia	Densidad de probabilidad	Zhu y Yuille [17]	Segmentación de áreas de interés en imágenes.
		Paragios y Deriche [18]	Segmentación automática de objetos.
	Patrones	Fieguth y Terzopoulos [19]	Seguimiento de caras y manos.
	Modelo de apariencia activo	Edwards, Taylor y Cootes [20]	Generación de caras.
		Cootes, Edwards y Taylor [21]	Detección de movimiento en tiempo real.
	Modelo de apariencia multivista	Moghaddam y Pentland [22]	Técnica de aprendizaje para la detección y reconocimiento de caras o manos.
		Black y Jepson [23]	Seguimiento y reconocimiento de gestos con manos.

2.1.2 Seguimiento de objetos

El objetivo principal del seguimiento de objetos es identificar las relaciones afines de un mismo objeto entre *frames* adyacentes con el fin de obtener su transformación de movimiento, su posición y el área que lo delimita en cada *frame* del vídeo. Es un proceso iterativo que se ejecuta a continuación de la detección del objeto y depende del modelo de representación del objeto seleccionado. Cada ciclo del algoritmo se divide en dos etapas:

- **Predicción:** se estima la posición del objeto en el siguiente fotograma de la secuencia.
- **Fase de corrección:** se actualiza la información del objeto con los datos extraídos en el nuevo *frame*, para conseguir una mejor estimación en los siguientes ciclos del seguimiento.

El seguimiento es la parte con mayor carga computacional ya que necesita información de la escena e información previa del objeto. Dependiendo de la calidad, tamaño y duración del vídeo, también puede ser necesaria una gran capacidad de almacenamiento. Los algoritmos de seguimiento de objetos pueden clasificarse en tres grandes grupos: Tracking por puntos, núcleo y silueta (ver figura 2.2). Mostramos ejemplos de esta clasificación en la tabla 2.3.

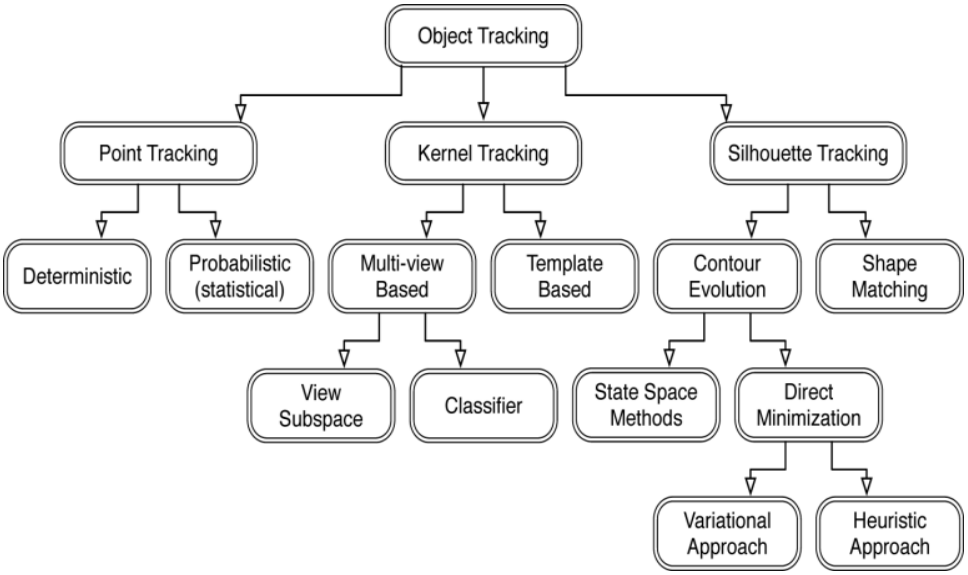

Figura 2.2. Clasificación de los métodos de seguimiento de objetos. Esta figura ha sido reproducida de [10].

El ***tracking por puntos*** se puede definir como la correspondencia entre los puntos de un *frame* que definen un objeto con respecto a los mismos puntos del *frame* anterior, para así poder predecir su movimiento [24]. Estos puntos se eligen de modo que sean robustos frente a cambios de escala, rotación o transformaciones afines. Estos algoritmos se pueden clasificar en dos categorías:

- **Métodos deterministas:** Buscan correspondencias entre los puntos del objeto en el instante de la secuencia $t-1$ con los puntos del objeto en el instante t (ver figura 2.3). Para ello se utilizan todas las combinaciones posibles de correspondencia de puntos y posteriormente se escogen las correctas mediante métodos de asignación óptimos. Para estas correspondencias, generalmente se definen una combinación de limitaciones como proximidad, máxima velocidad, pequeños movimientos, rigidez, etc. Este tipo de restricciones también se utilizan en métodos estadísticos.
  Salari y Sethi [25] proponen un algoritmo de seguimiento de objetos basado en métodos deterministas que soluciona los problemas de oclusión y de puntos característicos de baja calidad. Veenman [26] utiliza sólo correspondencias entre puntos que alcancen un modelo de transformación común para establecer el movimiento de los objetos. Shafique y Shah [27] proponen un método utilizado en sistemas de video vigilancia en tiempo real que busca correspondencias de los mismos puntos en varios *frames*.

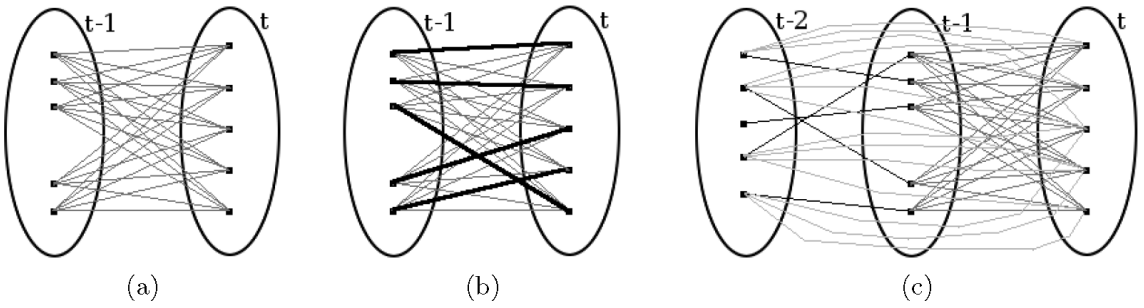

Figura 2.3. Correspondencia de puntos. Esta figura ha sido reproducida de [10].

- **Métodos estadísticos:** Estas técnicas, al igual que las anteriores, buscan correspondencias entre puntos de cada objeto en distintos instantes de tiempo y también utilizan las restricciones vistas en los métodos deterministas. La diferencia entre ambos métodos es la adopción de un cierto modelado en las medidas por parte de éste, cuantificado en forma de error. Broida y Chellappa [28] estiman el movimiento de los objetos con métodos estadísticos usando filtros de Kalman para lograr buenos resultados en escenarios con ruido. También existen estrategias basadas en RANSAC que consisten en seleccionar los conjuntos de puntos de consenso o inliers para estimar la transformación de movimiento [29].

El ***tracking de núcleo***, a diferencia del anterior, se basa en el seguimiento de la forma y apariencia de cada objeto. Los objetos son seguidos mediante el cálculo del movimiento del núcleo en *frames* consecutivos. Este tipo de tracking se puede dividir en dos métodos:

- **Tracking basado en patrones y modelos de apariencia:** Este modelo es muy utilizado debido a su sencillez y bajo coste computacional. Se basa en la utilización de patrones o características del objeto para prever el movimiento del objeto a través de las distintas imágenes, como por ejemplo, histogramas de color y gradientes para el seguimiento de caras [31]. Comaniciu propone Mean Shift [32], un método que utiliza histogramas de color del área de los objetos para buscar correspondencias entre *frames* y así estimar el modelo de transformación.
- **Tracking mediante modelos de apariencia multivista:** Consiste en aumentar la información de un objeto, realizando un aprendizaje previo, utilizando múltiples vistas del mismo. Con esto, el sistema será robusto ante problemas como cambios de orientación o apariencia del objeto. Avidan [33] presenta un método de seguimiento de objetos integrando *Support Vector Machine* en clasificadores de flujo óptico. También, Back y Jepson [23] proponen un algoritmo para calcular la transformación afín utilizando autovectores. El seguimiento se realiza mediante la estimación de los parámetros afines iterativamente hasta que, la diferencia entre la imagen de entrada y la imagen proyectada es mínima.

Por último, ***el tracking basado en siluetas*** es muy utilizado en el seguimiento de objetos con formas complejas, como manos, cabeza u hombros, que no pueden ser descritos de manera eficiente por formas geométricas simples. Se cataloga en dos grupos:

- **Tracking de siluetas**: Utiliza como patrones de búsqueda siluetas complejas. Huttenlocher [34] presenta un método caracterizado por descomponer la transformación del objeto en dos componentes, el movimiento y los cambios de forma. Realiza el seguimiento calculando la similitud entre patrones mediante la distancia Hausdorff.

- **Tracking de contorno**: Esta técnica intenta seguir el objeto adaptando iterativamente la transformación de su silueta. Peterfreund [35] propone un algoritmo para el seguimiento de objetivos utilizando filtros de Kalman para extraer la información definida por el contorno del objeto. Isard [36] presenta un método de seguimiento de contorno basado en filtros de partículas, conocidos como algoritmo de condensación.

Tabla 2.3. Métodos de seguimiento de objetos.

Tipo	Método	Referencia	Descripción
Tracking por puntos	Métodos deterministas	Salari y Sethi [25]	Algoritmo de seguimiento de objetos que soluciona problemas como la oclusión y puntos de interés de baja invarianza.
		Veenman, Reinders y Backer [26]	Seguimiento de objetos utilizando correspondencias de puntos.
		Shafique y Shah [27]	Seguimiento de objetos en movimiento.
	Métodos estadísticos	Broida y Chellappa [28]	Estimación del movimiento de objetos en secuencias con ruido.
		Lu, Dai y Hager [29]	Seguimiento de cara con cambios de expresión.
		Strandmark y Gu [30]	Seguimientos de múltiples objetos en vídeos grabados por cámaras en movimiento.
Tracking de núcleo	Tracking basado en patrones y modelos de apariencia	Birchfield [31]	Sistema de seguimiento de caras en tiempo real.
		Comaniciu y Meer [32]	Seguimiento de objetos utilizando histogramas de color (Mean Shift).
	Tracking mediante modelos de apariencia multivista	Avidan [33]	Seguimiento de vehículos en secuencias de vídeo con cámaras en movimiento.
		Black y Jepson [23]	Seguimiento y reconocimiento de gestos con manos.
Tracking basado en siluetas	Tracking de siluetas	Huttenlocher, Noh y Rucklidge [34]	Seguimiento de las siluetas de objetos.
	Tracking de contorno	Peterfreund [35]	Seguimiento del contorno de objetos.
		Isard y Blake [36]	Seguimiento del contorno de objetos.

2.2 Tracking de objetos basado en puntos de interés

En el contexto de este PFC se plantea un sistema que utiliza la localización y descripción de puntos de interés para la detección y seguimiento de objetos. Por ello, vamos a detallar los algoritmos detectores y descriptores de puntos de interés más utilizados en el estado del arte.

Hoy en día, muchos sistemas de seguimiento de objetos proponen la localización de puntos y la descripción de sus características para posteriormente encontrar correspondencias entre los puntos. Aunque los sistemas difieren, generalmente todos siguen las pautas que se citan a continuación:

- **Capturar un nuevo frame** de la secuencia de vídeo.
- **Detectar los puntos de interés** en la escena.
- **Describir las características** de cada uno de los puntos candidatos.
- **Buscar correspondencias** entre los descriptores detectados en el *frame* actual y los descriptores del objeto encontrados en *frames* anteriores.
- **Estimar el movimiento** del objeto utilizando la información de estas correspondencias. Por lo general, el número de correspondencias es mayor que los grados de libertad que se necesitan para estimar el movimiento. Esto permite eliminar correspondencias atípicas para mejorar la estimación utilizando algún método tipo RANSAC [37].

2.2.1 Detección de puntos de interés

Debido a la alta carga computacional para describir un píxel y al gran número de píxeles que hay en una imagen, es necesario utilizar un conjunto de puntos acotado para realizar el seguimiento. Por esta razón nacen los algoritmos de detección de puntos de interés. Estos puntos pueden ser, por ejemplo, esquinas y bordes de objetos. Shi y Tomasi [38] definen los puntos de interés como los que tienen las características adecuadas para realizar el mejor seguimiento. Esto significa que los puntos de interés deben ser repetitivos para que el algoritmo de seguimiento detecte dichos puntos en el máximo número de imágenes consecutivas.

A continuación se van a describir los principales detectores utilizados para la localización de puntos de interés.

Detector Harris

El detector de esquinas Harris [39] es uno de los detectores de puntos de interés más utilizados. Los puntos de interés detectados son invariantes frente a cambios de escala, rotación, iluminación y ruido de la imagen. Este detector calcula una subventana de tamaño $n \times n$ para cada punto de interés en la posición (x, y) dando lugar a la matriz $C(x, y)$.

$$C(x,y) = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \tag{2.4}$$

siendo I_x e I_y los gradientes en dirección horizontal y vertical de la subventana de la imagen. Posteriormente, se calculan los autovalores λ_1 y λ_2 de la matriz $C(x, y)$. Por último, siguiendo la ecuación *2.5*, se calcula la autocorrelación R que tendrá picos cuando los dos autovalores sean altos, indicando que se trata de una esquina.

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2) \quad (2.5)$$

Detector Shi-Tomasi

Este detector de esquinas está basado en el detector Harris. Como en Harris, utiliza la función de autocorrelación para medir cambios locales de la imagen. Shi y Tomasi [38] calculan el gradiente de intensidad de cada píxel de la imagen, el cual proporciona información sobre la no uniformidad en los niveles de gris. Los puntos que presentan cambios bruscos en los gradientes, tanto en dirección horizontal como en dirección vertical, corresponden a bordes de objetos.

Diferencia de Gaussianas (DoG)

Este método localiza puntos de interés utilizando operadores multiescala, y constituye el detector de puntos del algoritmo SIFT [40]. Su alta carga computacional hace que el uso de este detector sea inviable para sistemas en tiempo real. Se expone de manera detallada en el punto *2.2.3*.

Detector Fast Hessian

Es el detector de puntos de interés del algoritmo SURF [41]. Utiliza una aproximación del determinante de la matriz Hessiana para mejorar el tiempo de cálculo. Es el algoritmo de localización de puntos de interés que utilizaremos en este proyecto. Por su importancia en este PFC lo describimos en detalle en la sección *2.2.4*.

Detector FAST

El detector FAST [42] identifica localizaciones de puntos de interés de manera muy rápida y eficiente. Está orientado para la utilización en aplicaciones en tiempo real.

Detector de Harris-Laplace

Los puntos de interés del detector Harris-Laplace [24] son invariantes frente a cambios de rotación y escala. Los puntos son detectados mediante una función de Harris adaptada a la escala seleccionada por el operador Laplaciano. La escala seleccionada determina el tamaño de la región de interés. Este método detecta características localizadas espacialmente de

forma más precisa, lo que convierte a este detector en una de las opciones más recomendables para aplicaciones de reconstrucción y localización de características [43].

Detector SUSAN

El detector de esquinas SUSAN [44] emplea un enfoque morfológico para la evaluación de los puntos. Este método está orientado al procesamiento de imágenes de bajo nivel aplicando una máscara circular sobre cada píxel. Los puntos de interés se definen en función de la similitud de luminosidad entre los píxeles situados en la máscara circular y el píxel central. SUSAN se utiliza en aplicaciones de reconocimiento de objetos.

2.2.2 Descripción de características

Una vez que hemos localizado los puntos de interés de la imagen, describimos sus características. Un descriptor está definido por las especificaciones locales del punto como el brillo, el contraste o la textura. Idealmente, un descriptor es único para el mismo punto en cualquier instante de una secuencia. Para ello, algunos descriptores consiguen invarianza frente a la escala, rotación, traslación y cambios de iluminación.

Como contrapunto, es necesario mencionar que, la principal flaqueza de estas técnicas suele ser la complejidad de cálculo de los descriptores que representan cada uno de los puntos detectados, pero debido a los avances computacionales se ha convertido en una técnica estándar para el seguimiento de objetos.

Descriptor SIFT

SIFT [40] es un algoritmo de descripción de puntos locales en imágenes. Es capaz de definir puntos de forma que sean invariantes a la escala, rotación y traslación, y parcialmente a la orientación. Describiremos el método en detalle en la sección *2.2.3*.

Descriptor SURF

El detector de SURF [41] inspirado en SIFT, utiliza respuestas de Haar o *wavelets* para describir los puntos. Es la técnica de descripción de puntos de interés que utilizaremos en este proyecto. Por la relevancia que adquiere en este PFC definiremos sus características en el apartado *2.2.4*.

Descriptor BRISK

Es un método de detección, descripción y búsqueda de correspondencias de puntos clave [45]. Este algoritmo mejora en un orden de magnitud el cálculo del descriptor SURF. Esta

aceleración se debe a la combinación del algoritmo FAST para la detección de puntos y la construcción de un descriptor binario a partir de las comparaciones de intensidad, conseguidas al muestrear cada zona vecina de los puntos localizados.

Descriptor BRIEF

BRIEF [46] propone el uso de vectores binarios para formar el descriptor de puntos. Es altamente discriminativo, incluso si el descriptor está formado por un número pequeño de bits. Además, utiliza como método para encontrar correspondencias la distancia Hamming. Así, mejora el tiempo de procesamiento tanto en la creación del descriptor como en la búsqueda de correspondencias.

Descriptor ORB

ORB [47] es un descriptor binario basado en BRIEF, que mejora el tiempo de cálculo del descriptor SIFT en más de dos órdenes de magnitud. Además, aporta invarianza a la rotación y es resistente al ruido.

Descriptor FREAK

FREAK [48] se inspira en el sistema visual humano. Su descriptor está formado por vectores binarios que se calculan mediante la comparación de intensidades de imagen, sobre un patrón de muestreo de la retina. Se caracterizan por un consumo pequeño de memoria y un aumento de la robustez frente a otros descriptores como SIFT, SURF o BRISK. Son muy utilizados en aplicaciones para móviles.

En los siguientes apartados, *2.2.3* y *2.2.4*, vamos a profundizar en los detectores y descriptores que han sido considerados de mayor relevancia para este proyecto: los algoritmos SIFT [40] y SURF [41].

2.2.3 SIFT

SIFT [40] es un algoritmo capaz de detectar puntos de interés estables en una imagen. Este algoritmo fue desarrollado por Lowe [40] para el reconocimiento de objetos, realizando correspondencias entre puntos de interés de diferentes imágenes para posteriormente asociarlos. Los puntos locales escogidos por SIFT son invariantes frente a diferentes transformaciones como escala, rotación, traslación, ruido de la imagen y parcialmente invariantes frente a cambios de iluminación.

Este detector se caracteriza principalmente por la gran cantidad de descriptores estables que genera.

El algoritmo SIFT se define en cuatro etapas principales que detallaremos a continuación.

- **Detección de extremos**

En la primera fase, el algoritmo realiza una búsqueda de extremos sobre todos los píxeles de la imagen. Los extremos localizados serán invariantes frente a los cambios de orientación y escalado, y serán candidatos a ser *keypoints* o puntos de interés.

Como primer paso para detectar los puntos de interés se crean varios filtros Gaussianos $G(x, y, \sigma)$ con una variación de escala σ. Cada filtro Gaussiano se convoluciona con la imagen original $I(x, y)$, dando lugar a réplicas de la imagen con una diferencia de escala (scale-space), como mostramos en la primera imagen de la figura 2.4. Definimos scale-space como $L(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.6}$$

siendo (x, y) el eje de coordenadas y σ el factor de escala.

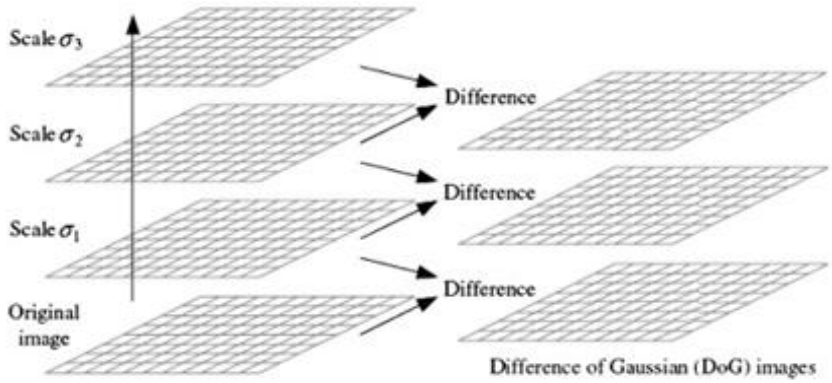

Figura 2.4. Imágenes espaciadas en escala. Diferencia de Gaussianas. Imagen extraída de [40].

Consecutivamente, para localizar los máximos y mínimos locales, se utiliza la diferencia de Gaussianas (DoG). La función $DoG(x, y, \sigma)$ se obtiene de la diferencia entre dos scale-space adyacentes, separadas por una escala $(k - 1)\sigma$, como mostramos a la derecha de la figura 2.4:

$$DoG(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.7}$$

siendo k la diferencia de escalas.

Por último, se detectan los puntos máximos y mínimos de la DoG comparando cada punto con sus ocho puntos vecinos y sus nueve vecinos en las escalas adyacentes, como vemos en la figura 2.5. Si el valor de la DoG del píxel es el máximo o mínimo en comparación con el valor de la DoG de sus 26 píxeles vecinos, el punto es un candidato a *keypoint*.

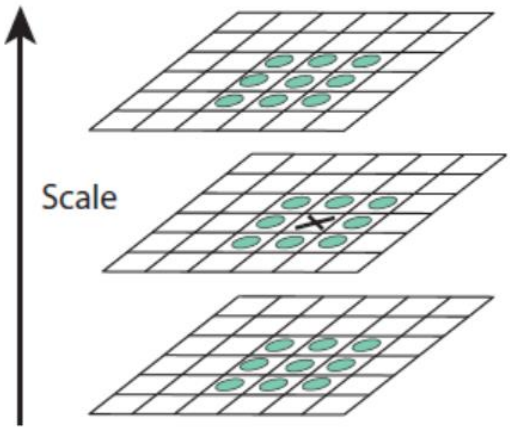

Figura 2.5. Píxeles con los que comparar cada punto en búsqueda de máximos o mínimos. Imagen extraída de [40].

A continuación, una vez que hemos encontrado los candidatos a *keypoints* en la imagen, almacenamos la localización y escala de los puntos de interés que son estables.

- **Localización de los puntos clave**

Cuando hemos localizado los candidatos a *keypoints* procedemos a realizar un estudio sobre la estabilidad de los mismos. Aquellos puntos que estén situados sobre los bordes, o tengan bajo contraste, son muy vulnerables al ruido y serán descartados. Para localizar los puntos inestables se utilizan los siguientes criterios:

- Los puntos cuyo valor de la DoG sea inferior a un umbral determinado, se eliminan de la siguiente etapa. Se considera que estos puntos tienen un bajo contraste y no son lo suficientemente estables.
- También se descartan los puntos localizados sobre bordes. Un *keypoint* situado sobre un borde contiene una alta inestabilidad incluso con pequeños ruidos. Además, su función DoG tiene una gran respuesta en dirección perpendicular del borde pero muy pobre en dirección del borde. Para realizar estos cálculos se utiliza el Hessiano y se limita la relación entre la respuesta paralela y perpendicular al borde. La proporción máxima utilizada es de 10.

Una vez descartados los puntos inestables, tendremos localizados el conjunto total de puntos de interés en la imagen.

- **Asignación de orientación**

En esta etapa, cada *keypoint* adquiere invarianza frente a la rotación mediante la asignación de una orientación dominante. Para esto, debemos calcular la magnitud $m(x, y)$ y

la orientación $\theta(x, y)$ del gradiente del keypoint y de los píxeles alrededor en una región vecina a través de las siguientes ecuaciones:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.8)$$

$$\theta(x,y) = \arctan \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \quad (2.9)$$

utilizando para L la escala del *keypoint* actual. Con las magnitudes y orientaciones del *keypoint* y sus puntos vecinos se crea un histograma. El histograma tiene 36 franjas de 10 grados cada una que cubren los 360 grados posibles. Para crear el histograma se pondera cada punto en función de la magnitud y de la distancia al *keypoint* con una gaussiana. La franja del histograma con un valor más alto corresponde con la dirección dominante del gradiente. Puede existir más de una dirección dominante, ya que, todas las franjas con un valor superior al 80% del valor de la magnitud principal se considerarán como orientaciones dominantes. Los puntos que tengan más de una dirección dominante poseerán una mayor estabilidad.

- **Generación de los descriptores de los puntos clave**

Una vez que se ha asignado para cada punto de interés una posición, una escala y una orientación dominante, se generará un descriptor que contenga información de su región vecina utilizando histogramas de orientaciones. La creación del descriptor se realiza con los datos previamente calculados en la fase de asignación de orientación.

Para cada punto en una región de 16×16 píxeles alrededor del *keypoint*, se calcula su magnitud del gradiente y se pondera por una Gaussiana. Esta región, a su vez, es dividida en subregiones de 4×4 píxeles. Por cada subregión se crea un histograma que contiene la suma de las magnitudes ponderadas, en función de la dirección de cada píxel, en 8 franjas proporcionales a 45 grados, como mostramos en la figura 2.6. Así obtenemos 16 histogramas por cada *keypoint*.

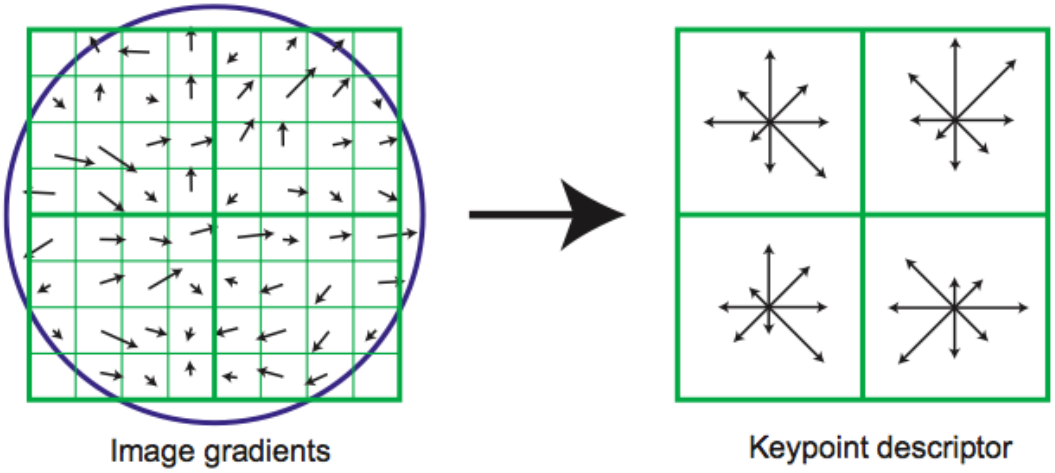

Figura 2.6. Matriz 2x2 del descriptor calculado a partir de un conjunto de 8x8 píxeles alrededor del punto de interés. Imagen extraída de [40].

El descriptor de cada *keypoint* está formado por un vector que tiene los valores de las 8 direcciones de cada uno de los 16 histogramas calculados. Así tenemos un vector de características de 128 elementos.

Por último, se tiene para cada punto de interés detectado la localización (x, y) en la imagen original, una orientación dominante, una escala y un vector de características asociado a su región vecina.

2.2.4 SURF

El algoritmo SURF [41], desarrollado por Herber Bay, es un detector y descriptor de puntos característicos. Se considera una evolución de SIFT [40] por estar inspirado en él. Su característica diferenciadora con SIFT es su mejora en la velocidad de cálculo gracias a que, en la generación de los descriptores, reduce la dimensión de los vectores y la complejidad en el cálculo, mientras que los puntos detectados continúan siendo suficientemente característicos e igualmente repetitivos.

A continuación se describe cada etapa para la creación de los descriptores SURF.

- **Detección de puntos de interés**

La primera etapa del descriptor SURF, análogamente a la del descriptor SIFT, consiste en la búsqueda de puntos característicos. A diferencia de SIFT, hace uso de la matriz Hessiana para la localización y escala de los puntos. Concretamente, utiliza el valor de una aproximación básica del determinante de la matriz Hessiana, consiguiendo un mayor rendimiento y precisión. Además, este detector, a diferencia de otros métodos, utiliza el mismo valor de este determinante para elegir la posición de los puntos (x, y) y su escala σ.

La matriz Hessiana $H(x,\sigma)$, dado un punto $p = (x,y)$ de la imagen $I(x,y)$, se define como:

$$H(x,\sigma) = \begin{pmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{pmatrix} \quad (2.10)$$

donde $L_{xx}(p,\sigma)$ representa la convolución de la derivada parcial de segundo orden de la Gaussiana $\frac{\partial 2}{\partial \mathrm{p}2}\, g(\sigma)$ con la imagen $I(x,y)$ en el punto $p = (x,y)$. Se realiza el mismo proceso para obtener $L_{xy}(p,\sigma)$ y $L_{yy}(p,\sigma)$.

Para aumentar la velocidad en el cálculo del espacio escala, el descriptor SURF [41] utiliza una aproximación a los filtros Gaussianos, los filtros de caja. Estos filtros aproximan las derivadas parciales de segundo orden de las gaussianas y pueden ser calculados de manera muy eficiente utilizando imágenes integrales.

Una imagen integral $I_{\sum}(p)$ de un punto $p = (x,y)^T$ representa la suma de todos los píxeles del rectángulo dentro de la imagen $I(x,y)$ formado por el origen y el punto p.

$$I_{\sum}(p) = \sum_{i=0}^{i\leq x} \sum_{j=0}^{j\leq y} I(i,j) \quad (2.11)$$

Una vez que se ha calculado la imagen integral, para obtener la suma de intensidades de un área rectangular, como el mostrado en la figura 2.7, se necesitan siempre tres operaciones. De esta forma, el tiempo necesario para el cálculo de las operaciones de convolución es independiente del tamaño del área rectangular. Un ejemplo de esta operación podría ser, teniendo los puntos de la figura 2.7:

$$I_{\sum}(área) = I_{\sum}(p4) + I_{\sum}(p1) - I_{\sum}(p2) - I_{\sum}(3) \quad (2.12)$$

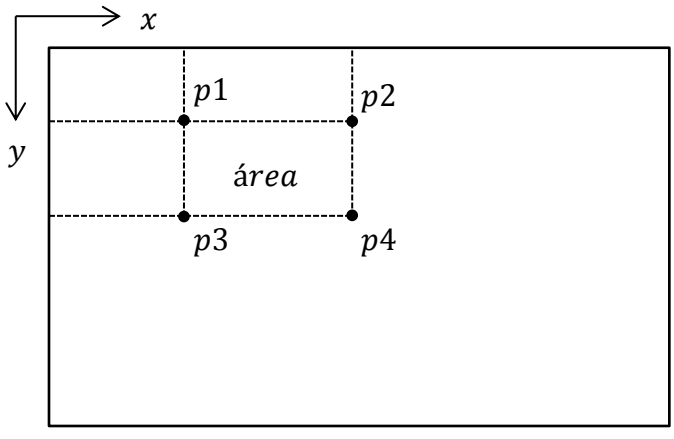

Figura 2.7. Representación de un área rectangular respecto a la imagen integral.

Una ventaja de la utilización de filtros de caja e imágenes integrales, con respecto al detector SIFT, es que en la formación del espacio escala no es necesario aplicar el mismo filtro iterativamente a la salida de una imagen escalada previamente, sino que se pueden aplicar

filtros de cualquier tamaño sobre la imagen original a la misma velocidad. De este modo, en vez de reducir el tamaño de la imagen como en SIFT, aumentamos el tamaño del filtro, como se ve en la figura 2.8.

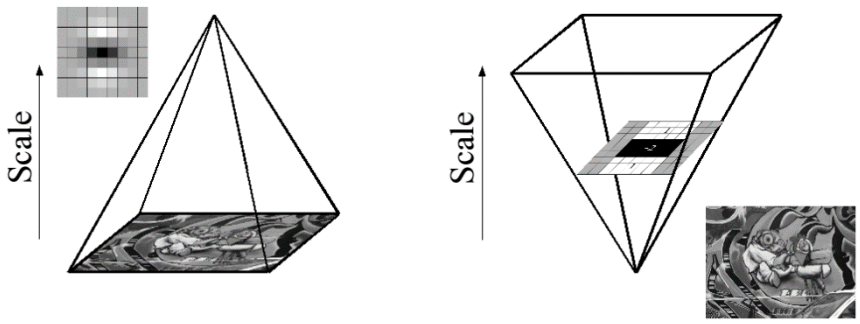

Figura 2.8. SIFT reduce el tamaño de la imagen para formar la diferencia de escalas. SURF aumenta el tamaño del filtro para formar la diferencia de escalas. Imagen extraída de [41].

A continuación, definimos el cálculo de los determinantes de la matriz Hessiana que localiza la escala del punto como:

$$Det\left(H_{aprox}\right) = D_{xx}D_{yy} - (\omega\, D_{xy})^2 \tag{2.13}$$

siendo D_{xx}, D_{yy} y D_{xy} las aproximaciones de las derivadas parciales, calculadas previamente mediante los filtros de caja e imágenes integrales, y ω el peso para poder balancear el determinante de la matriz Hessiana. En la figura 2.9 mostramos una representación de las derivadas parciales de segundo orden de un filtro gaussiano y las aproximaciones de estas derivadas parciales implementadas por SURF.

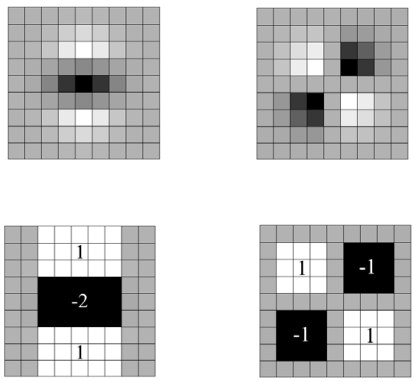

Figura 2.9. Derivadas parciales de segundo orden de un filtro gaussiano. Aproximación utilizada en SURF. Imagen extraída de [41].

Por último, una vez que hemos calculado el determinante de la aproximación de la matriz Hessiana se obtienen las localizaciones y escalas de los puntos de interés. De manera análoga a SIFT, los puntos de interés serán los que cumplan la condición de máximo valor de la aproximación con respecto a sus 8 píxeles vecinos, sus 9 píxeles vecinos de una escala mayor y sus 9 píxeles vecinos de una escala menor, interpolando su valor con el de los píxeles cercanos para mejorar su exactitud. Así, se da por concluida la etapa de detección de puntos.

- **Asignación de orientación**

La asignación de orientación es similar a la de SIFT. En esta fase se asigna a cada uno de los *keypoints* encontrados en la etapa anterior una orientación dominante centrada en el punto otorgando al descriptor invarianza frente a la rotación.

El primer paso para la obtención de la orientación dominante consiste en calcular la Respuesta de Haar en las direcciones x e y (Figura 2.10). La región de interés de este cálculo es un área circular centrada en el *keypoint* y con un radio de $6s$, donde s es la escala en la que se ha localizado dicho punto.

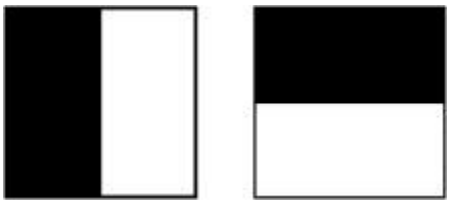

Figura 2.10. Funciones de la Respuesta de Haar utilizadas en SURF. Imagen extraída de [41].

Una vez calculadas las respuestas de Haar o *wavelets*, éstas son ponderadas por una gaussiana centrada en el punto de interés, para dar mayor importancia a las respuestas más cercanas al punto. Por último, se divide el área circular alrededor del punto de interés en 6 secciones, cubriendo un ángulo de $\frac{\pi}{3}$ cada una, como mostramos en la figura 2.11. La orientación dominante será la que sumando todas las direcciones dentro de cada sección tenga un vector de mayor longitud.

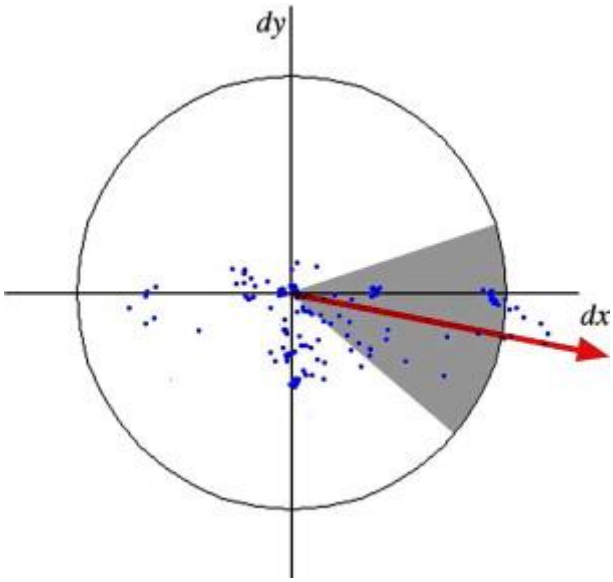

Figura 2.11. Asignación de la orientación de cada sector del área circular alrededor del punto, cubriendo un ángulo de $\frac{\pi}{3}$. Imagen extraída de [41].

- **Creación del descriptor**

Para finalizar, se crea el descriptor SURF. En esta última etapa se construye una región cuadrada de tamaño $20s$, centrada en el *keypoint* y orientada en función de la dirección dominante calculada en el apartado anterior. Esta región es dividida en subregiones de 4×4, y se calcula en cada una de ellas la respuesta de Haar de los puntos con una separación de muestreo de 5×5 en ambas direcciones.

Por simplicidad vamos a considerar a d_x y d_y las respuestas de Haar en las direcciones horizontal y vertical, respectivamente, referidas a la orientación del *keypoint*. Para asignar a las respuestas d_x y d_y una mayor robustez se ponderan por una gaussiana centrada en el punto, como en el apartado anterior.

A continuación, se suman las respuestas d_x y d_y de cada subregión. También, se suman los valores absolutos de las respuestas $|d_x|$ y $|d_y|$ de cada subregión para obtener información sobre los cambios de intensidad. Cada subregión se representa por el vector v con la siguiente estructura:

$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right) \tag{2.14}$$

Por lo que, se obtiene el descriptor SURF de 64 elementos dado por las 4 dimensiones del vector v de las 4×4 subregiones. Existe la posibilidad de aumentar el tamaño del descriptor a 128 elementos, como en SIFT.

2.2.5 Correspondencias de puntos y estimación de movimiento

En los sistemas de tracking por puntos de interés, se buscan correspondencias entre los *keypoints* detectados en el *frame* actual y los puntos almacenados. Los puntos almacenados pertenecen a *keypoints* detectados en *frames* anteriores. El objetivo de buscar estas correspondencias es calcular el modelo de transformación que siguen los puntos para estimar, utilizando ese modelo de transformación, la posición de los puntos en el *frame* actual.

Para la localización de correspondencias entre dos conjuntos de *keypoints* se utiliza el cálculo del valor de similitud, conocido como distancia o *score*. Para los descriptores SIFT y SURF se aplica la fórmula de la distancia euclídea entre dos descriptores para establecer el valor de similitud. Para otros descriptores, como los vectores binarios utilizados por ORB o BRIEF, se utiliza la distancia de Hamming.

En el caso particular de SURF, se calcula la distancia euclídea entre dos descriptores dx y dy, pertenecientes a dos puntos x e y. Para decidir si existe una correspondencia entre este par de puntos se establece que la distancia euclídea entre los dos vectores sea menor a k veces la distancia respecto al segundo vecino más cercano. La ecuación para calcular la distancia euclídea es la siguiente:

$$dist^2(dx, dy) = \sum_{i=1}^{64}(dx_i - dy_i)^2 \qquad (2.15)$$

Una vez calculados los valores de similitud entre todos los descriptores de los dos conjuntos de puntos, podremos asociar parejas de puntos que sean similares. Es importante destacar que para realizar esta operación se pueden fijar una serie de restricciones que disminuyan el coste computacional, como por ejemplo un umbral o un número máximo de descriptores por imagen.

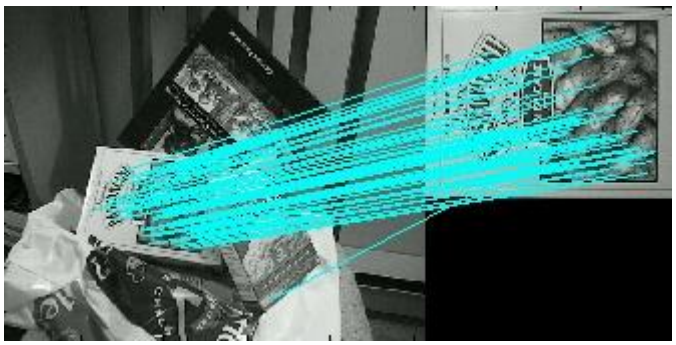

Figura 2.12. Matching entre *keypoints* de dos imágenes distintas. Imagen extraída de [40].

Como se puede ver en la figura 2.12, por cada punto de interés en la primera imagen, localizamos el punto con el que tiene una mayor similitud (menor distancia), en la segunda imagen. En ausencia de errores se puede determinar que estos dos puntos corresponden al mismo punto en el objeto.

Una vez que tenemos correspondencias entre pares de puntos de un objeto en dos imágenes consecutivas, podemos estimar el movimiento de éste. La eficacia de esta estimación es directamente proporcional a la calidad del conjunto de correspondencias y se ve afectada principalmente por dos tipos de errores.

- El matching detecta correspondencias incorrectas
- El matching no detecta correspondencias correctas.

Para evitar la dependencia a posibles correspondencias incorrectas, se utilizan algoritmos de alineamiento que estiman las transformaciones más robustas entre dos conjuntos de puntos y descartan las correspondencias con valores atípicos. En los algoritmos de alineamiento, cuando el número de puntos de interés supera los grados de libertad para definir un modelo de transformación, se es capaz de determinar el número de correspondencias válidas, *inliers*, y el número de correspondencias erróneas, *outliers*, dentro del conjunto total de puntos.

Los algoritmos de alineamiento calculan la transformación óptima utilizando aleatoriamente conjuntos reducidos de puntos y estimando su transformación particular. El modelo de transformación elegido es el que tenga una mayor puntuación en función del número de *inliers* que contenga. Los *inliers* y *outliers* se deciden mediante un umbral. La metodología habitual es utilizar el tipo de transformación que más se ajuste al movimiento del objetivo, y aplicar la transformación obtenida a todos los puntos del conjunto.

Las transformaciones más comunes son las siguientes (las mostramos en coordenadas homogéneas):

- **Traslación**

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

- **Escala**

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

- **Rotación**

$$R = \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

- **Cizalla (*shearing*)**

$$H = \begin{bmatrix} 1 & h_y & 0 \\ h_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

- **Transformación euclídea:** Traslación y rotación $(T * R)$.
- **Transformación de similitud:** Traslación, rotación y escala $(T * R * S)$.
- **Transformación afín:** Traslación, rotación, escala y cizalla (*shearing*) $(T * R * S * H)$.
- **Transformación proyectiva:** Traslación, rotación, escala, cizalla (*shearing*) y perspectiva.

Existen muchos sistemas que utilizan algoritmos de alineamiento para estimar el movimiento del objeto. En las secciones *2.2.6* y *2.2.7* describiremos en detalle los dos más relevantes para este proyecto, RANSAC [37] y RAMOSAC [30].

2.2.6 RANSAC

Random Sample Consensus, RANSAC [37], es el algoritmo de alineamiento más conocido. Consiste en realizar un ajuste robusto de un modelo, con un conjunto de datos S, que contiene valores atípicos u *outliers*.

Para comprender el funcionamiento de este algoritmo, primero ilustraremos un caso particular y posteriormente trataremos el caso general.

El funcionamiento del algoritmo, para ajustar una línea recta a un conjunto de puntos, es el siguiente:

1. Se seleccionan de forma aleatoria dos puntos del conjunto S. Estos dos puntos definen una recta. En la figura 2.13 se han seleccionado los puntos a y b que definen la recta dibujada.
2. Se comprueba la verosimilitud de esta recta con respecto al resto de puntos. Para ello, se calcula cuántos de estos puntos están a una distancia de la recta menor que un umbral previamente fijado (línea discontinua). Estos puntos se denominan *inliers* o puntos de consenso. Los puntos que quedan por encima del umbral se llaman *outliers*.
3. Esta selección aleatoria se repite un número de veces y la recta con más *inliers* se considera la recta robusta y se elige como modelo final.

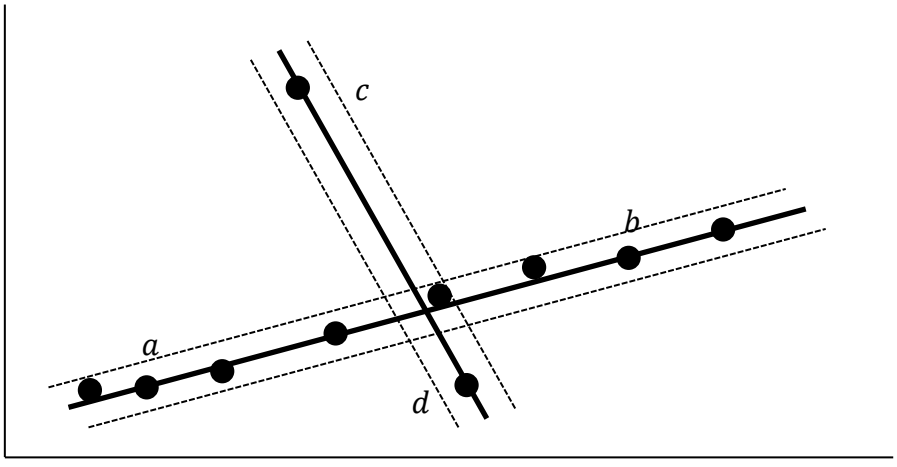

Figura 2.13. La recta robusta es la formada por los puntos a y b. Todo el conjunto de puntos, salvo dos *outliers* (c y d), forman el denominado conjunto de consenso.

En muchos casos en los que no es posible ajustar un modelo con dos únicos puntos, como en el caso anterior, se utiliza el caso general de RANSAC. El algoritmo general sigue los siguientes pasos [37]:

1. Se selecciona aleatoriamente una muestra s de puntos, del conjunto total S, y con este subconjunto se calcula el modelo de transformación.
2. Se determina el subconjunto de puntos S_i que está dentro del umbral $U1$ de distancia al modelo calculado en el punto anterior. El subconjunto S_i es el conjunto consenso y contiene los *inliers* del conjunto de puntos S.
3. Si el número de inliers del subconjunto S_i es mayor que un segundo umbral $U2$ se vuelve a estimar el modelo utilizando todos los puntos de S_i y se termina el algoritmo.
4. Si el número de inliers de S_i es menor que el umbral $U2$ se selecciona un nuevo subconjunto S_i y se repite el paso anterior.
5. Después de N intentos se selecciona el subconjunto S_i con mayor consenso. El modelo se recalcula utilizando todos los puntos del subconjunto S_i

En la figura 2.14 mostramos una corrección de *outliers* con RANSAC, aplicada a correspondencia de puntos. En azul podemos ver un conjunto de puntos en el instante de tiempo i, y en rojo sus correspondencias en el instante de tiempo $i + 1$. Mediante RANSAC calculamos el modelo de consenso (la línea azul muestra los *inliers* y la línea roja muestra los *outliers*), y aplicamos esa transformación a todos los puntos.

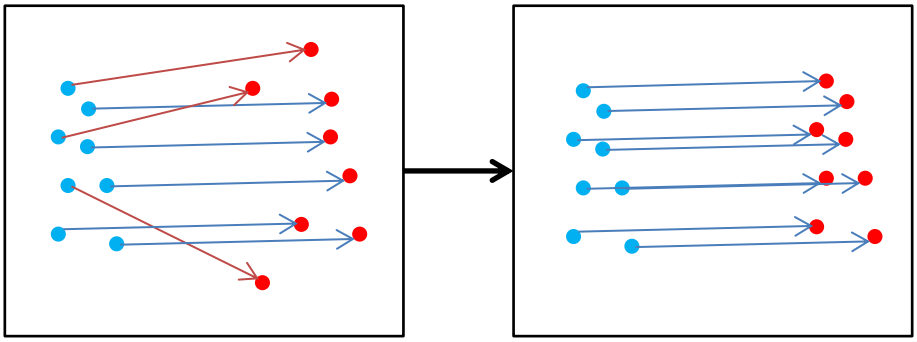

Figura 2.14. Correspondencias entre dos conjuntos de puntos. Transformación de todos los puntos siguiendo el modelo de consenso calculado por RANSAC.

2.2.7 RAMOSAC

En la literatura existen muchas variantes del método RANSAC [37] que consiguen aumentar su eficacia. Una de estas variantes es RAMOSAC [30]. Su principal ventaja frente a RANSAC es que utiliza diferentes tipos de transformaciones. Estos modelos de transformación son elegidos al azar consiguiendo tener un conjunto de modelos de diferente complejidad. La principal diferencia entre RANSAC y RAMOSAC es la ecuación de puntuación para elegir el mejor modelo de transformación. Mientras RANSAC utiliza únicamente el número de *inliers*, RAMOSAC añade dos nuevas variables. RAMOSAC asigna a cada modelo una probabilidad en función del mínimo número de puntos necesarios para calcular correctamente sus parámetros

y una probabilidad condicional a la transformación que sufrió el conjunto de puntos en el instante anterior. Así, junto al número de *inliers* del modelo, calcula la puntuación con la que se determina cuál es la transformación de consenso.

El algoritmo RAMOSAC [30] funciona del siguiente modo:

1. Se elige un modelo de transformación al azar.
2. Se elige un subconjunto de puntos s, aleatoriamente.
3. Se estima la transformación para el tipo elegido utilizando el subconjunto de puntos s.
4. Se evalúa la transformación resultante en función del número de inliers S_i, la probabilidad asignada al modelo elegido al azar y la probabilidad condicional.
5. Se repiten los pasos 1-4 N veces y se selecciona la transformación con la puntación más alta.

La elección de un modelo al azar, en cada iteración, permite que ningún modelo sea favorecido con respecto a otro.

Los modelos de transformación que RAMOSAC permite utilizar son los siguientes:

- Traslación.
- Transformación de similitud.
- Transformación afín.
- Transformación proyectiva.

El número mínimo de puntos necesarios para estimar los parámetros del modelo son 1, 2, 3 y 4 respectivamente. Si el número de puntos es mayor al mínimo se usa la técnica de mínimos cuadrados.

Para evaluar cada transformación se utiliza la siguiente ecuación, siendo Sc la puntuación de la transformación de RAMOSAC:

$$Sc = |C| + \log_{10} P(T|\lambda, p_{t-1}) + \varepsilon n_{min} \tag{2.20}$$

donde $|C|$ corresponde al número de *inliers* que siguen la transformación, ε es un factor de escala (ε = 0.1), n_{min} es el número de puntos necesarios para estimar correctamente el modelo y $P(T|\lambda, p_{t-1})$ es la probabilidad de que el objeto obtenga un movimiento mayor o igual al movimiento medio del objeto.

Para calcular la probabilidad $P(T|\lambda, p_{t-1})$ se parte de la suposición de que el movimiento de un objeto sigue la misma distribución en dos *frames* consecutivos. La ecuación para calcular dicha probabilidad es la siguiente:

$$P(T|\lambda, p_{t-1}) = e^{-\lambda\, dist(T|p_{t-1})} \tag{2.21}$$

donde $dist(T|p_{t-1})$ corresponde a la distancia entre la posición del polígono que delimita el objeto p entre el instante $t-1$ y la posición del polígono estimado $T(p_{t-1})$ en el instante t,

$$dist(T|p_{t-1}) = \sum_{k=1}^{n} \left\| p_{t-1}^{k} - T\left(p_{t-1}^{k}\right) \right\| \tag{2.22}$$

y el parámetro λ toma el valor de la media del movimiento que ha sufrido el objeto en *frames* anteriores. De esta manera nos aseguramos que, las transformaciones resultantes de grandes movimientos en comparación con el movimiento medio necesiten un alto número de puntos de consenso para no ser penalizadas, mientras que las transformaciones que provienen de pequeños movimientos se vean favorecidas. La transformación que tiene la puntuación más alta se selecciona como el modelo de transformación de consenso.

2.3 Fusión de información en sistemas de múltiples cámaras

El uso de múltiples cámaras en sistemas de vigilancia o análisis de eventos deportivos está muy extendido. Existen diversas aplicaciones en las que se utiliza la fusión de información de la escena, como pueden ser la resolución de oclusiones y la reconstrucción de trayectorias. Como en los sistemas de seguimiento en una cámara, el objetivo de los sistemas de seguimiento multicámara es fusionar los datos para detectar y seguir de forma más robusta recursivamente los objetos de interés que hay en la escena aumentando además el plano de visión.

El análisis de información en sistemas de múltiples cámaras se puede clasificar en tres tipos: basados en apariencia [49], basados en la geometría [50] y sistemas híbridos [51]. Además existe otro método de clasificación en función del orden de análisis de la información: fusionar la información antes del tracking [52] y fusionar la información después del tracking [53].

La fusión de datos es de gran utilidad para poder ampliar el escenario de una sola vista a un marco más amplio. Kettnaker y Zabih [54], siguiendo un enfoque probabilístico, reconstruyen trayectorias de objetos en sistemas con múltiples cámaras con áreas solapadas. El sistema requiere información previa sobre el escenario y la forma de moverse a través de él. Huang [55] presentó un método probabilístico para el seguimiento de vehículos que se mueven en una dirección, en un solo carril, utilizando dos cámaras espaciadas en una autopista. Dick [56] utiliza una aproximación estocástica para describir la relación entre las cámaras. Las correspondencias entre las cámaras se asignan en una fase de entrenamiento. Sheikh y Shah [57] presentan otro enfoque en el que detectan si dos trayectorias vistas en dos cámaras distintas corresponden al mismo objeto en función de un error de proyección. Wang [58] busca correspondencias entre trayectorias de objetos detectados en varias cámaras en función de su información temporal. Las trayectorias corresponden al mismo objeto si se solapan en el tiempo y corresponden a la misma actividad. Los sistemas de múltiples cámaras también han sido utilizados para solucionar problemas como la oclusión [51].

En este proyecto utilizaremos la fusión de información basada en la geometría de las cámaras después de realizar el tracking en cada cámara. Los métodos basados en geometría utilizan generalmente geometría epipolar, homografía o calibración de la cámara. En nuestro proyecto supondremos que los parámetros internos de las cámaras, parámetros intrínsecos y extrínsecos, no son conocidos, por lo que utilizaremos la homografía como método para relacionar las cámaras. Como veremos en los siguientes apartados, para adquirir la relación geométrica entre las cámaras hay que disponer de un conjunto de puntos físicos en el escenario visible desde todas las cámaras.

Tabla 2.4. Sistemas de seguimiento multicámara.

Orden de análisis	Fusión de información	Referencia	Descripción
Antes del tracking	**Apariencia**	Kang, Cohen y Medioni [49]	Detección y seguimiento de objetos en movimiento en sistemas multicámara.
Después del tracking	**Híbrido**	Du, Hayet, Piater y Verly [51]	Seguimiento de atletas en sistemas multicámara.
Antes del tracking	**Geometría**	Taj y Cavallaro [52]	Seguimiento de múltiples objetos en sistemas multicámara.
Después del tracking	**Híbrido**	Anjum y Cavallaro [53]	Fusión de trayectoria de objetivos en sistemas multicámara.
Después del tracking	**-**	Kettnaker y Zabih [54]	Reconstrucción de trayectorias de objetos en movimiento en sistemas multicámara.
-	**Apariencia**	Huang y Russell [55]	Reconocimiento de vehículos en una autopista utilizando dos cámaras.
Después del tracking	**Geometría**	Dick y Brooks [56]	Seguimiento de personas en tiempo real en sistemas multicámara.
Después del tracking	**Geometría**	Sheikh y Shah [57]	Correspondencia de trayectorias en sistemas multicámara.
Después del tracking	**-**	Wang [58]	Correspondencia de trayectorias utilizando métodos probabilísticos en sistemas multicámara.

2.3.1 Homografía

En geometría, se denomina homografía a toda transformación proyectiva que determina una correspondencia entre dos figuras geométricas planas, de forma que a cada uno de los puntos y las rectas de una de ellas le corresponden, respectivamente, un punto y una recta de la otra. La homografía tiene multitud de aplicaciones en el tratamiento de imágenes como son la reconstrucción 3D, la calibración de cámaras, la creación de panoramas o la ayuda en el tracking.

Normalmente, se utiliza la homografía para lograr una transformación perspectiva entre planos y así llevar la proyección de un objeto desde una vista a otra. En la figura 2.15 se puede observar un ejemplo de correspondencia de puntos físicos del escenario entre dos cámaras.

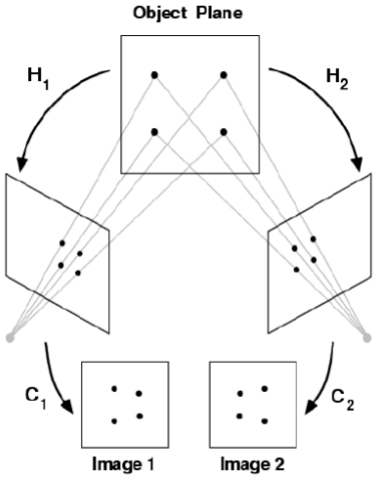

Figura 2.15. Mapeo de puntos entre dos vistas.

La homografía entre dos vistas se puede estimar con correspondencias de puntos entre las dos imágenes, de forma que obtenemos una matriz H que mapea puntos de una vista a otra:

$$\boldsymbol{p}' = H\boldsymbol{p} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.23}$$

donde $\boldsymbol{p}' = (x', y', 1)$ es un punto de la primera vista y $\boldsymbol{p} = (x, y, 1)$ es un punto de la segunda vista. Cada correspondencia de puntos genera dos ecuaciones,

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{2.24}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.25}$$

donde h_{ij} corresponden a los elementos que componen la matriz H. Así, con cuatro correspondencias de puntos conseguimos ocho ecuaciones, con las que se obtienen todos los elementos de la matriz H (nótese que h_{33} es igual a 1). Normalmente, se utilizan más de cuatro correspondencias para tener un sistema sobredimensionado.

En este proyecto vamos a utilizar este método para la fusión de información entre dos cámaras. Para calcular la homografía vamos a utilizar el algoritmo DLT 2D descrito a continuación.

2.3.1.1 Algoritmo DLT 2D

Direct Linear Transformation (DLT), es un algoritmo adecuado para calcular transformaciones proyectivas entre dos conjuntos de datos de al menos 4 puntos, aunque en este PFC se va a utilizar para mapear puntos entre dos cámaras.

La transformación proyectiva u homografía H que mapea el conjunto de puntos $\boldsymbol{p}$ de la vista 1 a puntos $\boldsymbol{p'}$ de la vista 2 es una matriz 3×3 y cumple la siguiente ecuación $\boldsymbol{p'_i} = H\boldsymbol{p_i}$, para todo i. La homografía también puede representarse en términos de un vector de producto vectorial $\boldsymbol{p'_i} \times H\boldsymbol{p_i} = \boldsymbol{0}$, permitiendo una simple solución.

Si se expresa la j-ésima fila de la matriz H como $\boldsymbol{h^{jT}}$ para $j = 1,2,3$, se puede escribir:

$$H\boldsymbol{p_i} = \begin{pmatrix} \boldsymbol{h^{1T}p_i} \\ \boldsymbol{h^{2T}p_i} \\ \boldsymbol{h^{3T}p_i} \end{pmatrix} \tag{2.26}$$

y definiendo $\boldsymbol{p'_i} = (x'_i, y'_i, w'_i)^T$, el producto vectorial puede expresarse como:

$$\boldsymbol{p'_i} \times H\boldsymbol{p_i} = \begin{pmatrix} y'_i\boldsymbol{h^{3T}p_i} - w'_i\boldsymbol{h^{2T}p_i} \\ w'_i\boldsymbol{h^{1T}p_i} - x'_i\boldsymbol{h^{3T}p_i} \\ x'_i\boldsymbol{h^{2T}p_i} - y'_i\boldsymbol{h^{1T}p_i} \end{pmatrix} \tag{2.27}$$

Teniendo en cuenta que $\boldsymbol{h^{jT}p_i} = \boldsymbol{p_i^T h^j}$ para $j = 1,2,3$ e igualando a cero se consigue un conjunto de tres ecuaciones que se puede expresar de la siguiente manera:

$$\begin{bmatrix} \mathbf{0^T} & -\mathrm{w'_i}\mathbf{p_i^T} & \mathrm{y'_i}\mathbf{p_i^T} \\ \mathrm{w'_i}\mathbf{p_i^T} & \mathbf{0^T} & -\mathrm{x'_i}\mathbf{p_i^T} \\ -\mathrm{y'_i}\mathbf{p_i^T} & \mathrm{x'_i}\mathbf{p_i^T} & \mathbf{0^T} \end{bmatrix} \begin{pmatrix} \mathbf{h^1} \\ \mathbf{h^2} \\ \mathbf{h^3} \end{pmatrix} = 0 \tag{2.28}$$

Por cada par de correspondencias, se obtiene un conjunto de ecuaciones con la forma $A\boldsymbol{h} = 0$, donde A es una matriz 3×9 y $\boldsymbol{h}$ es un vector de 9 elementos. De las 3 ecuaciones que se obtienen sólo 2 son linealmente independientes. Por lo tanto, para el cálculo de la matriz H se puede ignorar la tercera ecuación, convirtiendo el conjunto de ecuaciones en:

$$\begin{bmatrix} \boldsymbol{0^T} & -w'_i\boldsymbol{p_i^T} & y'_i\boldsymbol{p_i^T} \\ w'_i\boldsymbol{p_i^T} & \boldsymbol{0^T} & -x'_i\boldsymbol{p_i^T} \end{bmatrix} \begin{pmatrix} \boldsymbol{h^1} \\ \boldsymbol{h^2} \\ \boldsymbol{h^3} \end{pmatrix} = 0 \tag{2.29}$$

donde A es una matriz 2×9. Así, por cada par de correspondencias se obtienen 2 ecuaciones linealmente independientes. Para resolver la matriz H se necesitan al menos 4 pares de puntos, con lo que obtenemos 8 ecuaciones. El último término de la matriz H corresponde a $h_{33} = 1$. Si tenemos más de 4 correspondencias conseguimos un sistema de ecuaciones sobredimensionado reduciendo los posibles errores que se puedan producir.

2.3.2 Geometría epipolar

Otro método de fusión de información basada en la geometría utilizado para el tracking es la geometría epipolar. La geometría epipolar expresa la relación geométrica entre dos proyecciones de un mismo punto físico. Una de las principales aplicaciones que podemos darle a la geometría epipolar es la de determinar si la proyección p en una vista, y la proyección p' en otra vista, corresponden al mismo punto en la escena P.

Independientemente de lo que forma la escena tridimensional, todo par de imágenes tiene unas restricciones geométricas. Una de estas restricciones es la restricción epipolar, que ayuda a limitar el espacio de búsqueda de correspondencia de puntos. Esta restricción dice que todos los planos epipolares originan líneas horizontales al cortarse con los planos de las imágenes reduciendo el espacio de búsqueda a una dimensión. Como vemos en la figura 2.16, cada punto de la primera vista pertenece a una línea en la segunda vista llamada línea epipolar, para p es la recta l' que forman los puntos p' y e'. La restricción epipolar se puede representar algebraicamente mediante la matriz fundamental F.

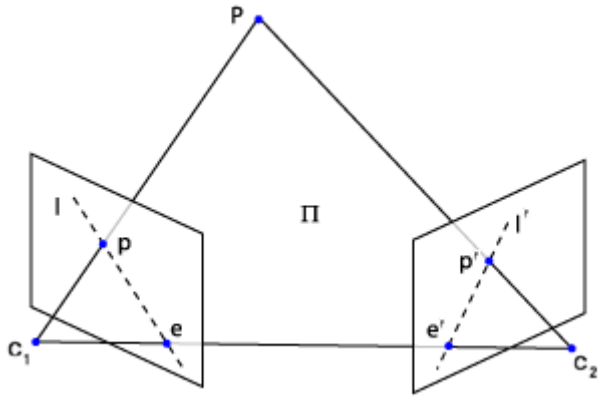

Figura 2.16. Geometría epipolar.

Los elementos básicos de la geometría epipolar son los siguientes:

a) El **plano epipolar** es el plano que contiene un punto P del escenario y los centros de las cámaras. En la figura 2.16 corresponde al plano π.
b) La **línea epipolar** es la intersección entre el plano epipolar con los planos de la imagen. En la figura 2.16, para la cámara 1 corresponde a l y para la cámara 2 a l'.
c) Los **epipolos** son los puntos de la intersección entre la imagen de las cámaras y la línea que une los centros de la cámara con un punto P del escenario.

3 DISEÑO

En este capítulo se va a describir en detalle la metodología seguida en el desarrollo del proyecto.

El objetivo de este proyecto es crear un sistema detector y *tracker* de movimiento de objetos. Tanto el detector como el módulo de seguimiento de objetos utilizarán puntos característicos para definir los fotogramas. Hemos elegido puntos característicos frente a otros métodos para evaluar el funcionamiento de estos algoritmos.

Tras el estudio del estado del arte hemos decidido utilizar el método SURF para localizar y describir los puntos de interés, la distancia euclídea para encontrar correspondencias entre puntos, RAMOSAC como método de estimación de movimiento de los objetos y homografía proyectiva para fusionar la información de los objetos detectados entre varias cámaras. El motivo de estas decisiones se explicará en detalle a lo largo de este capítulo.

El sistema que vamos a desarrollar, **M**oving **O**bject **DE**tection and **T**racking (**MODET**), tiene como objetivo la detección y seguimiento de objetos de interés en movimiento en distintos escenarios. Estos objetos pueden ser vehículos, bicicletas e incluso peatones en escenas de tráfico o coches de juguete en entornos controlados. Para que los objetos sean detectados se tienen que cumplir ciertos requisitos como que tengan los suficientes puntos de interés para definirlos.

El algoritmo está destinado a abordar casos donde los objetos puedan tener movimientos no uniformes, por ejemplo objetos con distintas velocidades, cambios de trayectorias o posibles oclusiones.

MODET está diseñado para entornos de grabación con cámaras estáticas, aunque como mostraremos en las pruebas también se logran buenos resultados con cámaras en movimiento. Las pruebas se realizarán en entornos, como máximo, de dos cámaras, así que de ahora en adelante hablaremos de escenarios de una o dos cámaras.

La arquitectura de **MODET** se divide en tres grandes etapas, precedidas de la lectura de cada fotograma de la secuencia, como se puede observar en la figura 3.1.

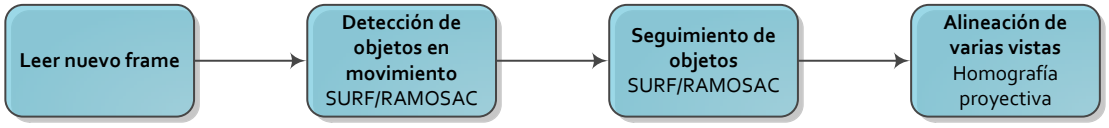

Figura 3.1. Etapas de MODET.

Las fases, que se repiten recursivamente en cada *frame* de la secuencia de vídeo, son las siguientes:

- ***Detección de objetos en movimiento***: Localizar los objetos en movimiento que entran en la escena.
- ***Seguimiento de objetos***: Seguir los objetos detectados por la fase anterior utilizando puntos característicos.
- ***Alineación de varias vistas***: Fusionar la información de los distintos objetos almacenados entre las dos cámaras.

En la primera parte, describiremos la localización de objetos en movimiento que se ha diseñado. Para detectar las regiones en movimiento, el detector utiliza la correspondencia de puntos característicos entre *frames* adyacentes. Una vez que se obtienen las correspondencias y el movimiento de cada par de puntos, se decide qué puntos corresponden al fondo y cuáles corresponden a regiones en movimiento o primer plano. A continuación, se realiza una segmentación y agrupamiento de los puntos de primer plano tomando sólo los pertenecientes a objetos en movimiento y representando el área de cada objeto detectado.

En la segunda parte del algoritmo estimaremos el movimiento de los objetos almacenados en los siguientes *frames* de la secuencia de vídeo. Cada objeto estará definido por un conjunto de descriptores, los cuales serán invariantes en escala, rotación y traslación usando el descriptor SURF. Los puntos de interés de cada objeto, son almacenados y actualizados para cada uno de los *frames* en los que aparece. Con este conjunto de información, además, podremos estimar el mejor modelo de transformación a través del algoritmo de alineamiento RAMOSAC.

Por último, se describirá la técnica utilizada para fusionar información de los objetos entre las dos cámaras del escenario y así establecer correspondencias entre las posiciones de los objetos en cada una de las vistas. Como hemos mencionado anteriormente, aunque este sistema está diseñado para establecer una correspondencia entre sólo dos cámaras, es extrapolable para más vistas. Esas correspondencias de posición se rigen por la homografía definida entre puntos fijos de la escena, seleccionados manualmente y vistos desde las dos cámaras.

En los siguientes puntos se explicará detalladamente cada fase del sistema y si se da el caso de haber distintas soluciones, se enumerarán las decisiones tomadas para decantarse por una u otra.

3.1 Detección de objetos en movimiento

La detección de objetos en movimiento es uno de los puntos más críticos de los sistemas de seguimiento, ya que una falsa detección se arrastrará durante todo el proceso del mismo. Como hemos estudiado, en la literatura existen muchos métodos de detección de objetos. Los más utilizados son los métodos basados en sustracción de fondo, aunque numerosas investigaciones resuelven este problema seleccionando manualmente los objetos a seguir.

En este proyecto se ha querido utilizar, tanto en la detección como en el seguimiento de objetos, estrategias que hagan uso de los puntos de interés para probar su eficiencia. Hemos diseñado un detector que localiza regiones en movimiento mediante correspondencias de

puntos característicos entre dos *frames* consecutivos. Este detector se clasifica dentro de los métodos de flujo ópticos, visto en el apartado *2.1.1.1*, y tiene un buen rendimiento tanto en cámaras fijas como en cámaras en movimiento.

El método propuesto está formado por las siguientes fases:

- ***Detección de movimiento***: Localización del conjunto de puntos en movimiento de la imagen, utilizando una correspondencia de puntos característicos entre el *frame* en el instante $i-1$ y el *frame* en el instante i, siendo i el número de fotograma de la secuencia de vídeo en el que se están detectando los objetos.
- ***Representación de objetos:*** Agrupación en objetos del conjunto de puntos en movimiento.

Para evitar que un objeto sea detectado antes de que haya entrado completamente en el área de visión de la cámara, restringimos la detección a que un objeto tenga todos sus puntos de interés dentro del área de detección. El área de detección se define como el área centrada en la imagen y que tiene un 5% de margen a cada lado con respecto a la imagen captada por la cámara. En la figura 3.2 mostramos el área de detección en color blanco y el área en la que no se detectan objetos en movimiento en color gris. La suma de ambas áreas hace el total del plano de visión de la cámara.

Figura 3.2. En blanco, área de detección de MODET.

3.1.1 Detección de puntos de interés en movimiento

El primer paso del método que hemos propuesto para la detección de objetos se basa en la búsqueda de correspondencias de puntos de interés entre dos *frames* consecutivos.

Posteriormente, se evalúa el movimiento producido por cada correspondencia y se identifica qué conjunto de puntos no corresponde al fondo. Podemos dividirlo en:

- *Localización* de puntos característicos en el fotograma $i-1$ y en el fotograma i.
- *Correspondencia* entre los descriptores de los puntos de interés del *frame* $i-1$ y del *frame* i.
- *Decisión* de qué puntos corresponden al fondo de la escena y cuáles al primer plano.

Una vez realizados dichos pasos, se alcanza un conjunto de puntos correspondiente a objetos en movimiento (adicionalmente puede introducirse ruido producido por correspondencias erróneas).

3.1.1.1 Extracción de características

La localización y descripción de puntos de interés, vistos en los apartados *2.2.1* y *2.2.2*, es una técnica utilizada para adquirir puntos distintivos de imágenes. Tiene como objetivo localizar un conjunto de puntos representativo y definir cada punto mediante un descriptor. Se utiliza para reconocer la misma característica entre diferentes vistas de un mismo objeto, ya que son repetitivos.

Esta es la parte de mayor carga temporal del sistema **MODET**, en consecuencia, vamos a utilizar los puntos y descriptores detectados para el resto de módulos que los requieran como en el seguimiento de objetos. Además, en esta parte se necesitan comparar los puntos detectados en dos *frames* consecutivos, es por ello que, para optimizar el tiempo de ejecución, sólo buscaremos los puntos característicos en el *frame* i reutilizando los descritos en el instante $i-1$. Observaremos este hecho en el capítulo *5*.

Existen multitud de métodos utilizados en la literatura para la localización de puntos de interés. Los más utilizados en investigación son SIFT y SURF.

- *SIFT* se define por su invarianza frente a las rotaciones, traslaciones, escala y cambios de iluminación. Para alcanzar estas especificaciones hay que realizar una alta carga computacional. El tamaño del descriptor con el que se define cada punto es de 128 elementos.
- *SURF* mantiene aproximadamente la misma robustez de SIFT, en cambio, tiene un descriptor de 64 elementos y, en consecuencia, una mayor velocidad en el cálculo y en la búsqueda de correspondencias.

Para decidir qué algoritmo utilizar hemos comparado los métodos SIFT y SURF. El motivo principal para decantarnos por estos algoritmos, frente a otros más recientes como ORB o BRIEF, ha sido su respaldo en distintas investigaciones científicas, ya que existen multitud de trabajos previos de tracking que los utilizan. Además, son más robustos frente a rotaciones y cambios de escala. La principal mejora de los algoritmos más recientes es su reducción en el tiempo de cálculo, pudiendo así ser aplicadas en sistemas de tiempo real.

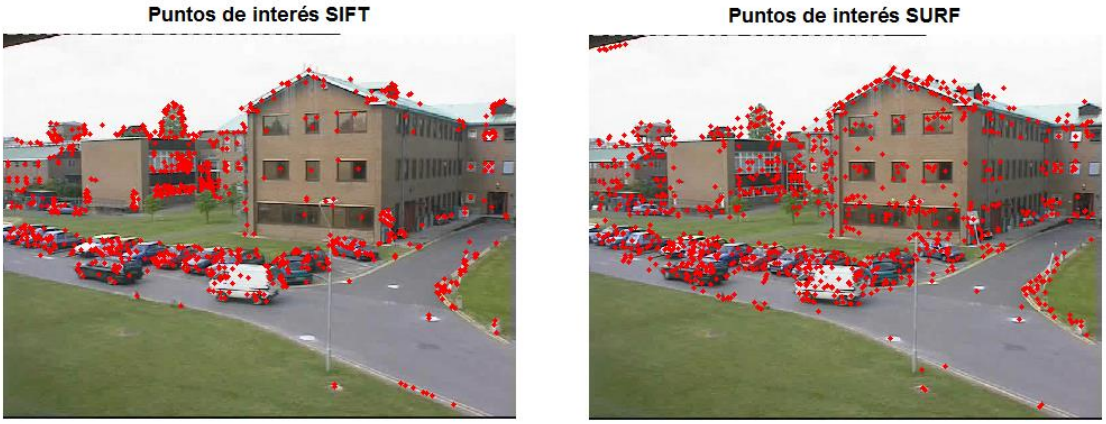

Figura 3.3. Puntos característicos SIFT y SURF.

En la figura 3.3 podemos observar los puntos de interés localizados por el algoritmo SIFT y los puntos de interés localizados por el algoritmo SURF, para una misma imagen. Hemos establecido sendos umbrales para que cada algoritmo localice el mismo número de puntos de interés. En este ejemplo, tanto para SIFT como para SURF se detectan 1030 puntos de interés. El algoritmo SIFT tarda en localizar y describir los puntos de interés 4.35 segundos, por el contrario, el algoritmo SURF tarda 1.16 segundos. Además, el algoritmo SURF encuentra puntos de interés más dispersos.

En una fase preliminar hemos utilizado SIFT como algoritmo de descripción de puntos característicos consiguiendo muy buenos resultados. Sin embargo, en las pruebas realizadas a lo largo de este proyecto hemos comprobado cómo la calidad entre SIFT y SURF es muy similar, por tanto, de ahora en adelante utilizaremos SURF para localizar y describir puntos de interés, logrando una mejora temporal de aproximadamente el 400%.

Así, para cada instante i de la secuencia, conseguimos un conjunto de puntos de interés, descritos por el algoritmo SURF. A continuación, pasamos a la fase de búsqueda de correspondencias entre el conjunto de puntos detectado en el *frame* $i-1$ y el conjunto detectado en el *frame* i.

3.1.1.2 Correspondencias

Las correspondencias, como hemos estudiado en detalle en el apartado *2.2.5*, identifican el mismo punto entre dos conjuntos de puntos, cada uno perteneciente a una imagen distinta. En la fase de detección de movimiento, vamos a buscar siempre correspondencias entre *frames* adyacentes.

Figura 3.4. Correspondencia entre puntos característicos del *frame* i - 1, círculo rojo, y del *frame* i, cruz verde.

Como podemos observar en la figura 3.4, gracias a estas correspondencias adquirimos el movimiento de cada punto basándonos en las localizaciones en el *frame* $i-1$ (círculo rojo) y en el *frame* i (cruz verde) y así podemos identificar qué puntos de interés pertenecen a objetos en movimiento y cuáles a fondo.

3.1.1.3 Decisión de los puntos correspondientes al fondo.

Después de obtener el movimiento de cada punto de interés localizado, establecemos las reglas para decidir qué puntos corresponden al fondo y qué puntos corresponden al primer plano.

Para establecer los criterios de decisión partimos de la siguiente hipótesis:

- Suponemos que la mayoría de puntos característicos localizados pertenecen a puntos del fondo del escenario. Basamos esta afirmación en que el área correspondiente a objetos en movimiento, generalmente, va a ser mínima en comparación con el escenario completo. Esto es correcto para todos los vídeos analizados en el apartado 5.

Apoyándonos en esta hipótesis podemos establecer que, los puntos que tengan consenso con el modelo de movimiento mayoritario del conjunto total de puntos, pertenecen al fondo de la escena. En escenarios con cámaras fijas el movimiento mayoritario será de cero píxeles y la matriz de transformación en coordenadas homogéneas:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

Para alcanzar este modelo de transformación, así como los puntos que contiene, vamos a utilizar algoritmos de alineamiento. Gracias a estos algoritmos seremos capaces de agrupar los puntos que pertenecen al fondo y los puntos que pertenecen al primer plano.

Como revisamos en el apartado *2.2.6* del estado del arte, el algoritmo de alineamiento más utilizado es RANSAC. El problema de RANSAC en vídeos de baja calidad, es que puede fallar al no encontrar el número de inliers suficientes para estimar el movimiento. Para este módulo del algoritmo siempre vamos a tener el número de puntos suficientes para estimar el movimiento mayoritario ya que utilizamos todos los de la imagen. Pero como veremos más adelante, en el módulo de seguimiento buscamos correspondencias entre conjuntos reducidos de puntos, pertenecientes a puntos del objeto, por lo que pueden no ser suficientes para RANSAC. RAMOSAC en cambio, estudiado en detalle en el apartado *2.2.7*, establece un modelo de transformación más o menos complejo, ajustándose al tamaño del conjunto de correspondencias, y así ofreciendo mejores resultados en estimaciones de movimiento con conjuntos de correspondencias reducidos. Por consistencia con el módulo de seguimiento, vamos a utilizar también en la detección RAMOSAC, aunque obtendríamos resultados comparables con RANSAC. Una vez que, mediante RAMOSAC, encontramos el modelo de transformación que contiene el mayor número de puntos de consenso o *inliers*, podemos etiquetar los *inliers* como fondo de la escena. Los puntos de interés que no siguen el modelo u *outliers*, corresponden al primer plano. Este conjunto contiene inevitablemente, además de los puntos correspondientes a objetos en movimiento, puntos generados por errores en el cálculo de correspondencias.

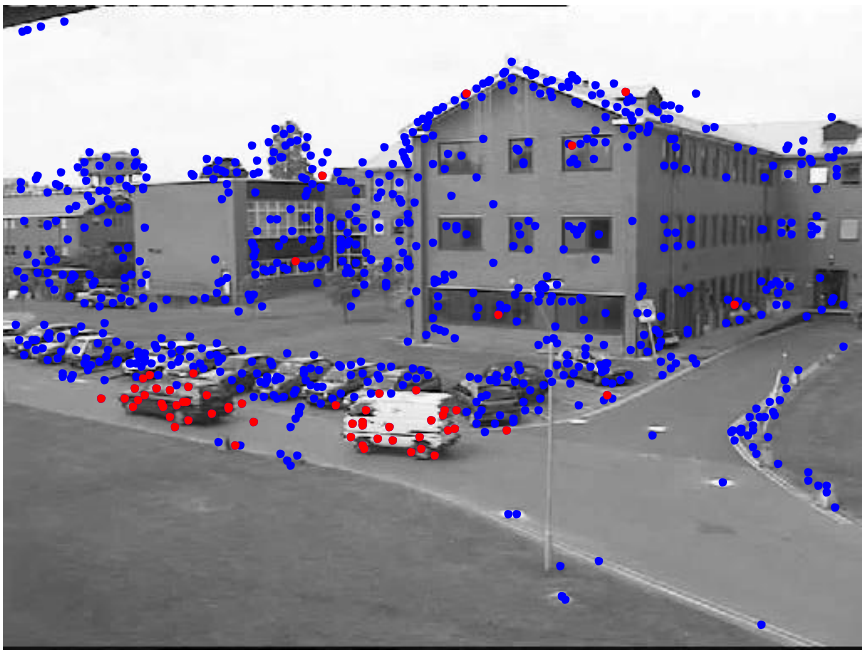

Figura 3.5. En azul, puntos de interés etiquetados como fondo, y en rojo, puntos de interés etiquetados como primer plano.

En la figura 3.5 se puede observar un ejemplo de puntos correspondientes al fondo y los puntos correspondientes al primer plano. En el siguiente apartado conectaremos los puntos del primer plano que correspondan a objetos, filtrando los puntos que pueden ser debido a errores.

3.1.2 Representación de objetos en movimiento

En esta parte vamos a segmentar el conjunto de puntos de interés que corresponden a puntos en movimiento. La segmentación es el proceso por el cual divides un conjunto de puntos en distintos grupos. Después de esta fase, tendremos todos los objetos en movimiento dentro del escenario.

Ésta es la última etapa del algoritmo de detección de objetos en movimiento. Una vez que hemos localizado el conjunto de puntos del escenario en movimiento, tanto puntos correspondientes a objetos en movimiento como posibles errores generados por fases anteriores, procedemos a su segmentación. Esta segmentación nos permite ser capaces de discernir entre los distintos objetos que aparezcan en la escena e, incluso, eliminar los errores existentes.

Primero, se realiza un análisis de conglomerados o clúster con el fin de clasificar en grupos el conjunto de puntos. El objetivo de esta técnica es la de agrupar elementos de forma que, el grado de cercanía entre los miembros del mismo grupo sea más fuerte que el grado de cercanía entre puntos de distintos grupos. Existen dos grandes tipos de algoritmos de

agrupación que son jerárquicos y no jerárquico. Dentro de los métodos jerárquicos, en este proyecto vamos a utilizar el método aglomerativo por distancia mínima. El método aglomerativo de la distancia mínima asigna como puntuación entre dos grupos el valor mínimo de las distancias entre los puntos del grupo 1 y los puntos del grupo 2. Una vez que están definidas todas las distancias entre los distintos grupos, se agrupan escogiendo un umbral correspondiente a una distancia mínima de corte.

En la figura 3.6 observamos un dendrograma, que es la representación gráfica en forma de árbol que mejor ayuda a interpretar el resultado de un análisis de cluster. En la gráfica podemos diferenciar dos grupos de elementos distanciados entre sí. Cada uno de los grupos corresponde a un objeto en movimiento. Eligiendo un adecuado parámetro de distancia mínima de corte, línea azul, y gracias al análisis de conglomerados, podremos separar el conjunto de puntos en grupos que serán candidatos a objetos.

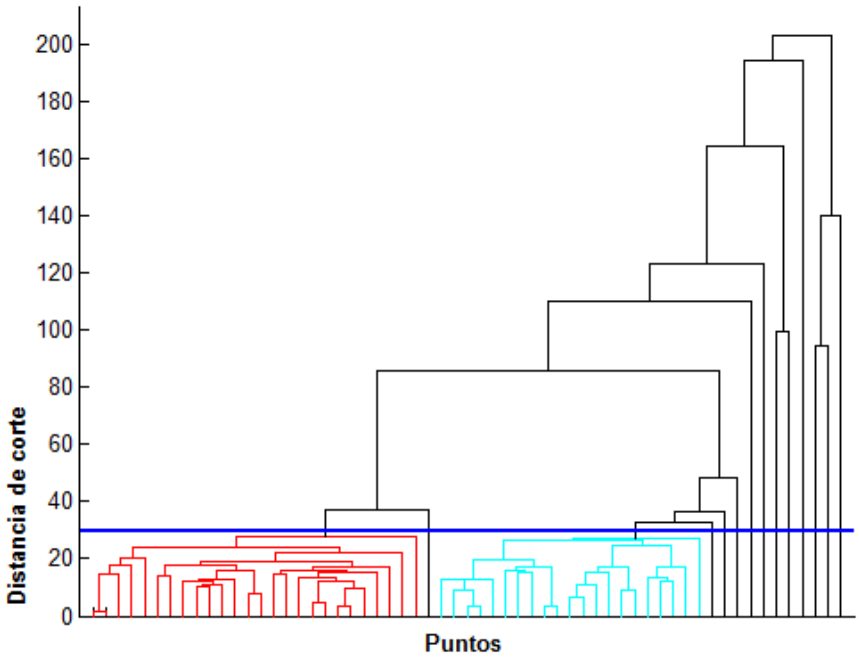

Figura 3.6. Dendrograma de un conjunto de puntos donde existen dos objetos diferenciados.

Una vez que tenemos dividido en grupos el conjunto de puntos en movimiento utilizamos las siguientes restricciones para minimizar los posibles errores que puedan producirse:

- Para formar un objeto, éste tiene que tener un número mínimo de puntos .
- El objeto tiene que estar completamente dentro del área de detección.

Como resultado para el ejemplo de la figura 3.6, se obtienen dos objetos mostrados en la figura 3.7.

El valor de los parámetros de estas restricciones varía en función del escenario utilizado y se escoge en la fase de entrenamiento.

Cuando ya tenemos los puntos que definen cada objeto detectado, seleccionamos el área que rodea el objeto o región de interés. Representamos el objeto con un rectángulo de color definido por los puntos máximos del eje X y del eje Y, y los puntos mínimos del eje X y del eje Y del objeto detectado, alcanzando por cada objeto el sistema de coordenadas homogéneas siguiente:

$$\begin{bmatrix} x_{min} & x_{min} & x_{max} & x_{max} \\ y_{min} & y_{max} & y_{max} & y_{min} \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.2}$$

siendo cada columna un vector que representa cada una de las esquinas del objeto.

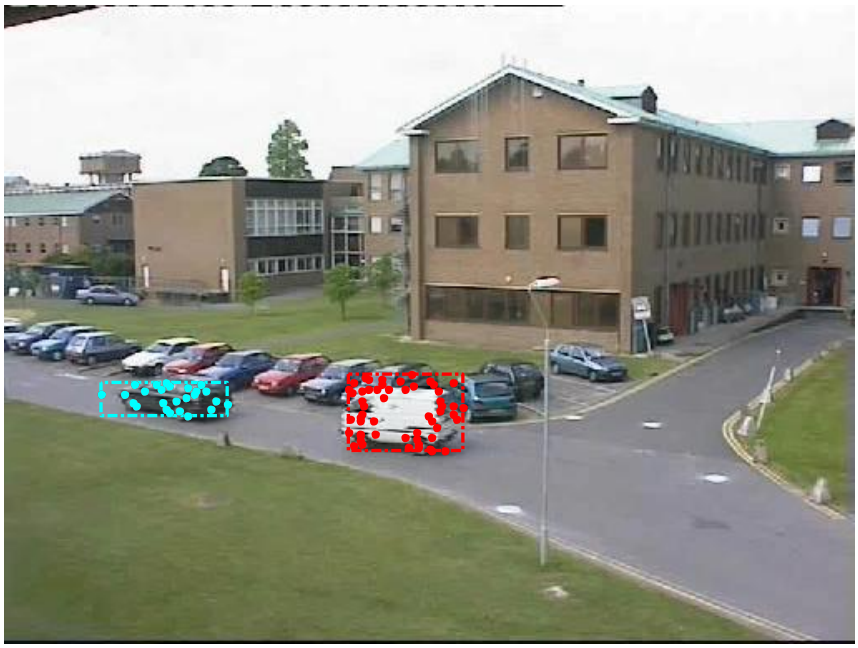

Figura 3.7. Representación de objetos detectados en movimiento.

Finalmente, comprobamos que los objetos detectados en la escena no estén en el módulo de detección. Decidimos que un objeto está ya almacenado si su superficie tiene solapamiento con el área de un objeto ya detectado. Si el nuevo objeto detectado contiene área de solapamiento con un objeto almacenado, este objeto es desechado. Por cada nuevo objeto que detectamos, almacenamos las localizaciones y descriptores de los puntos de interés que están situados dentro de la superficie que lo delimita y etiquetamos el objeto asignándole un número identificativo.

Para evitar que se compare el área de los objetos detectados y el área de los objetos almacenados situados en instantes de tiempo distintos, antes de ejecutar esta fase

ejecutaremos la fase de seguimiento (punto *3.2*), situando los objetos almacenados también en el *frame* i. Así conseguiremos que la posición de los objetos detectados y la posición de los objetos almacenados pertenezcan al instante de tiempo i en la comparación.

3.2 Seguimiento de objetos

El seguimiento de objetos es la segunda etapa del algoritmo **MODET**. Para esta fase se ha decidido utilizar tracking por puntos. Como en la fase de detección, se utilizan puntos de interés para localizar correspondencias del objeto a lo largo de la secuencia de vídeo. Para estimar la nueva posición del objeto se utiliza un método estadístico.

La finalidad de este módulo es estimar, en el *frame* i, la posición de cada objeto almacenado por el detector. Para ello, busca correspondencias entre las características de cada objeto guardado y los descriptores de los puntos de interés extraídos en el *frame* i. Con esta información, podremos calcular la transformación que va a sufrir cada objeto y actualizar el conjunto de características que lo define.

El método de tracking propuesto se compone de las siguientes fases:

- ***Extracción de características***: Para aumentar la eficiencia del **MODET** se utilizan los mismos puntos encontrados y descritos del *frame* i que en la fase de detección.
- ***Correspondencias:*** Se realiza una comparativa entre el conjunto de características almacenadas de cada objeto (*frame* $i-1$) y el conjunto de características del *frame* i y se buscan las correspondencias de puntos.
- ***Transformación 2D***: Se estima la transformación de movimiento de cada objeto basándose en las correspondencias conseguidas en la fase anterior.
- ***Actualización***: Se actualiza la información recogida de cada objeto.

Todo este conjunto de fases se ejecutan secuencialmente en cada *frame*.

Las dos primeras fases, extracción de características y correspondencias, no las describimos en detalle ya que son idénticas a las anteriormente explicadas en los apartados *3.1.1.1* y *3.1.1.2*. En la primera etapa del módulo de seguimiento, se utilizan los mismos puntos de interés localizados por **MODET** que en el módulo de detección del *frame* i. Este conjunto de puntos característicos se compara con los puntos característicos de los objetos almacenados localizados en el *frame* $i-1$, encontrando correspondencias entre ellos. Una vez que tenemos las correspondencias entre los puntos almacenados y los puntos del *frame* i, pasamos a la fase de estimación de movimiento donde se localizará la nueva posición de cada objeto.

3.2.1 Cálculo de la transformación de cada objeto.

Esta etapa es la encargada de estimar la nueva posición de los objetos. Para ello, se van a utilizar las correspondencias entre los puntos característicos almacenados de los objetos y los puntos característicos detectados en el *frame* i.

Normalmente, de este conjunto de correspondencias no todas son correctas. Existen coincidencias erróneas que pueden estar alejadas de la posición real del punto y desvirtuar el modelo de transformación. Este efecto tiene mayor influencia cuanto menor sea el número de correspondencias. Para minimizar este problema vamos a utilizar el mismo algoritmo de alineamiento que usamos en la fase de detección, RAMOSAC. Como hemos visto, este tipo de algoritmos nos permiten encontrar un modelo de transformación basado en el desplazamiento del conjunto de puntos de interés que han tenido correspondencia, desechando los puntos que tengan una transformación diferente a la de consenso, ya sea por errores en las correspondencias o porque esos puntos no pertenecen al objeto. Además, estas transformaciones, como hemos visto en el apartado *2.2.7*, se ajustan en escala y rotación al objeto, no como Mean Shift [32] que sólo traslada el objeto. En la figura 3.8 mostramos cómo se ajusta en escala y orientación el recuadro que delimita al objeto a lo largo de la secuencia.

Figura 3.8. Ajuste en escala y orientación del recuadro que delimita el objeto a lo largo de la secuencia de vídeo.

En vídeos de baja calidad, en los que los objetos están definidos por un número pequeño de puntos, existe la posibilidad de no conseguir el número suficiente de correspondencias y, en consecuencia, estimar una transformación errónea. RAMOSAC nos permite utilizar múltiples modelos de transformación de diferente complejidad. Gracias a esto, cuando se tienen muchas correspondencias entre pares de puntos, se puede utilizar un modelo complejo y cuando se tienen pocas, uno más simple. Para determinar qué modelo elegir, se asigna a cada tipo una puntuación permitiendo utilizar el más adecuado.

Para estimar la posición del objeto en el *frame* i, obtenemos con RAMOSAC el modelo de transformación de cada objeto que contiene el mayor número de puntos de consenso. En la figura 3.9, mostramos la transformación del rectángulo que delimita el objeto, en base a las correspondencias localizadas entre los puntos de interés almacenados y los puntos de interés en el *frame* i.

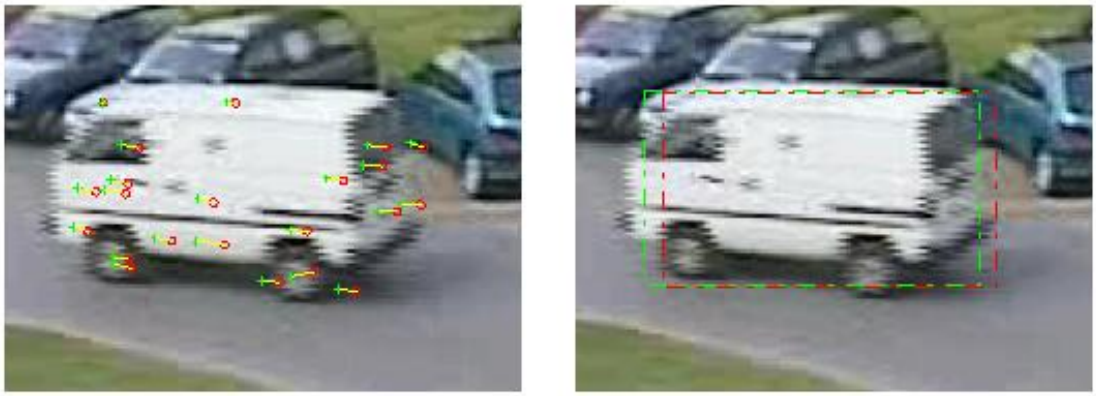

Figura 3.9. En la imagen izquierda, en círculos rojos la posición de los puntos de interés almacenados (*frame* i-1) y en cruces verdes sus correspondencias en el *frame* i. En la imagen derecha, el rectángulo rojo indica el área del objeto almacenado (*frame* i-1) y el rectángulo verde indica la posición estimada del objeto en el *frame* i.

En la fase de detección, hemos restringido (figura 3.2) el escenario de captación de movimiento de objetos a un porcentaje de la superficie de la imagen grabada por la cámara, asegurándonos así que el objeto ha entrado completamente en el plano. En la fase de seguimiento, utilizamos la misma área de detección para estimar cuándo un objeto va a salir del escenario y detener su seguimiento. Cuando el centro de masa del objeto se encuentre fuera del área de detección, detendremos su seguimiento porque estimaremos que va a salir de la escena. Además también detendremos el movimiento de los objetos que obtengan una puntuación del modelo de transformación, ecuación 2.20, inferior a un umbral.

3.2.2 Actualización del conjunto de puntos de cada objeto.

Una vez obtenido el modelo de transformación óptimo del objeto, se actualizan los puntos de interés almacenados y se añaden los puntos nuevos detectados en el área del objeto. Este paso es fundamental, ya que si no se actualizasen estos puntos siempre se partiría de la información inicial. En este proyecto se va a partir del método de actualización de información utilizado en [30]. El método consiste en actualizar el conjunto de características almacenado del objeto a través de un sistema de puntuación que evalúa cada punto. Cada punto se clasifica de acuerdo a estos valores de puntuación y sólo se utilizan en la etapa de seguimiento los puntos del objeto con mayor puntuación. La puntuación de cada objeto en el instante de tiempo i se actualiza por la siguiente regla:

$$Score_i = \begin{cases} Score_{i-1} + 2 \rightarrow \text{Hay correspondencia con un punto de interés del } frame\ i \\ \qquad\qquad \text{y sigue el modelo mayoritario de transformación.} \\ Score_{i-1} - 1 \rightarrow \text{Hay correspondencia con un punto de interés del } frame\ i \\ \qquad\qquad \text{y no sigue el modelo mayoritario de transformación.} \\ Score_{i-1} \rightarrow \text{No hay correspondencia con un punto de interés del } frame\ i. \end{cases}$$

Además, se añaden al conjunto de puntos del objeto todos los puntos de interés detectados en la nueva localización y que no pertenezcan a otro objeto (objetos con áreas

solapadas). Estos nuevos puntos tendrán la valoración media de todos los puntos de ese objeto para así, no interferir en los puntos con menor y mayor puntuación.

3.3 Alineación de varias vistas

Las tareas de detección y seguimiento de objetos de **MODET** se realizan para cada una de las cámaras utilizadas, ofreciéndonos la posibilidad de fusionar toda esta información para obtener una visión multicámara de la escena.

En esta sección se propone el uso de la homografía proyectiva para fusionar la información de detección y seguimiento almacenada por cada una de las cámaras. Se ha decidido utilizar homografía proyectiva para conseguir un sistema independiente del tipo de cámaras, ya que así no será necesario calibrarlas ni disponer de información previa de ellas.

Para poder calcular la homografía, es necesario disponer de un conjunto de correspondencias de puntos de la escena que sean visibles desde ambas cámaras. Existen dos opciones:

- *Especificar* manualmente el número de puntos estáticos del escenario y visibles por las dos cámaras. Su aplicación queda reducida al uso de cámaras estáticas.
- *Buscar correspondencias* de puntos entre las dos vistas. Para este método, se pueden utilizar algoritmos de extracción de puntos de interés. Esta opción es muy adecuada para vídeos con cámaras en movimiento, aunque por el contrario, exige que la apariencia de los objetos en la escena no sea muy diferente desde los distintos puntos de vista.

En este proyecto vamos a especificar manualmente los puntos estáticos de la escena, por lo que sólo podremos utilizar la fusión de información en escenarios con cámaras fijas. Mediante esta fusión de información vamos a detectar cuándo las dos cámaras están captando el mismo objeto y vamos a ser capaces de recuperar objetos perdidos por la fase de seguimiento o detectar objetos en las dos cámaras que solamente han sido captados por una de las cámaras. Estas aplicaciones se explicarán en detalle en las secciones *3.3.2* y *3.3.3*.

3.3.1 Cálculo de homografía

Para encontrar una correspondencia entre las dos vistas y poder fusionar los datos de los objetos vamos a utilizar homografía proyectiva. Para realizar el cálculo de la homografía, tenemos que escoger manualmente puntos estáticos de la escena visibles desde ambas cámaras (al menos 4 puntos). En el ejemplo de la figura 3.10 hemos seleccionado 7 puntos marcados en el escenario.

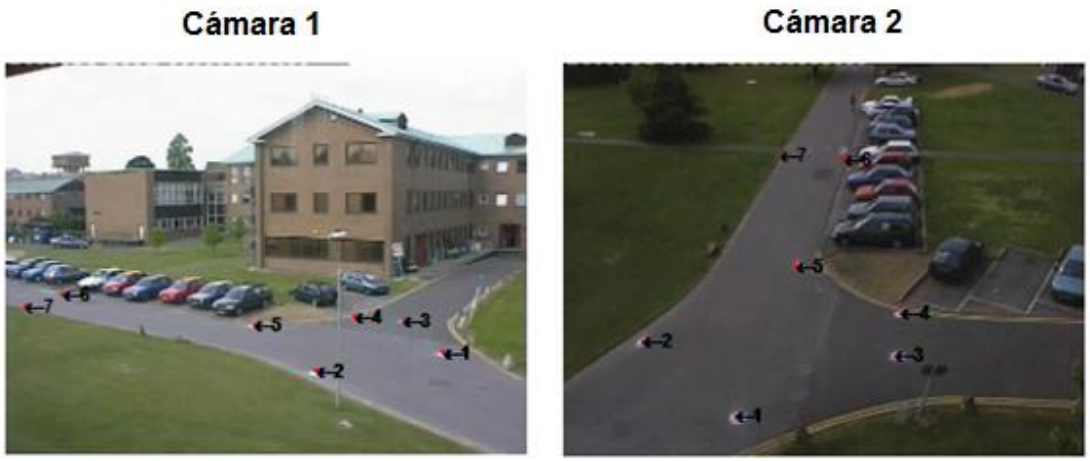

Figura 3.10. Puntos fijos utilizados para el cálculo de la homografía.

Con estos puntos de referencia calculamos la matriz H, utilizando el algoritmo DLT 2D descrito en el apartado *2.3.1.1*, que mapea puntos de la primera cámara a la segunda cámara.

3.3.2 Algoritmo para asignar objetos

Las aplicaciones de la fusión de datos que vamos a usar en este proyecto para hacer de **MODET** un sistema más robusto son:

- Localizar en las dos cámaras objetos perdidos por alguna de las cámaras.
- Localizar en las dos cámaras objetos que sólo han sido detectados por una de las cámaras.

Para esto, el primer paso es averiguar cuándo un objeto es detectado por las dos cámaras. Para ello hemos implementado un algoritmo que empareja objetos de la cámara 1 y de la cámara 2, cuando se decide que pertenecen al mismo objeto físico del escenario. La notación empleada es la siguiente:

1 Calculo el punto central de todos los objetos que aparecen en ambas vistas.
2 Calculo las distancias entre los centros de los objetos que capta la cámara 1 y los centros de los objetos detectados en la cámara 2, desplazados mediante la homografía a su posición en la cámara 1.
3 Obtengo el par de objetos que tiene la menor distancia entre sus centroides y decido que corresponden al mismo objeto, siempre que la distancia sea menor que un umbral determinado. Elimino ambos objetos de la matriz de distancias.
4 Repito el punto 3 hasta que todos los objetos hayan sido asignados o desechados.
5 Pintamos los marcos del mismo objeto con el mismo color.

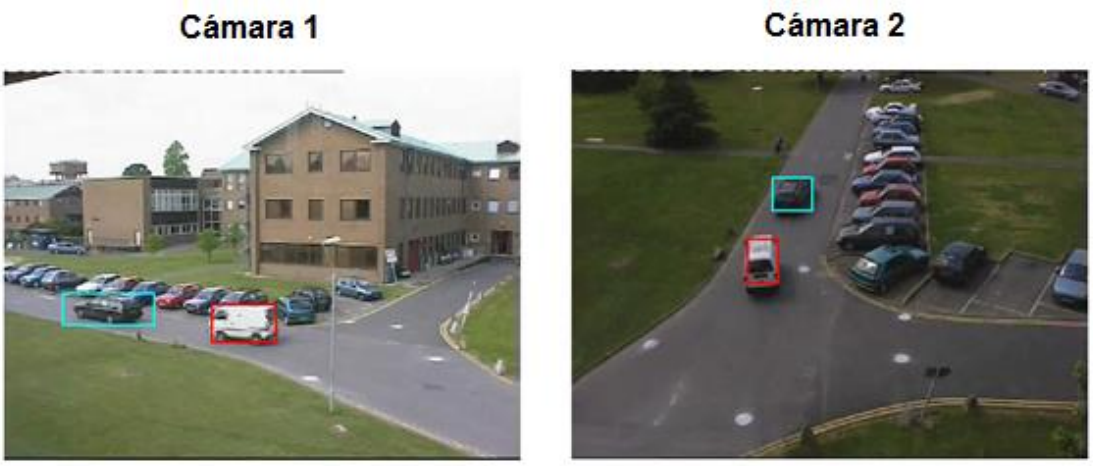

Figura 3.11. Emparejamiento de objetos detectados en dos cámaras.

En la figura 3.11 podemos observar un ejemplo del emparejamiento de objetos en el que se representan del mismo color los marcos que recuadran el mismo objeto del escenario.

3.3.3 Cálculo de la posición de un objeto en la vista alterna

Otra de las ventajas de los sistemas multicámara es la posibilidad de triangular información de los objetos entre imágenes. Así, podemos utilizar la información del objeto en una de las cámaras para localizar la posición del mismo objeto en la otra cámara. Esto se puede dar en los casos en los que el detector no ha captado el objeto en las dos vistas o el módulo de seguimiento lo ha perdido en una de las vistas.

Hemos desarrollado un algoritmo que nos permite estimar la posición de un objeto en la vista contraria a la que se ha detectado. Para ello, se determina el número de vistas en las que aparece cada uno de los objetos mediante el algoritmo del apartado *3.3.2*. Una vez que tenemos identificados los objetos que aparecen únicamente en una de las vistas calculamos su posición gracias a la homografía proyectiva calculada en el punto *3.3.1*. Si el objeto ha sido localizado en la vista 2, tendremos que utilizar la homografía inversa para estimar su posición en la vista 1. Este procedimiento nos permite mejorar el rendimiento de **MODET** frente al de datos de una sola cámara.

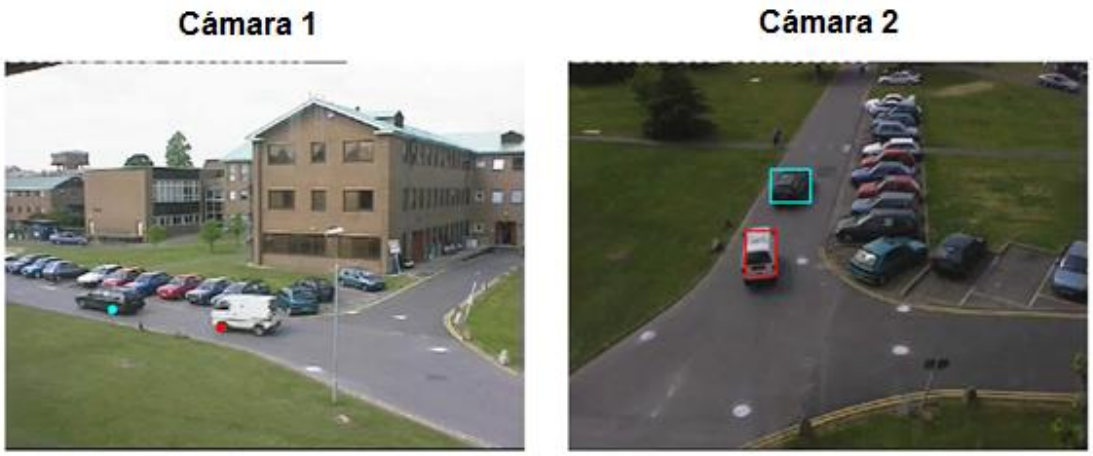

Figura 3.12. Estimación del centro de los objetos en la cámara 1 con la información de la cámara 2.

En la figura 3.12 observamos cómo, mediante esta técnica, podemos estimar la posición de objetos en la cámara en la que no están siendo detectados.

4 DESARROLLO

En este capítulo vamos a establecer los detalles de implementación del algoritmo **MODET**.

Para la implementación de este proyecto se ha decidido utilizar la herramienta de desarrollo MATLAB. MATLAB (*MATrix LABoratory*) es una herramienta de software matemático muy utilizado en la docencia y la investigación. Este software dispone de múltiples herramientas denominadas *toolboxes*, entre las cuales hay una específica para el procesamiento de imágenes. Esta *toolbox* está formada por un conjunto de funciones que facilitan enormemente el análisis de imágenes. Para la simplificación del proyecto hemos utilizado algunas de estas funciones.

Hemos estructurado la implementación del sistema **MODET** en las siguientes partes principales:

- Leer los vídeos de entrada de cada cámara.
- Procesar la información de cada fotograma para detectar los objetos en movimiento.
- Realizar un seguimiento de cada objeto almacenado estimando su posición.
- Actualizar la información de los objetos
- Calcular la homografía entre las dos cámaras
- Asignar objetos coincidentes en ambas cámaras.
- Recuperar objetos perdidos en el escenario con la información de la otra cámara.
- Generar un vídeo donde se visualiza el seguimiento de los objetos detectados.

A continuación, detallamos el desarrollo y las funciones utilizadas en cada una de estas partes.

4.1 Leer los vídeos de entrada

Para leer la información de los vídeos de entrada hemos utilizado la función de MATLAB `VideoReader`[1]. Esta función tiene de parámetro de entrada el nombre del vídeo y de parámetro de salida un objeto con toda la información del vídeo.

```
moviereader = VideoReader(inputVideo);
```

Utilizamos los datos del objeto de la clase `VideoReader` para leer iterativamente cada fotograma del vídeo, siendo `i` el número de fotograma que captura,

```
movieFrame = read(moviereader,i);
```

y para almacenar el número total de fotogramas que contiene el video,

```
endframe = moviereader.NumberOfFrames;
```

[1] http://www.mathworks.es/es/help/matlab/ref/videoreaderclass.html

Una vez que hemos leído el fotograma correspondiente, convertimos la imagen a escala de grises para localizar y describir los puntos de interés mediante el algoritmo SURF (ver figura 4.1).

Figura 4.1. A la izquierda, fotograma en rgb, y a la derecha, fotograma en escala de grises.

4.2 Detectar los objetos en movimiento

Siguiendo la metodología definida en el apartado *3.1*, para localizar los objetos en movimiento en la escena, hacemos una búsqueda, en cada *frame*, de posibles objetos en movimiento. Una vez localizados los objetos en movimiento del escenario, comprobamos que no estén en la fase de seguimiento de **MODET**. Si son objetos nuevos, los añadimos para que pasen al módulo de seguimiento.

Seguidamente, describiremos el desarrollo realizado en la fase de detección y adhesión de nuevos objetos en movimiento.

4.2.1 Detección de objetos en movimiento

Para localizar los objetos en movimiento de la escena vamos a seguir los pasos definidos en el apartado *3.1.1*.

El primer paso es localizar y describir los puntos de interés tanto del *frame* en el instante $i-1$ como del *frame* en el instante i (cabe recordar que los puntos de interés del frame $i-1$ serán los detectados en el instante de tiempo $i-1$). Para localizar los puntos hemos utilizado la función de MATLAB `detectSURFFeatures`[2],

```
points = detectSURFFeatures(frameActual,'MetricThreshold',SURFThreshold);
```

[2] http://www.mathworks.es/es/help/vision/ref/detectsurffeatures.html

y para describirlos la función `extractFeatures`[3],

```
[features,valid_points]=extractFeatures(frameActual,points,'Method','SURF');
```

La variable `frameActual` contiene la imagen captada por la cámara en el instante i, convertida a escala de grises, y el parámetro `SURFThreshold` especifica el valor del umbral de la función `detectSURFFeatures` que selecciona los puntos de interés con características más invariantes. Su valor dependerá de la calidad del vídeo. Para vídeos con poca calidad tendrá valores en torno a 100 y para vídeos con mucha calidad tendrá valores en torno a 1000. El resto de parámetros utilizados son los que fija MATLAB por defecto.

Estas funciones nos devuelven el conjunto de localizaciones y descriptores de los puntos de interés SURF que hay en cada imagen.

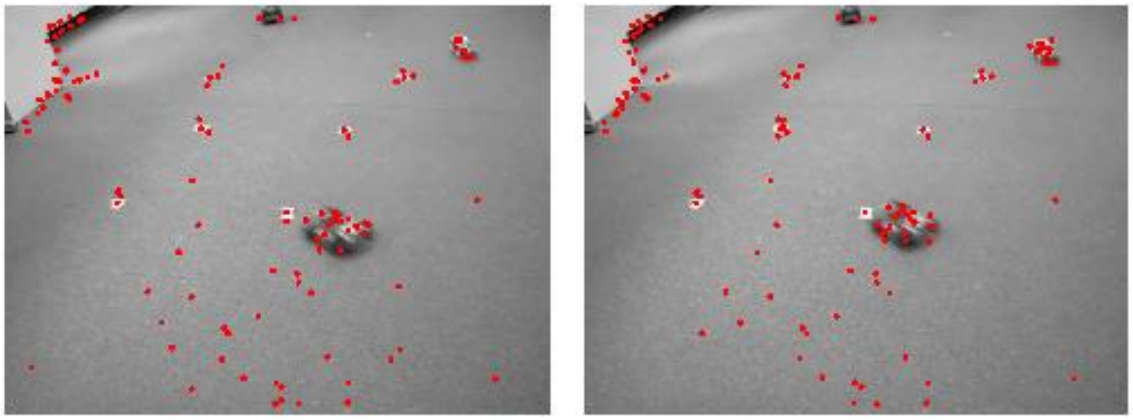

Figura 4.2. A la izquierda puntos de interés del *frame* en el instante i - 1 y a la derecha en el instante i.

En la figura 4.2 podemos observar que el número de puntos correspondientes al fondo es mayor que el número de puntos correspondientes al primer plano. Gracias a esto podemos afirmar que el modelo de transformación que siga el mayor número de puntos será el modelo correspondiente a los puntos de fondo.

Después de extraer los puntos característicos buscamos las correspondencias entre los puntos del *frame* $i-1$ y los puntos del *frame* i (ver figura 4.3). Para la búsqueda de correspondencias hemos implementado en MATLAB la función `surfmatch`. Esta función encuentra para cada punto del conjunto 1, correspondiente a los de puntos de interés del fotograma en el instante $i-1$, la correspondencia con mayor similitud del conjunto de puntos 2, correspondiente a los de puntos de interés del fotograma en el instante i, siempre que sea mayor que el umbral definido por la función de MATLAB `matchFeatures`[4]. Si algún punto del conjunto 2 tiene asignada más de una correspondencia del conjunto 1, se elige la que tenga mayor similitud y el resto se desechan.

[3] http://www.mathworks.es/es/help/vision/ref/extractfeatures.html
[4] http://www.mathworks.es/es/help/vision/ref/matchfeatures.html

```
[correspondencias] = surfmatch(descrPrev,descrActual);
```

Los parámetros de entrada de la función `surfmatch` contienen los descriptores de los puntos de interés encontrados en el instante $i-1$ y en el instante i respectivamente.

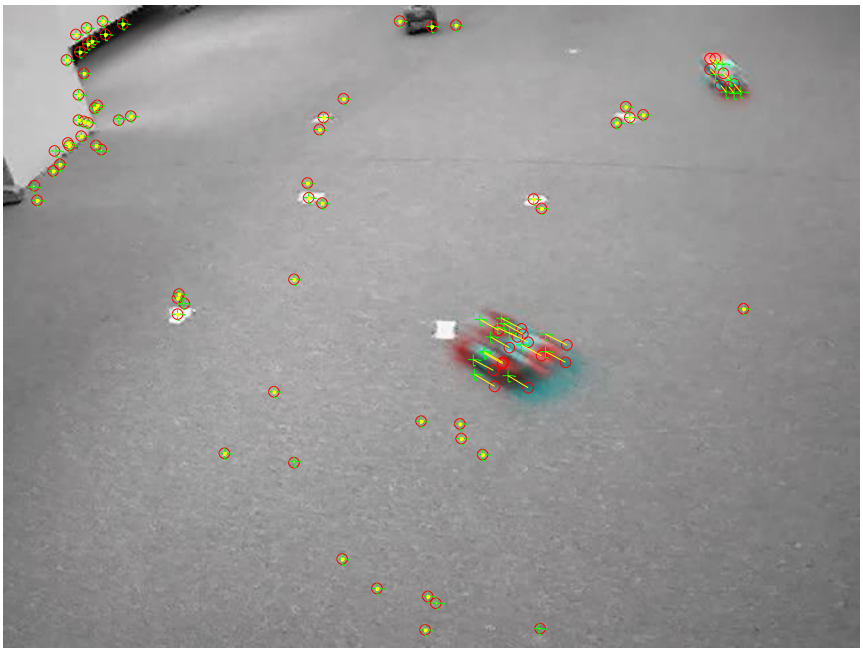

Figura 4.3. Correspondencias entre los puntos de interés del *frame* en el instante i-1 (circulo rojo) y en el instante i (cruz verde).

Una vez que tenemos los emparejamientos de puntos, se decide qué puntos corresponden al fondo y cuáles al primer plano. Para ello, se busca cuál es el modelo mayoritario de movimiento y se asigna al fondo los puntos de consenso con ese modelo.

Para obtener el modelo de transformación de fondo utilizamos RAMOSAC. Como hemos visto en el apartado *2.2.7*, este algoritmo, para calcular la puntuación de la trasformación, hace uso de una probabilidad condicional de los movimientos anteriores que ha sufrido el objetivo. Para este módulo supondremos una probabilidad nula, ya que no almacenamos estas transformaciones (utilizaremos un valor para lambda, `lambda = 0`). Hemos utilizado el módulo RAMOSAC implementado por P. Strandmark [30].

```
[~, inliers]=RAMOSAC(MatchedPointsPrev, MatchedPointsActual,...
                     box, lambda, Models, dist_thresh);
```

Las variables `MatchedPointsPrev y MatchedPointsActual` contienen las posiciones de las correspondencias encontradas. Los parámetros `Models` y `dist_thresh` son fijados en la fase de entrenamiento, y los ajustaremos en función de la rigidez de las cámaras. En las pruebas, para escenarios con cámaras fijas seleccionamos los siguientes valores:

- `Models = 1;`

- `dist_thresh = 1 o 3;`

y para escenarios con las cámaras móviles:

- `Models = [1 2 3];`
- `dist_thresh = 1;`

La variable `Models` almacena los diferentes modelos de transformación posibles y `dist_thresh` es el umbral que decide si un punto es un *inlier* o un *outlier.*

Todos los puntos que no sigan el modelo de transformación elegido por RAMOSAC serán considerados puntos de primer plano. En la figura 4.4 observamos un ejemplo de correspondencias de puntos de primer plano.

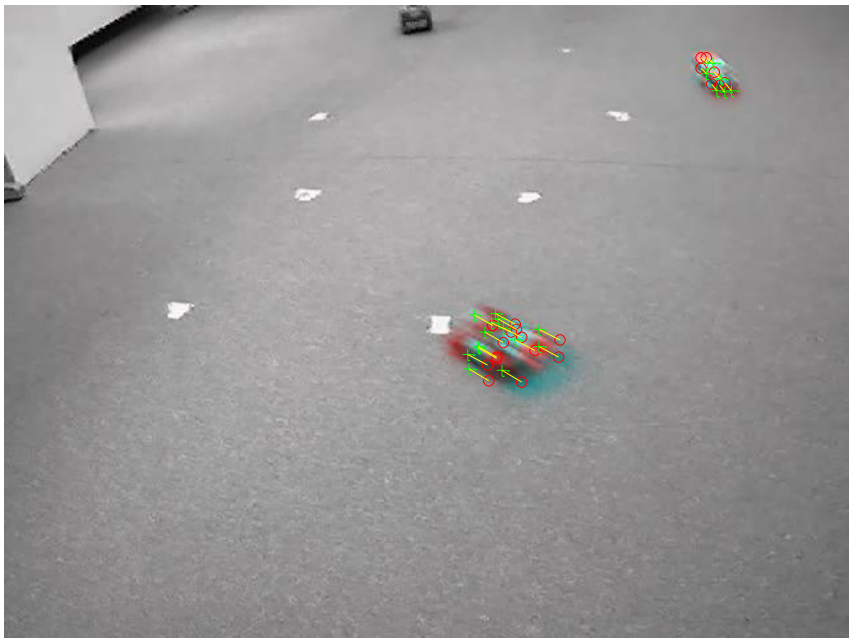

Figura 4.4. Correspondencias de los puntos de primer plano.

Por último, se segmenta el conjunto de puntos obtenido, dando lugar a un conjunto de puntos por cada objeto en movimiento en la escena. Para segmentar todos los puntos correspondientes al primer plano, primero los agrupamos utilizando clustering jerárquico aglomerativo por distancia mínima. La distancia mínima queda definida por el parámetro `kCluster`, y para el agrupamiento se utiliza la función de MATLAB `cluster`[5]. El valor de `kCluster` se decide en la fase de entrenamiento y depende de la densidad de puntos característicos que se encuentren en la imagen.

```
grupos = cluster(Z,'cutoff',kCluster,'criterion','distance');
```

[5] http://www.mathworks.es/es/help/stats/cluster.html

En la figura 4.5 vemos cómo seleccionando un valor adecuado para el parámetro `kCluster` podemos agrupar los dos objetos en movimiento del ejemplo que mostramos en la imagen 4.4.

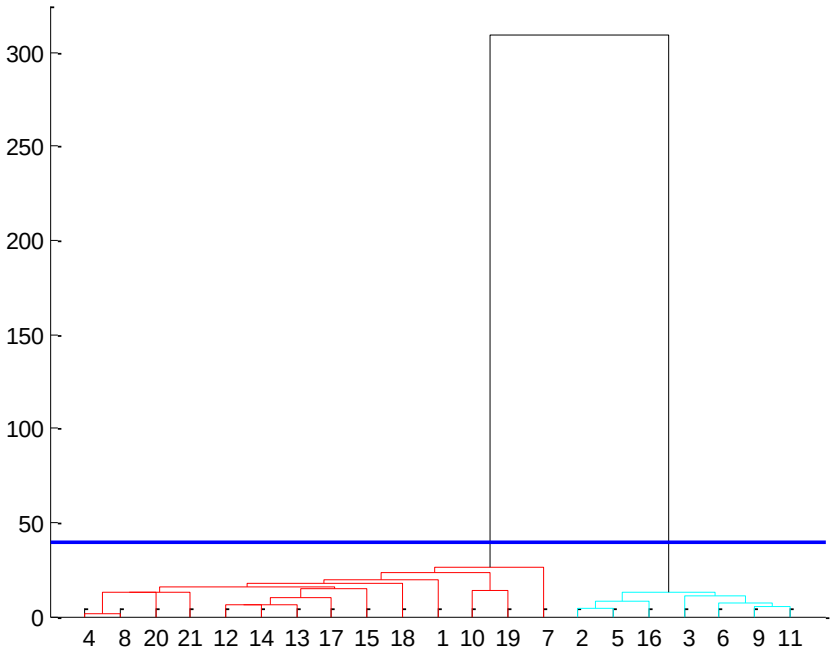

Figura 4.5. Dendrograma con los puntos correspondientes al primer plano. La línea azul corresponde al valor de la variable `kCluster`.

Cuando ya hemos agrupado las diferentes regiones de puntos conectados, limitamos los objetos siguiendo los criterios de restricciones detallados anteriormente, por el número de correspondencias mínimas y por la completa localización del objeto dentro del área de detección. El número de correspondencias mínimas por objeto se restringe por la variable `numMinPoints` y, como `kCluster`, está ligado a la densidad de puntos de interés localizados por SURF. Para las pruebas realizadas varía entre 6 para vídeos con baja detección de puntos característicos y 25 para vídeos con alta densidad de puntos característicos. El área de detección supone el área que contiene el 90% de del ancho y el 90% del alto del total de la escena, en la figura 4.6 mostramos un ejemplo de esta superficie.

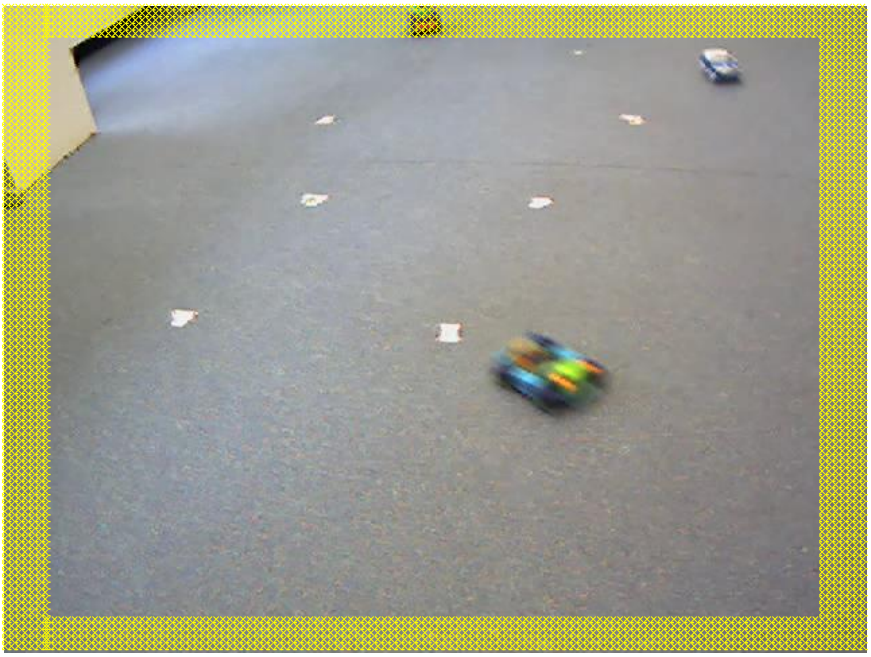

Figura 4.6. La malla amarilla limita el área de detección de la imagen.

Finalmente, representamos todos los objetos que cumplen las restricciones. Para representarlos formamos un rectángulo con los valores máximos y mínimos entre todos los puntos que lo forman como observamos en la imagen 4.7.

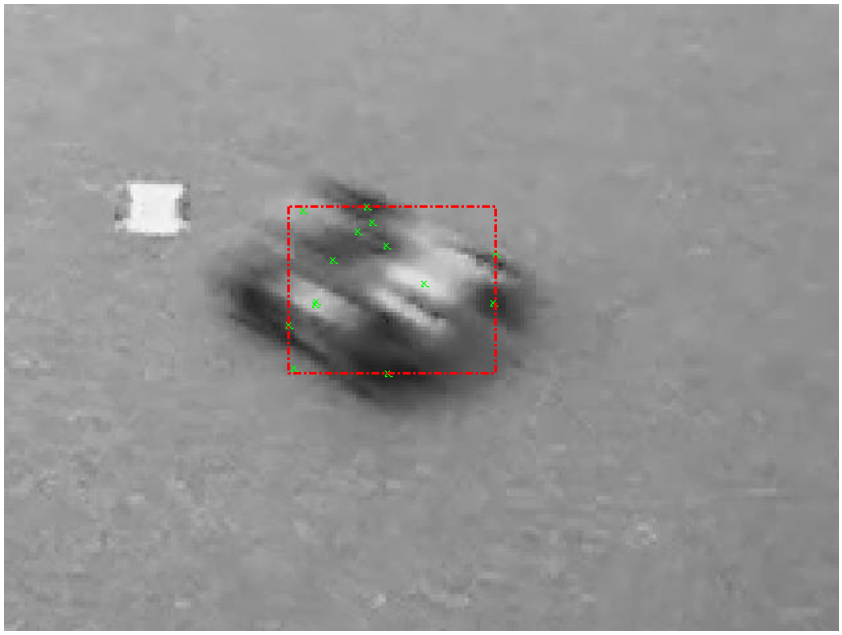

Figura 4.7. Zoom de la representación del objeto detectado.

Con los objetos en movimiento de la escena ya definidos pasamos al siguiente módulo dónde añadimos los objetos nuevos.

4.2.2 Añadir nuevos objetos

Una vez que tenemos el área que representa a los objetos en movimiento localizados en la fase anterior, añadimos los que se consideren como objetos nuevos. Comprobamos que el área de cada objeto nuevo no solape con el área de ningún objeto existente, de ser así añadimos el objeto. En la figura 4.8 mostramos el rectángulo que delimita el área de dos objetos añadidos.

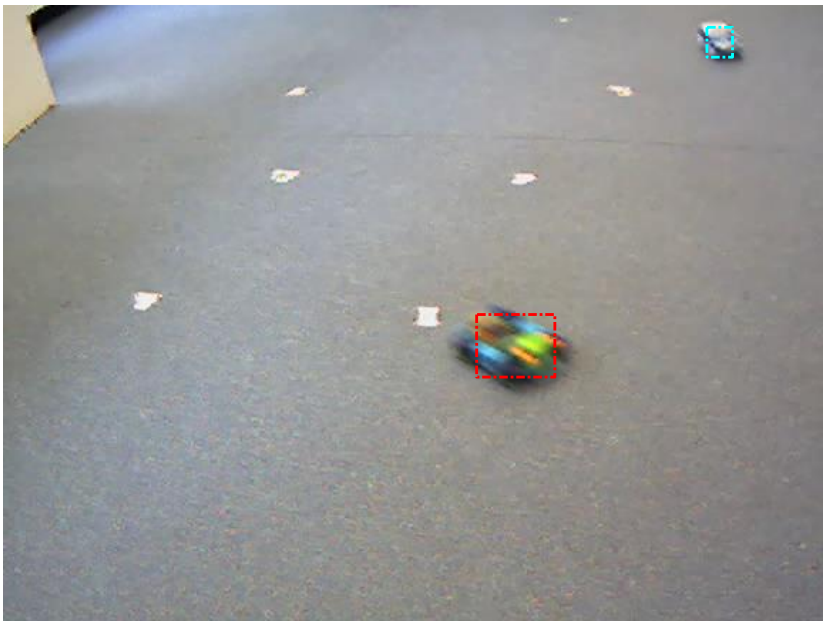

Figura 4.8. Objetos añadidos.

Como se vio en el apartado *3.1.2*, para evitar que en la comparación entre los objetos en movimiento detectados y los objetos almacenados, ambos conjuntos no pertenezcan al mismo instante de tiempo, se ejecutará previamente la fase de seguimiento (apartado *4.3*), logrando que tanto los objetos detectados como los objetos almacenados estén situados en el instante de tiempo i.

Para comprobar que el área entre los objetos detectados y los objetos almacenados (ambos situados en el *frame* i) no solape utilizamos la función de MATLAB `rectint`[6]:

```
areaSolapamiento(m)= rectint(a,b);
```

[6] http://www.mathworks.es/es/help/matlab/ref/rectint.html

siendo `a` el área de un objeto detectado y `b` el área de un objeto almacenado. Si se cumple que la suma del área común de un objeto nuevo con todos los objetos existente es nula, añadimos ese objeto.

Para cada objeto nuevo almacenamos todas las localizaciones y descriptores de los puntos de interés dentro del área definida con un rectángulo. Además, etiquetamos el objeto para asignarlo posteriormente de manera eficiente a su objeto correspondiente en la vista opuesta.

4.3 Seguimiento de objetos

Una vez que tenemos identificados todos los objetos que hay en la escena pasamos al módulo de tracking. En él, estimamos la posición en el *frame* i de los objetos almacenados. Una vez que hemos calculado la transformación de movimiento de cada objeto, estimamos si el objeto va a abandonar el escenario. De ser así, eliminamos el objeto.

4.3.1 Estimación de la transformación

Como describimos en el apartado *3.2.1*, en esta fase estimamos la posición de los objetos almacenados para el instante i. Para ello utilizamos los mismos algoritmos que en la fase de detección de objetos, SURF para la extracción de características y RAMOSAC para la estimación de movimiento.

Los *keypoints* extraídos por **MODET** en el instante i, son utilizados tanto en la fase de detección como en la fase de seguimiento.

Seguidamente, buscamos correspondencias entre los *keypoints* detectados en el *frame* i y los *keypoints* almacenados de cada objeto (localizados en el instante de tiempo $i-1$). El número máximo de puntos almacenados del objeto que se usan, está definido por la variable `numMaxPointsTracking`. Si almacenamos más puntos, solamente se utilizarán los que tengan una mayor puntuación, que se asignará en la fase de actualización (apartado 4.4). Utilizamos la misma función de matching que en el apartado *4.2.1*, `surfmatch`.

Posteriormente, estimamos la nueva posición de cada objeto calculando el modelo de transformación que contiene el mayor número de puntos de consenso. Para calcular esta transformación utilizamos la misma función de RAMOSAC que en el apartado *4.2.1*. En este caso, a diferencia de la fase de detección, se actualiza el valor de lambda con la información del coste de movimiento del objeto en el *frame* anterior:

```
lambda = 1/mean(cost);
```

Al detectar un objeto inicializamos el valor de lambda asociado con un coste de movimiento del 10% del área del objeto.

```
[H{obj} inliers{obj}] = RAMOSAC(MatchedObjPoints, MatchedPoints,...
                                box{obj}, lambda{obj}, Models{obj},...
                                dist_thresh);
```

Fijando para las pruebas realizadas en el capítulo 5 los siguientes parámetros:

- `Models = [1 2];`
- `dist_thresh = 3;`

Esta función devuelve la matriz de transformación `H` que cumple la siguiente ecuación:

```
MatchedPoints = H * MatchedObjPoints;
```

siendo `MatchedObjPoints`, las localizaciones de los puntos de interés almacenados localizados en el instante $i-1$, y `MatchedPoints` las localizaciones de los puntos de interés detectados en el instante i.

Una vez calculado el modelo de transformación, podemos estimar la posición del objeto en el nuevo fotograma haciendo una simple multiplicación entre la matriz de transformación y la posición del objeto en el *frame* $i-1$.

```
box{obj} = H{obj}*box{obj};
```

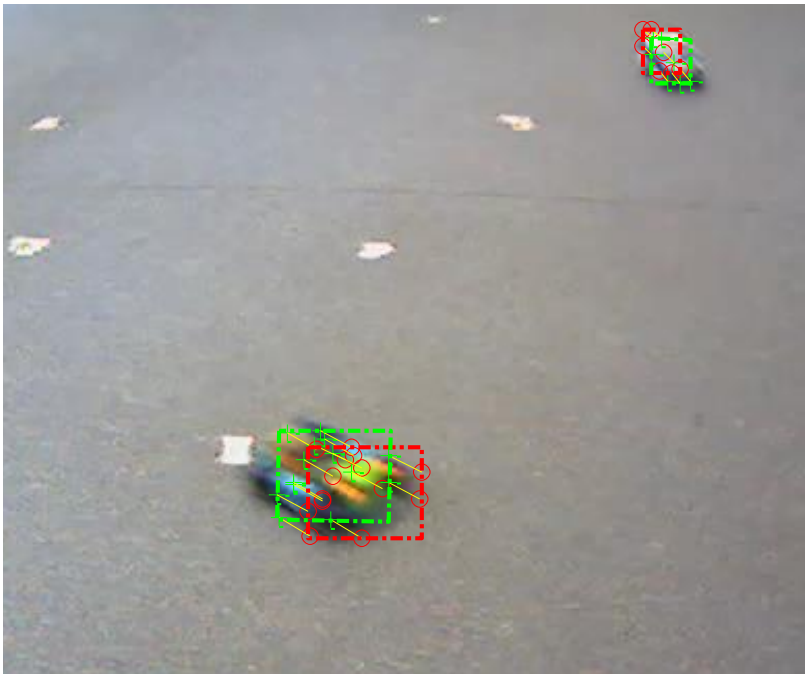

Figura 4.9. Estimación de movimiento. En rojo, box almacenado, en verde, box estimado.

En la figura 4.9 mostramos el rectángulo que delimita a los objetos en el *frame* $i-1$ (box rojo) y el rectángulo estimado que delimita a los objetos en el *frame* i (box verde).

4.3.2 Eliminación de objetos

Por último, pasamos a la fase de eliminación de objetos. Para el sistema diseñado, decidimos eliminar un objeto cuando estimamos que éste va a salir de los límites de la imagen

en el *frame* $i + 1$, o ha sufrido una transformación con una puntuación dada por RAMOSAC menor que el umbral `minScore`. Para eliminar el objeto, borramos de memoria toda la información almacenada de éste.

En la figura 4.10 mostramos un ejemplo de cuando un objeto sale del área de detección y en consecuencia estimamos que va a salir del escenario en el *frame* $i + 1$. El objeto del ejemplo, recuadro rojo, tiene su centro de masas fuera del área de detección (punto rojo) y es eliminado.

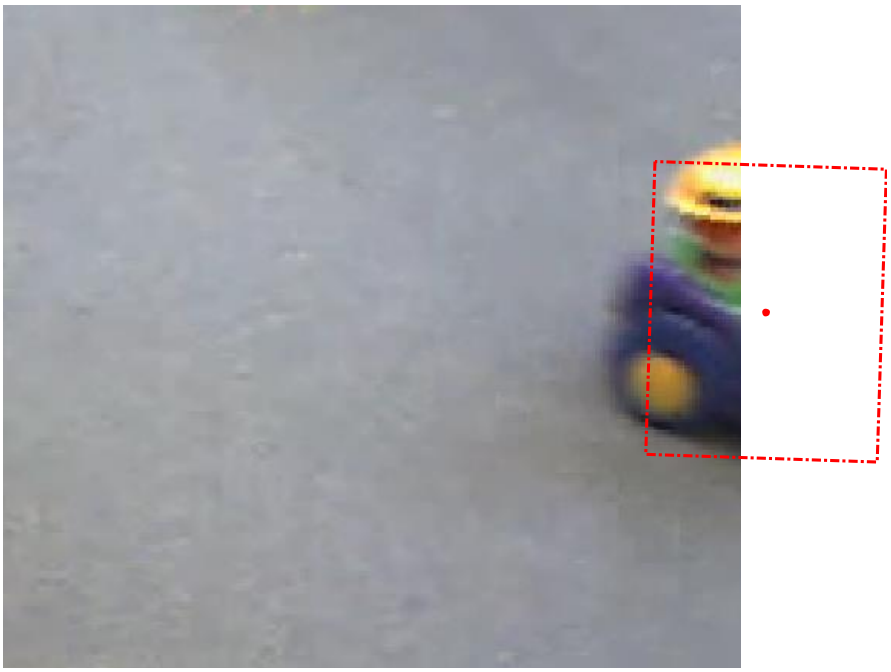

Figura 4.10. Objeto que tiene su centro de masas fuera del área de detección y es eliminado.

4.4 Actualización de información

En esta fase actualizamos toda la información de los objetos en seguimiento.

En primer lugar, transformamos la posición de los puntos de interés almacenados previamente utilizando la matriz `H`.

Después, actualizamos las puntuaciones de todos los puntos de cada objeto, sumando el valor 2 a los puntos de consenso con la transformación y restando el valor 1 a los *outliers*.

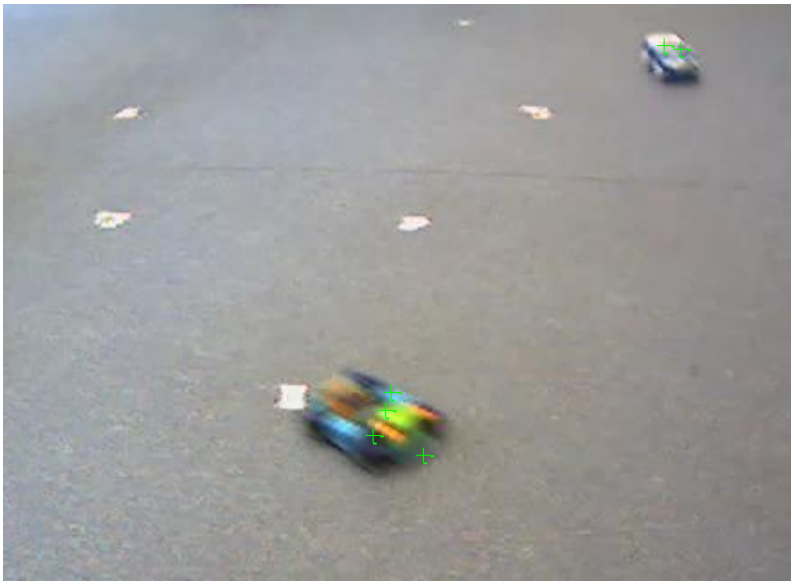

Figura 4.11. Puntos de interés añadidos en la fase de actualización de objetos.

Por último, añadimos todos los puntos característicos encontrados en el *frame i*, situados en el área que delimita el objeto y que no estén situados en el área de otro objeto (objetos con áreas solapadas). Inicializamos la puntuación de estos puntos con el valor medio de las puntuaciones de los puntos almacenados. En la figura 4.11 mostramos un ejemplo de puntos añadidos en la fase de actualización de objetos.

4.5 Calculo de homografía

Para fusionar los datos de cada cámara recopilados en la fase anterior vamos a utilizar homografía proyectiva.

El primer paso para calcular la homografía es seleccionar puntos fijos de la escena que sean comunes a las dos vistas. Por esa razón implementamos en MATLAB la función `getPoints`. El funcionamiento de esta función es muy simple, con el ratón seleccionamos puntos fijos de una imagen, y a continuación seleccionamos los mismos puntos fijos vistos desde la otra imagen.

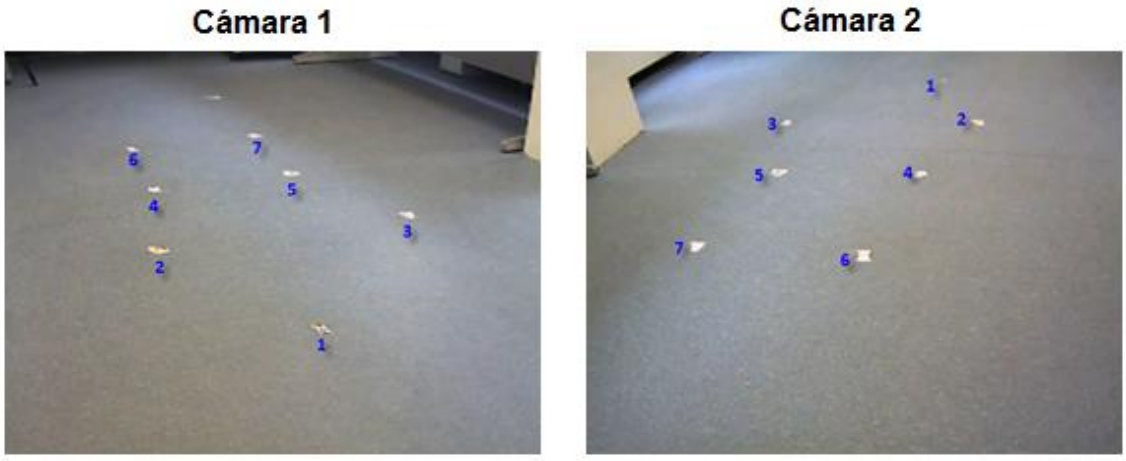

Figura 4.12. Puntos estáticos utilizados para el cálculo de la homografía.

Finalmente, utilizando una transformación homográfica 2D, y usando el algoritmo DTL 2D implementado por Kovesi, `homography2d`[7], calculamos la homografía para mapear puntos de la cámara 1 a la cámara 2 y viceversa,

```
Hom = homography2d(points1, points2);
HomInv = homography2d(points2, points1);
```

conteniendo la variable `points1`, las posiciones de los puntos estáticos seleccionados en la cámara 1 y la variable `points2`, las posiciones de los mismos puntos estáticos seleccionados en la cámara 2.

4.6 Asignación de objetos coincidentes

Para asignar los objetos, implementamos el script MATLAB `asignaObj`. Esta implementación nos permite detectar si un objeto ha sido localizado por las dos cámaras o solamente por una de ellas. Gracias a esta información podremos estimar la posición de objetos que solamente están siendo localizados en una vista.

```
[numObj, numObj2] = asignaObj(box, box2, Hom, nobjects, nobjects2, disObj)
```

Los parámetros de entrada de la función `asignaObj` son:

- `box`: recuadros que delimitan cada uno de los objetos almacenados de la cámara 1.
- `box2`: recuadros que delimitan cada uno de los objetos almacenados de la cámara 2.
- `Hom`: matriz de homografía que mapea puntos de la cámara 1 a la cámara 2.
- `nobjects`: número de objetos almacenados de la cámara 1.
- `nobjects2`: número de objetos almacenados de la cámara 2.

[7] http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/Projective/homography2d.m

- `disObj`: umbral que determina si dos objetos detectados en dos cámaras distintas corresponden al mismo objeto físico en el escenario.

Los parámetros de salida de la función asignaObj son:

- `numObj`: vector de correspondencias de los objetos de la cámara 1.
- `numObj2`: vector de correspondencias de los objetos de la cámara 2.

Para asignar los objetos de ambas cámaras se sigue el algoritmo descrito en el apartado *3.3.2*.

4.7 Recuperación de objetos perdidos

Si un objeto se detecta en una cámara pero en la otra no, calculamos su localización mediante la homografía y la posición del objeto en la cámara en la que aparece.

```
box2{end+1} = Hom * box{objetos(j)};
```

A continuación mostramos un ejemplo de la recuperación de objetos perdidos.

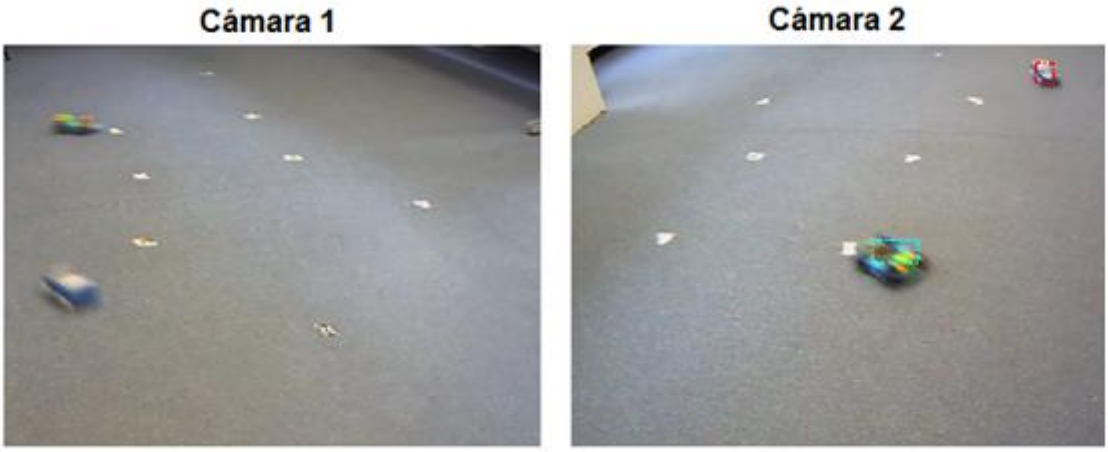

Figura 4.13. Objetos detectados sólo en la cámara 2.

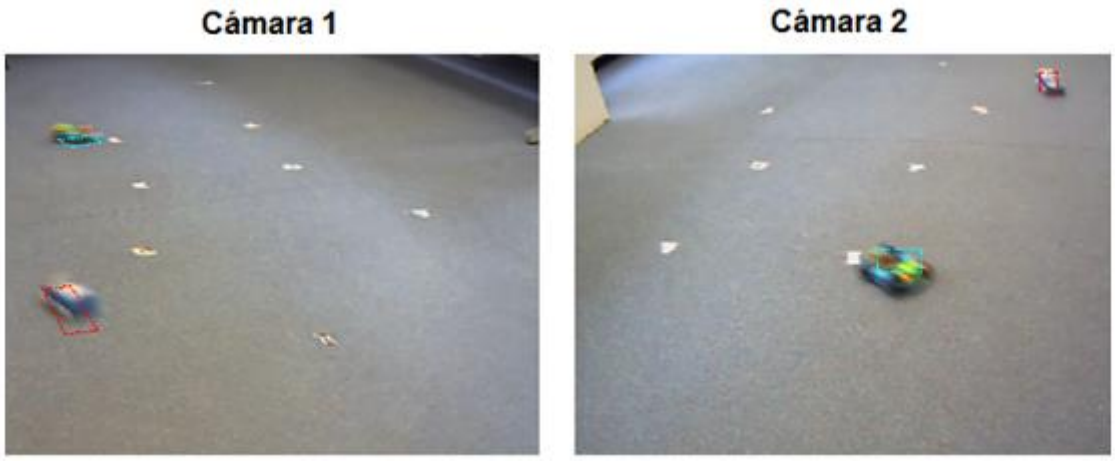

Figura 4.14. Objetos recuperados en cámara 1 mediante su posición en la cámara 2.

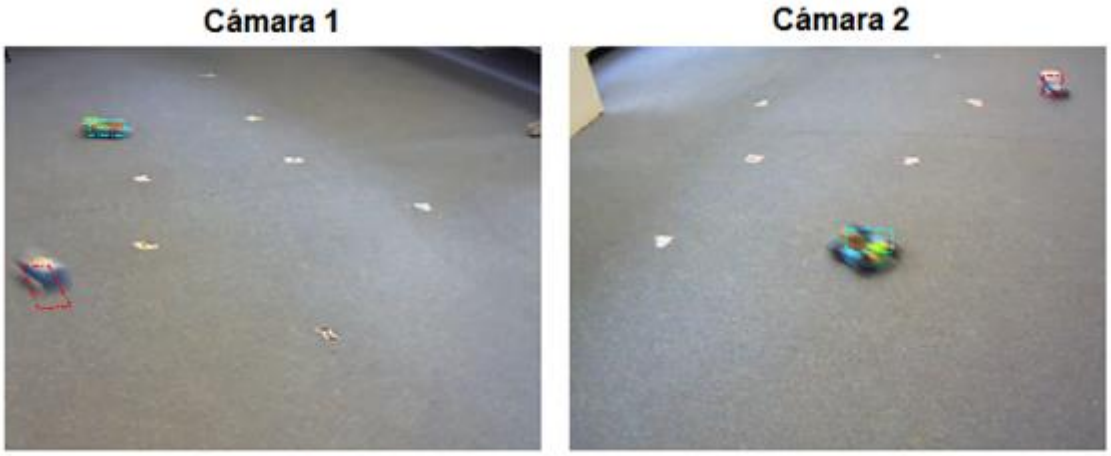

Figura 4.15. El recuadro del objeto rojo de la cámara 1 es estimado utilizando el recuadro del objeto rojo de la cámara 2.

En la figura 4.13, **MODET** solamente ha localizado los dos objetos en movimiento de la escena en la cámara 2. Mediante la función `asignaObj` se detecta este hecho y se estiman las posiciones de estos objetos en la cámara 1 mediante la matriz de homografía, ver figura 4.14. En el siguiente *frame*, ver figura 4.15, **MODET** ha detectado los dos objetos en la cámara 2, pero sólo el objeto azul en la cámara 1. Idénticamente que en el paso anterior, se estima la posición del objeto rojo en la vista 1 utilizando la localización de la vista 2.

4.8 Generación de vídeo

Para facilitar la evaluación de **MODET** en la fase de pruebas y resultados, grabamos cada una de las salidas del algoritmo en un fichero de vídeo. Para realizar las grabaciones del algoritmo en funcionamiento hemos usado la función de MATLAB `VideoWriter`, con la que

obtenemos un vídeo de salida en formato avi, con una velocidad de 15 fps y una calidad del 75%.

```
writerObj = VideoWriter(videoSalida);
writerObj.Quality = 75;
writerObj.FrameRate = 15;
```

En la tabla 4.1, recogemos todos los parámetros necesarios para configurar el algoritmo **MODET**. Los valores de los parámetros mostrados serán estudiados y escogidos en la fase de entrenamiento de cada escenario.

Tabla 4.1. Parámetros del algoritmo MODET, escogidos en la fase de entrenamiento.

Fase	Parámetro	Valor típico mínimo	Valor típico máximo
Extracción de puntos SURF	SURFThreshold	100	2000
Detección	Models	1	[1,2,3]
	dist_thresh	1	3
	kCluster	5	25
	numMinPoints	30	50
Seguimiento	minScore	-2	-20
	numMaxPointsTracking	200	2000
Fusión de datos	disObj	200	300

5 PRUEBAS Y RESULTADOS

A lo largo de este capítulo se van a presentar los diferentes resultados experimentales del sistema de detección y seguimiento de objetos **MODET**, descrito en los capítulos anteriores. El conjunto de pruebas se enfocará principalmente en evaluar el funcionamiento del sistema en distintos escenarios. Todas las pruebas realizadas se pueden visualizar en la página web del proyecto[8].

Para evaluar el objetivo final del proyecto, desarrollar un sistema de detección y seguimiento de objetos que utilice puntos de interés, se comparará la posición de los objetos obtenidos con la posición real de los mismos. En la elaboración de las pruebas se han utilizado dos grupos de escenarios diferentes: interior y exterior.

Las primeras pruebas se han realizado sobre secuencias grabadas por nosotros mismos en un escenario interior controlado, como mostramos en la tabla 5.1. Este escenario se define como un espacio cerrado y con iluminación artificial, grabado por dos cámaras fijas que generan unas imágenes de tamaño 640x480 píxeles. Sobre este escenario hemos realizado dos tipos de pruebas. La primera prueba, realizada sobre secuencias con un objeto en movimiento, se centra en la comparación de nuestro algoritmo de tracking con Mean Shift [32]. En la segunda prueba se evalúa el funcionamiento de **MODET** sobre vídeos con varios objetos en movimiento a la vez.

Una vez comprobado el funcionamiento de **MODET** en un entorno controlado, se valora el algoritmo sobre secuencias de mayor dificultad grabadas en escenarios exteriores. Para la tercera prueba se utilizan secuencias de vídeos de la base de datos de PETS 2001[9]. Estas secuencias de vídeo han sido grabadas con dos cámaras fijas e intervienen distintos objetos en movimiento, como son grupos de personas o coches en movimiento. La cuarta prueba se ha realizado sobre secuencias de vídeo grabadas con una cámara en movimiento. En esta prueba no hemos implementado el último módulo de **MODET**, correspondiente a la mejora de **MODET** utilizando fusión de datos entre varias cámaras, debido a que esta mejora sólo es válida para secuencias con fondo estático y además sólo hay una cámara.

Finalmente hemos probado en vídeos de otros dominios para verificar la generalidad de nuestro enfoque.

[8] http://modet.clegua.es
[9] http://www.cvg.rdg.ac.uk/slides/pets.html

Tabla 5.1. Escenarios utilizados en las pruebas.

# de prueba	Escenario	Movimiento de la cámara	# de secuencias	# de objetos
1	Interior	Cámara fija	8	Uno
2	Interior	Cámara fija	5	Hasta cinco
3	Exterior	Cámara fija	4	Hasta tres
4	Exterior	Cámara móvil	3	Uno
5	Exterior	Cámara fija/móvil	-	-

Una vez escogidos los escenarios que intervienen en la evaluación del algoritmo, se calibran todos los parámetros que intervienen en **MODET** para cada escenario. Esta calibración se realiza en la fase de entrenamiento.

5.1 Fase de entrenamiento

En la fase de entrenamiento se ajustan todos los parámetros de **MODET** vistos en la implementación (tabla 4.1), ya que es el paso previo a realizar en las pruebas de cada escenario. Una vez se han decidido los valores óptimos de los parámetros, éstos se fijan y se utilizan para todas las secuencias grabadas en el mismo escenario. Para ajustar los parámetros utilizamos una de las secuencias. Con esos parámetros testeamos el resto de las secuencias para valorar la calidad de calibración.

Para determinar los valores de los parámetros hay que realizar un estudio del entorno del escenario y las cámaras a utilizar. Los parámetros dependerán del tamaño de la imagen generada, la calidad de definición de la misma, el tipo de objetos que intervienen, el ruido del entorno y la rigidez de la cámara (móvil o fija).

A continuación, vamos a discutir los elementos que intervienen en la elección de los valores de cada parámetro a calibrar.

- **SURFThreshold:** Este parámetro es común para la fase de detección y la fase de seguimiento. Corresponde al umbral que selecciona los puntos de interés SURF más robustos. Para vídeos con una alta calidad y un gran tamaño de imagen se utilizará un valor de `SURFThreshold` alto. Para secuencias de vídeos con un menor tamaño de imagen y calidad se fijará un valor de `SURFThreshold` bajo, y así disponer del número de puntos de interés suficientes con los que obtener resultados satisfactorios.
- **Models:** Corresponde al conjunto de modelos de transformación de fondo de la fase de detección. Para escenas con fondo fijo se utilizará un modelo de transformación de traslación pura y para escenarios con fondo en movimiento, un conjunto de modelos que abarca la traslación pura, la transformación de similitud y la transformación afín.
- **dist_thresh:** Es el valor del umbral de RAMOSAC que diferencia las correspondencias que siguen el modelo de transformación (*inliers*) y las

correspondencias que no lo siguen (*outliers*). Cuanto mayor sea el umbral se necesitará una mayor diferencia entre el modelo de transformación del punto y el modelo de transformación de consenso, para que sea considerado como *outlier*.

- **kCluster:** Esta variable determina la distancia mínima de corte entre los grupos de puntos en movimiento existentes en la escena. Sólo se utiliza en la fase de detección. Su valor depende del tamaño del objeto y de la densidad de puntos de interés en la imagen.
- **numMinPoints:** Indica el número mínimo de puntos de un grupo necesarios para formar un objeto. Cuanto mayor sea la densidad de puntos de interés de la escena mayor debe ser la exigencia de puntos para conformar un objeto.
- **minScore:** Índica la puntuación de RAMOSAC mínima en la transformación que sufre el objeto para que se considere válida y el objeto no sea eliminado.
- **numMaxPointsTracking:** Número máximos de puntos de interés del objeto utilizados en la fase de seguimiento. Si el número de puntos de interés almacenado es menor que numMaxPointsTracking, se utilizarán todos los puntos. Si el número de puntos de interés almacenado es mayor que numMaxPointsTracking, se descartarán los que tengan una menor puntuación.
- **disObj:** Este parámetro contiene el umbral de la distancia entre correspondencias de objetos. Se utiliza en la etapa de fusión de información, para decidir si un objeto de la cámara 1 y otro de la cámara 2 corresponden al mismo objeto físico.

En la tabla 5.2 mostramos el factor de dependencia de todos los parámetros a calibrar en el sistema.

Tabla 5.2. Clasificación de los parámetros a calibrar en la fase de entrenamiento.

Fase	Parámetro	Factor de dependencia
Extracción de puntos SURF	`SURFThreshold`	Calidad de definición de imagen.
Detección	`Models`	Rigidez de la cámara.
	`dist_thresh`	Ruido de entorno.
	`kCluster`	Tamaño del objeto y densidad de puntos de interés.
	`numMinPoints`	Tamaño del objeto y densidad de puntos de interés.
Seguimiento	`minScore`	Tipo de movimiento de los objetos.
	`numMaxPointsTracking`	Densidad de puntos de interés.
Fusión de datos	`disObj`	Altura de los objetos.

En esta fase también se definirá la transformación homográfica para escenarios grabados con dos cámaras. Para ello, se seleccionan manualmente los puntos de referencia del

escenario vistos por las dos cámaras que posteriormente utilizaremos para calcular la homografía entre ambas vistas.

Una vez que hemos fijado todos los parámetros procedemos a obtener y evaluar los resultados de todas las secuencias de vídeo.

5.2 Escenario interior

En esta sección se van a desarrollar los bloques **Prueba 1** y **Prueba 2**. Los vídeos para estas pruebas han sido grabados en el mismo escenario interior. Como podemos ver en la recreación de la figura 5.1, este escenario consiste en una habitación iluminada artificialmente y con un espacio central en el que no hay obstáculos. En el centro del escenario hemos marcado 7 puntos que se emplearán para el cálculo de la homografía proyectiva entre las dos cámaras. Las dos cámaras están situadas en dos esquinas de la habitación y sus ángulos de visión tienen áreas solapadas.

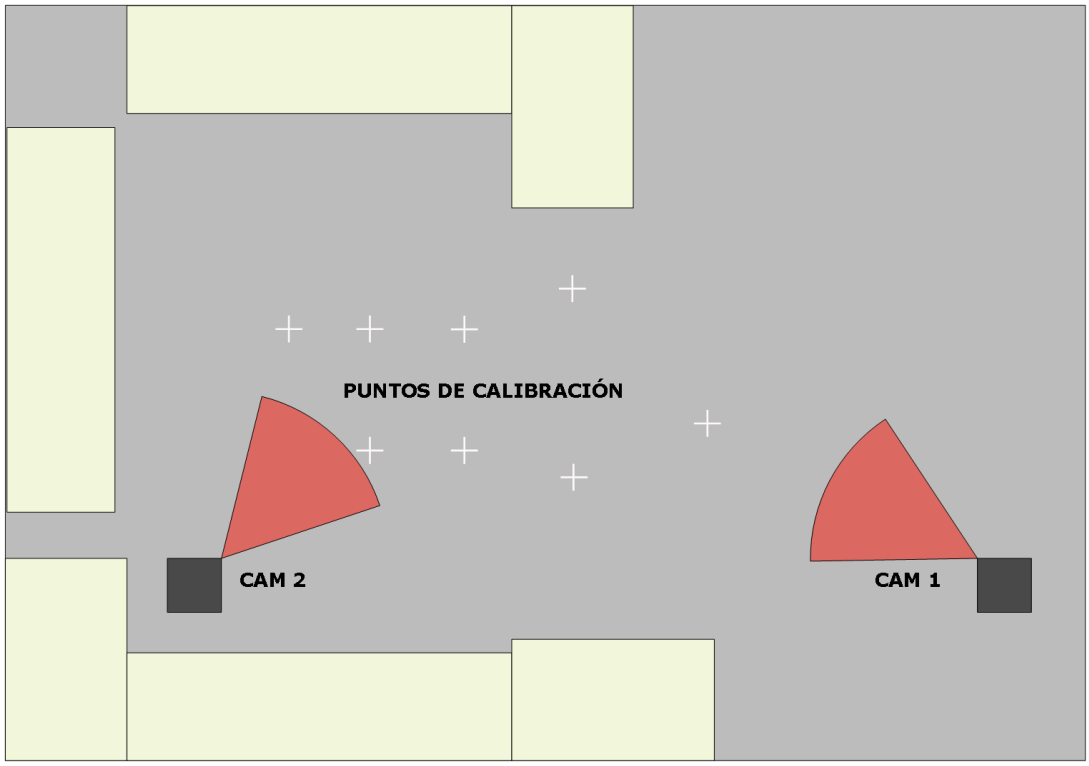

Figura 5.1. Recreación del escenario interior para realizar las pruebas.

A lo largo de este apartado describiremos el tipo de cámaras utilizadas en la grabación, los objetos que intervienen en la secuencia de vídeo y los parámetros escogidos en la fase de entrenamiento realizada.

5.2.1 Parámetros de MODET

Los parámetros escogidos se pueden observar en la tabla 5.3 y se mantendrán en todas las secuencias de la Prueba 1 y Prueba 2. Para efectuar esta calibración se han utilizado los primeros *frames* del Vídeo 5 de la Prueba 1.

Tabla 5.3. Valores de los parámetros de MODET escogidos en la fase de entrenamiento para el escenario interior.

Fase	Parámetro	Valor
Extracción de puntos SURF	SURFThreshold	100
Detección	Models	1
	dist_thresh	1
	kCluster	40
	numMinPoints	6
Seguimiento	minScore	-2
	numMaxPointsTracking	200
Fusión de datos	disObj	250

Además, como hemos visto en la figura 5.1, hemos fijado 7 puntos en el plano del suelo de la escena, para obtener la transformación proyectiva.

A continuación, vamos a detallar el tipo de cámara utilizado y los objetos que intervienen en estos escenarios.

5.2.2 Cámaras IP

Para la grabación de los vídeos sobre el escenario interior hemos utilizado dos cámaras IP. Hemos elegido el modelo DCS 3110 de la marca D-Link (figura 5.2). Cada una de estas cámaras tiene un precio aproximado de 340€. Estas cámaras nos proporcionan una velocidad de grabación de hasta 30 fps para imágenes de 640x480 píxeles con una buena calidad.

Los vídeos generados para este escenario, tienen un tamaño de imagen 640x480 píxeles y una velocidad de fotogramas por segundo para la cámara 1 de 15 fps y para la cámara 2 de 23 fps. Esta diferencia entre las velocidades de *frames* por segundo se debe a la diferencia de iluminación entre cámaras.

Figura 5.2. Cámara D-Link DCS 3110.

Antes de poder realizar la grabación hemos tenido que conectar las cámaras a la red, permitiéndonos la opción de grabar desde cualquier lugar con acceso a internet.

5.2.3 Objetos

En este primer bloque de pruebas, hemos utilizado 4 objetos distintos. Todos ellos han sido empleados para las pruebas con un objeto y múltiples objetos sobre el escenario interior. En la tabla 5.4 mostramos las imágenes de los objetos.

Tabla 5.4. Objetos que intervienen en las pruebas sobre el escenario interior.

Objeto 1		
Objeto 2		
Objeto 3		
Objeto 4		

A continuación, vamos a explicar con detalle las pruebas que se han realizado en este escenario.

5.3 Prueba 1: Escenario interior, cámara fija y un objeto

Las primeras pruebas realizadas, consisten en vídeos sencillos en los que aparece un sólo objeto. Hemos grabado 8 secuencias. Cada secuencia ha sido grabada con las dos cámaras,

obteniendo por tanto, 16 videos. Todas las secuencias de esta prueba se pueden visualizar en la web del proyecto[10]. En la tabla 5.5 listamos las características de cada secuencia.

Tabla 5.5. Características de las secuencias con un objeto en movimiento.

Secuencia	Objeto	Duración (s)	Velocidad en cámara 1 (fps)	Velocidad en cámara 2 (fps)	Tamaño de imagen (píxeles)	# de objetos
1	1	3	15	23	640x480	1
2	1	3	15	23	640x480	1
3	2	4	15	23	640x480	1
4	2	3	15	23	640x480	1
5	3	3	15	23	640x480	1
6	3	3	15	23	640x480	1
7	4	3	15	23	640x480	1
8	4	6	15	23	640x480	1

Para evaluar **MODET**, en cada vídeo, vamos a comparar el centro de masas del objeto que se detecta y estima, con la posición real del centro del objeto. Para seleccionar la posición real, hemos implementado un script de MATLAB en el que escogemos manualmente la posición del objeto en cada fotograma. Además, con este conjunto de posiciones tenemos la trayectoria real del objeto a lo largo del vídeo. Para estimar la posición del objeto, devuelta por **MODET**, ejecutamos el algoritmo diseñado e implementado en MATLAB. Con ello, también conseguimos la trayectoria **MODET** del objeto. Después de tener ambas trayectorias medimos el error medio, la desviación típica y la correlación. Para evaluar la eficacia del algoritmo de detección, contamos para cada vídeo el número de fotogramas en los que no se detecta el objeto y sí aparece.

Una vez que hemos evaluado **MODET** lo comparamos con Mean Shift. Para evaluar los dos algoritmos en igualdad de condiciones y debido a que en Mean Shift es necesario fijar la posición inicial del objeto a seguir, vamos a utilizar como punto de partida el área rectangular que delimita el objeto cuando se detecta por primera vez en **MODET**. Por tanto, volvemos a cotejar la posición real (x, y) del centro del objeto con el estimado por Mean Shift, calculando el error medio, la desviación típica y la correlación.

Las figuras y tablas que se van a mostrar a lo largo de esta sección corresponden a la secuencia 1 de la Prueba 1. En la figura 5.3 se muestra la trayectoria que sigue el objeto en esta secuencia sobre el escenario interior.

[10] http://modet.clegua.es/prueba1.html

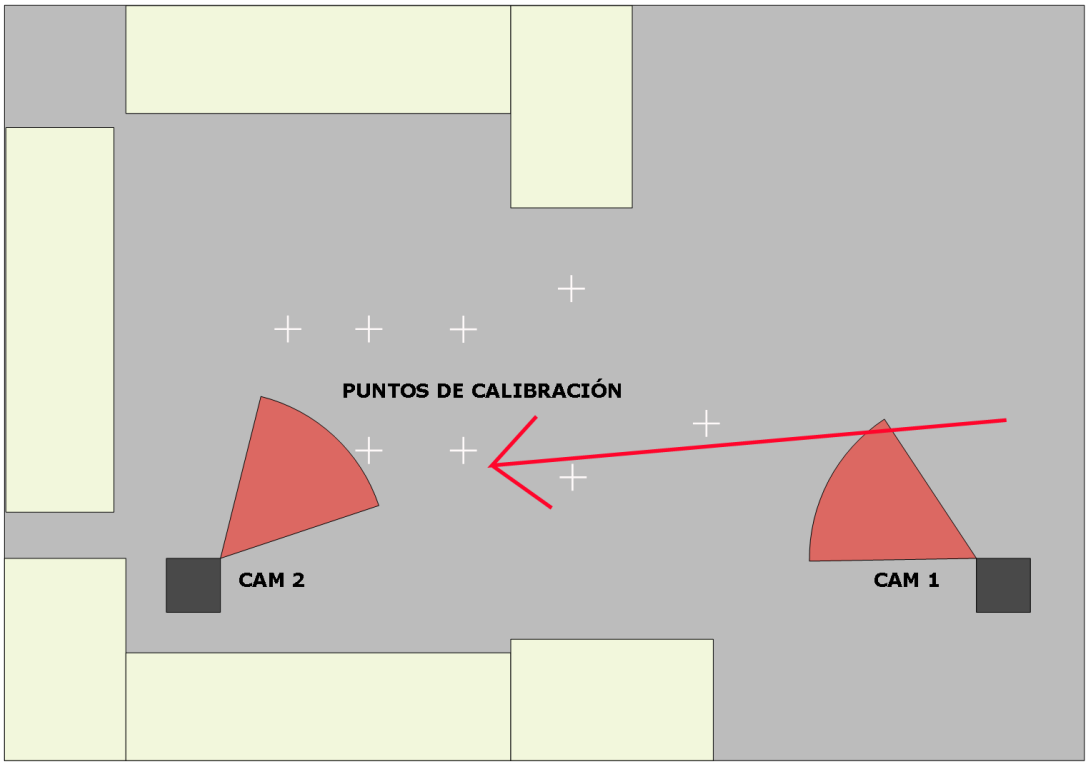

Figura 5.3. Recreación de la trayectoria del objeto en la secuencia 1 sobre el escenario interior.

En la figura 5.4, dibujamos sobre la imagen captada por las cámaras la trayectoria real (línea azul), la trayectoria **MODET** (círculo rojo), la trayectoria Mean Shift (círculo verde) y la proyección de la trayectoria **MODET** en la cámara opuesta (cruz amarilla). Podemos observar en la cámara 1 que la trayectoria Mean Shift y **MODET** son muy similares. Sin embargo, la proyección de la trayectoria **MODET** de la vista opuesta (cruz amarilla) contiene un pequeño error a causa de que la posición central del rectángulo que delimita el objeto, no se encuentra en el plano de los puntos de referencia (plano z=0) utilizados para estimar la transformación de la homografía, ya que el objeto tiene una altura respecto a este plano. Cabe recordar que la trayectoria representada por las cruces amarillas, será la que se utilice para el algoritmo **MODET** Mejorado en aquellos *frames* que el objeto no sea detectado. También se puede observar como en la cámara 2, Mean Shift pierde el objeto. Detallaremos la causa de esta pérdida con posteriores gráficas comparativas.

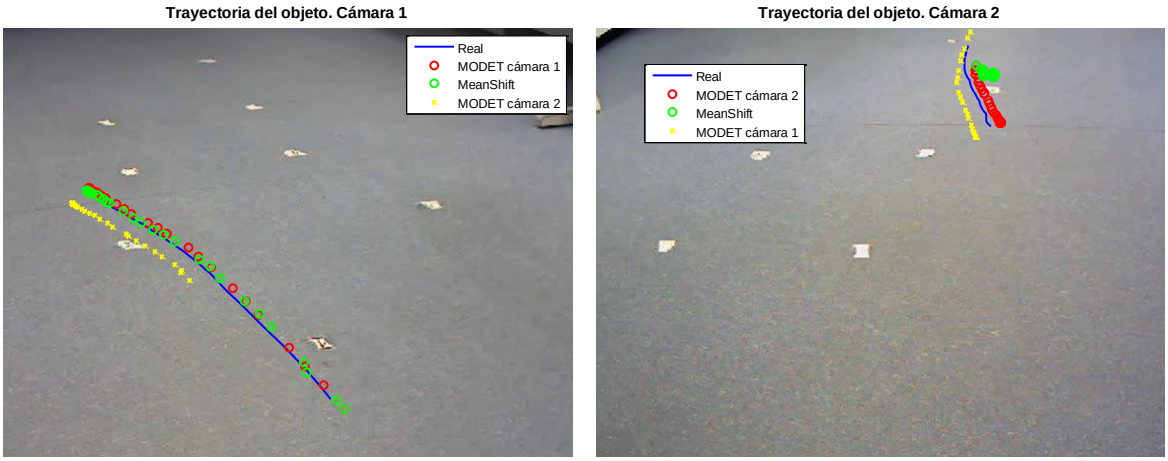

Figura 5.4. Trayectorias del objeto a lo largo de la secuencia 1.

En la figura 5.5 y 5.6, se muestra la comparativa entre la posición real, Mean Shift y **MODET** desde cada cámara.

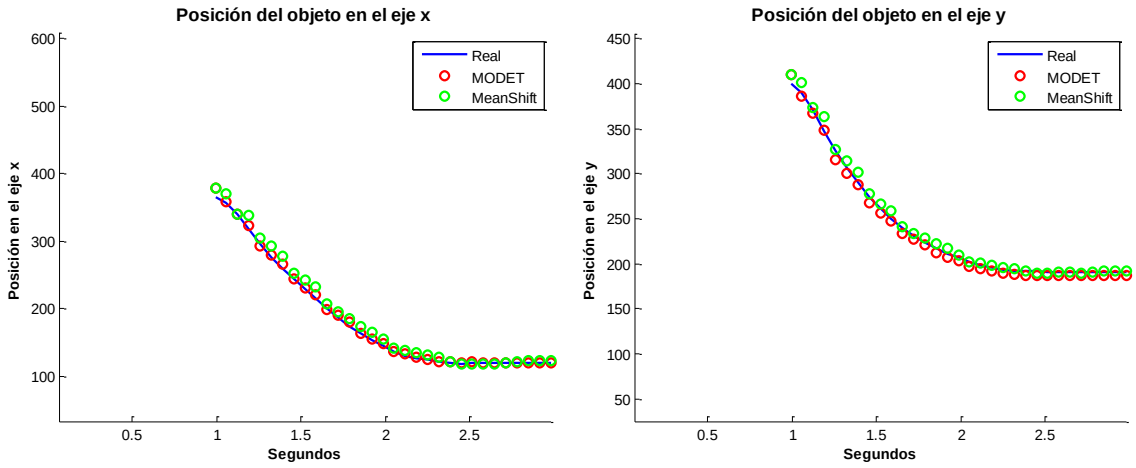

Figura 5.5. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

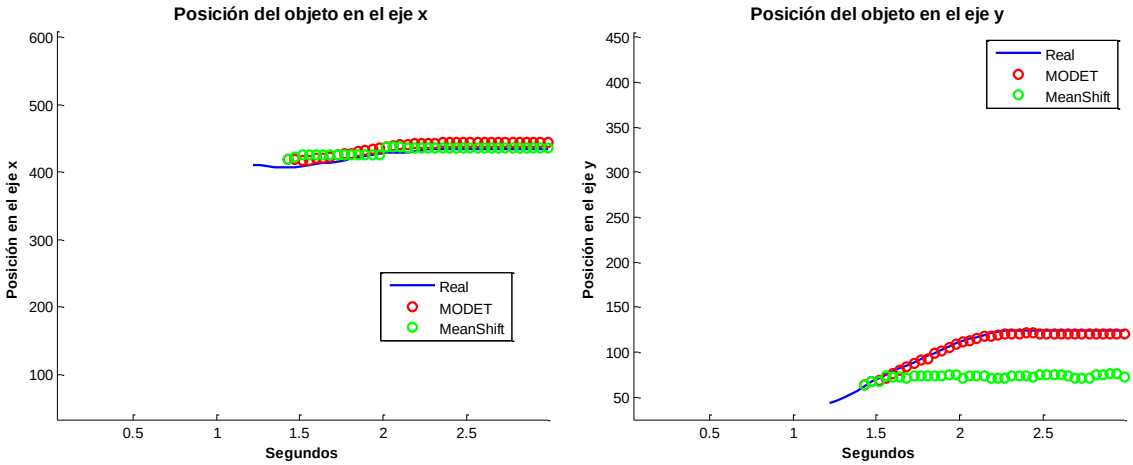

Figura 5.6. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

La figura 5.5, deja de manifiesto que los resultados que consigue **MODET** son muy semejantes a los del algoritmo Mean Shift. En la figura 5.6, debido a que la superficie que delimita al objeto contiene parte del fondo del escenario (ver figura 5.11), el algoritmo Mean Shift pierde la trayectoria real del objeto.

Posteriormente, se analizan las pruebas con la mejora de **MODET**, que se utiliza en el supuesto de que el objeto sólo haya sido detectado por una de las cámaras. Esta mejora consiste en obtener la posición del objeto en una cámara, proyectando mediante homografía la localización de este objeto en la otra cámara. Realizamos los mismos cálculos que para **MODET** y Mean Shift.

En las figuras 5.7 y 5.8 se comparan las trayectorias real, **MODET** y **MODET** Mejorado desde cada cámara.

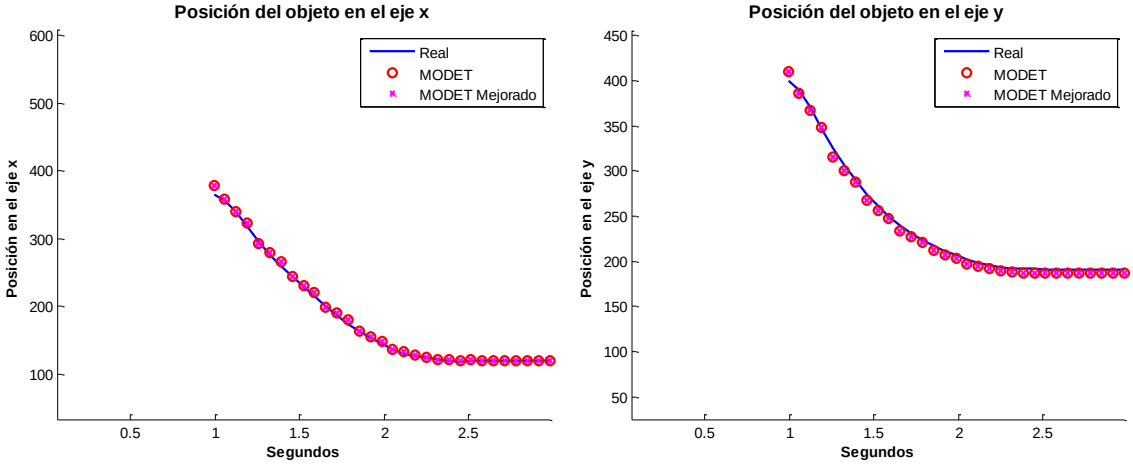

Figura 5.7. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

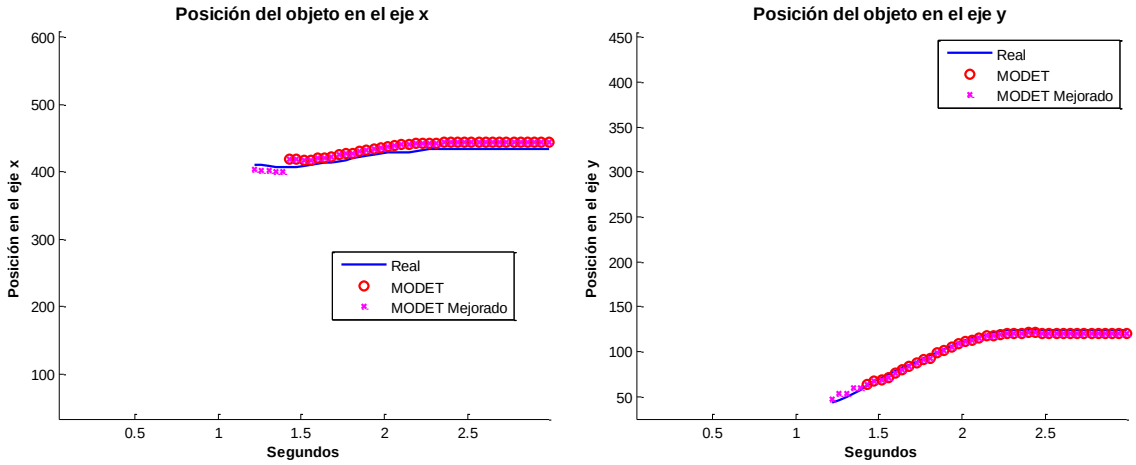

Figura 5.8. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

En la figura 5.7 no se observan diferencias entre los dos algoritmos debido a que el objeto se detecta en todos los fotogramas. Sin embargo, en la figura 5.8, vemos cómo **MODET** Mejorado proyecta la posición del objeto en los 5 primeros fotogramas en los que **MODET** no detecta el objeto. Esta proyección se ajusta con gran precisión a la trayectoria real de objeto.

En las tablas 5.6 y 5.7, mostramos los cálculos realizados para estos algoritmos. Para realizar los cálculos del error medio, la desviación típica y la correlación hemos utilizado las siguientes fórmulas (tanto el error medio como la desviación típica se miden en píxeles):

$$E_M = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{5.1}$$

$$d_i = \|\boldsymbol{w}_i - \boldsymbol{w}_i'\| \tag{5.2}$$

siendo $\boldsymbol{w}_i = (x_i, y_i)$ la posición real del objeto, $\boldsymbol{w}_i' = (x_i', y_i')$ la posición del objeto devuelta por el algoritmo y n el número total de *frames* en los que aparece el objeto.

$$\sigma_E = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - E_M)^2} \tag{5.3}$$

siendo d_i la distancia en el instante i, E_M el error medio y n el número total de *frames* en los que aparece el objeto.

$$corr_x = \frac{cov(x_i, x_i')}{\sigma_{x_i}\sigma_{x_i'}} \tag{5.4}$$

$$corr_y = \frac{cov(y_i, y_i')}{\sigma_{y_i}\sigma_{y_i'}} \tag{5.4}$$

siendo para el cálculo de la correlación en el eje x, $cov(x_i, x_i')$ la covarianza entre la posición real en el eje x del objeto, x_i, y la posición del objeto en el eje x devuelta por el algoritmo, x_i', y $\sigma_{x_i}\sigma_{x_i'}$ las desviaciones típicas de la trayectoria real del objeto y la trayectoria del objeto devuelta por el algoritmo para el eje x. Exactamente igual para el cálculo de la correlación en el eje y.

Tabla 5.6. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/31	0/31	0/31
Error medio	5.64 ± 2.54	8.82 ± 6.78	5.64 ± 2.54
Correlación en eje x	0.9994	0.9985	0.9994
Correlación en eje y	0.9991	0.9990	0.9991

Tabla 5.7. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	5/43	5/43	0/43
Error medio	9.92 ± 1.06	39.17 ± 14.67	9.82 ± 1.14
Correlación en eje x	0.9934	0.8950	0.9436
Correlación en eje y	0.9989	0.5643	0.9982

En las tablas se aprecia que **MODET** obtiene mejores resultados que Mean Shift para esta secuencia. En la tabla 5.6 tanto **MODET** como Mean Shift siguen el objeto correctamente si bien el error de MODET es ligeramente inferior. En la tabla 5.7, la diferencia es mayor debido a que el algoritmo Mean Shift pierde el objeto según se aprecia en la figura 5.6. Por otro lado, en la cámara 2 (tabla 5.7) **MODET** tarda 5 *frames* en detectar el objeto. En cambio, con **MODET** Mejorado conseguimos proyectar la posición del objeto en estos 5 *frames*, de la cámara 1 a la cámara 2. A pesar de que se proyecta la posición en los 5 fotogramas, disminuye ligeramente el error medio, aunque aumenta la desviación típica y disminuye la correlación con respecto a **MODET**.

En las figuras 5.9 y 5.10, podemos observar la posición real del objeto frente a la posición **MODET** del mismo. Los resultados obtenidos en estas figuras, se aproximan a una recta que forma un ángulo de 45^{o} con respecto ambos ejes, dejando de manifiesto que la correlación entre la trayectoria real y la **MODET** se puede considerar 1.

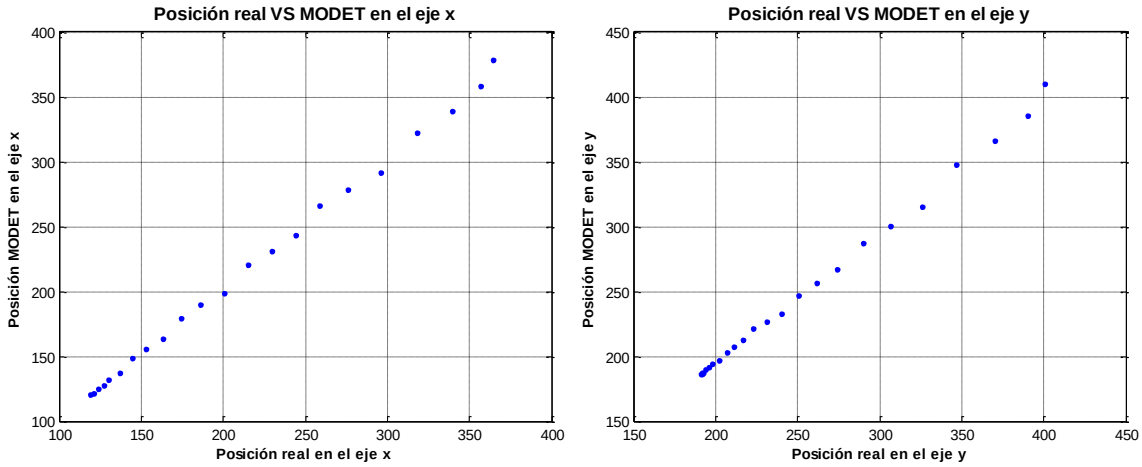

Figura 5.9. Posición real del objeto frente a posición MODET del objeto en la cámara 1.

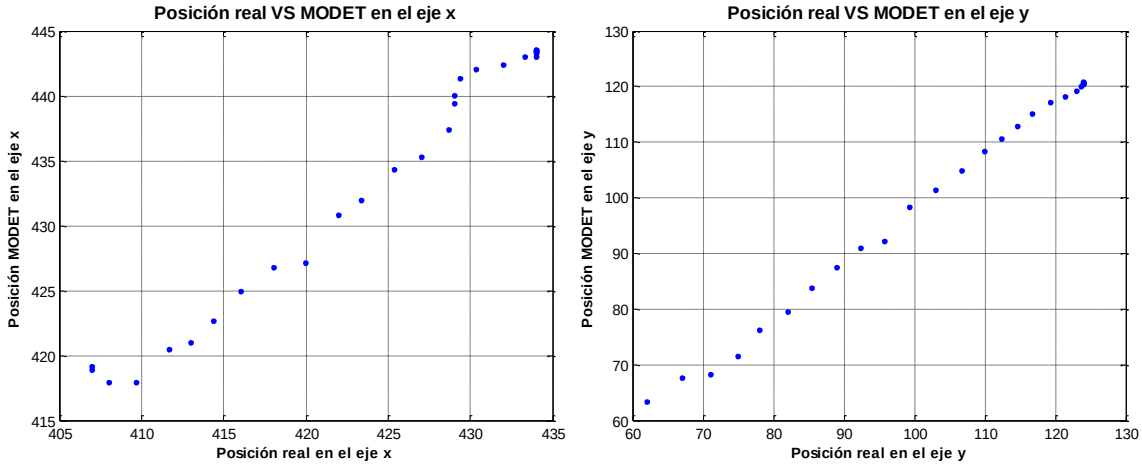

Figura 5.10. Posición real del objeto frente a posición MODET del objeto en la cámara 2.

Por último, se muestra la evolución del rectángulo que delimita el objeto. En la figura 5.11, se observa cómo el rectángulo sufre una transformación de similitud aplicada por el algoritmo durante la fase de seguimiento.

Figura 5.11. Delimitación del objeto por MODET a lo largo de la secuencia 1.

Como hemos mencionado, las figuras y tablas anteriormente mostradas corresponden a la secuencia 1. Todas las tablas y figuras obtenidas para el resto de secuencias se recogen en el anexo *A.1* y pueden verse en la web del proyecto. A continuación, vamos a mostrar las tablas que engloban el conjunto de estos resultados.

Tabla 5.8. Resultados del algoritmo MODET.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	0/31	5.64 ± 2.54	0.9994	0.9991
	C2	5/43	9.92 ± 1.06	0.9934	0.9989
Secuencia 2	C1	4/30	6.96 ± 1.25	0.9993	0.9952
	C2	1/56	5.14 ± 2.57	0.9997	0.9992
Secuencia 3	C1	1/47	7.72 ± 2.35	0.9994	0.9997
	C2	3/29	2.14 ± 1.26	0.9999	0.9997
Secuencia 4	C1	0/27	6.29 ± 2.58	0.9996	0.9992
	C2	0/59	8.65 ± 1.57	0.9998	0.9997
Secuencia 5	C1	0/30	20.45 ± 4.98	0.9988	0.9985
	C2	0/21	4.49 ± 2.52	0.9997	0.9997
Secuencia 6	C1	0/29	5.74 ± 3.11	0.9997	0.9987
	C2	0/52	17.54 ± 2.81	0.9995	0.9995
Secuencia 7	C1	1/25	6.78 ± 3.38	0.9983	0.9993
	C2	3/24	4.31 ± 2.81	0.9995	0.9997
Secuencia 8	C1	0/76	2.21 ± 2.38	0.9999	0.9998
	C2	9/95	3.12 ± 1.03	0.9997	0.9999
Promedio		27/674	7.08 ± 5.52	0.9989	0.9977

Tabla 5.9. Resultados del algoritmo Mean Shift.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	0/31	8.82 ± 6.78	0.9985	0.9990
	C2	5/43	39.17 ± 14.67	0.8950	0.5643
Secuencia 2	C1	4/30	8.49 ± 3.07	0.9979	0.9727
	C2	1/56	15.31 ± 6.26	0.9980	0.9956
Secuencia 3	C1	0/47	9.65 ± 3.46	0.9982	0.9996
	C2	3/29	13.74 ± 4.20	0.9982	0.9989
Secuencia 4	C1	0/27	8.48 ± 3.24	0.9991	0.9970
	C2	0/59	12.16 ± 5.64	0.9988	0.9984
Secuencia 5	C1	0/30	303.73 ± 118.07	-0.7423	0.8517
	C2	0/21	5.49 ± 2.57	0.9990	0.9987
Secuencia 6	C1	0/29	5.62 ± 2.40	0.9996	0.9988
	C2	8/52	17.90 ± 3.83	0.9996	0.9989
Secuencia 7	C1	3/25	5.72 ± 3.35	0.9990	0.9990
	C2	3/24	12.58 ± 10.62	0.9997	0.9988
Secuencia 8	C1	0/76	5.47 ± 2.54	0.9998	0.9984
	C2	9/95	27.33 ± 35.71	0.9861	0.9676
Promedio		36/674	28.11 ± 68.42	0.9311	0.8411

Tabla 5.10. Resultados del algoritmo MODET Mejorado.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	0/31	5.64 ± 2.54	0.9994	0.9991
	C2	0/43	9.82 ± 1.14	0.9436	0.9982
Secuencia 2	C1	0/30	10.34 ± 9.78	0.9953	0.9913
	C2	1/56	5.14 ± 2.57	0.9997	0.9992
Secuencia 3	C1	1/47	7.72 ± 2.35	0.9994	0.9997
	C2	0/29	2.88 ± 2.55	0.9998	0.9986
Secuencia 4	C1	0/27	6.29 ± 2.58	0.9996	0.9992
	C2	0/59	8.65 ± 1.57	0.9998	0.9997
Secuencia 5	C1	0/30	20.45 ± 4.98	0.9988	0.9985
	C2	0/21	4.49 ± 2.52	0.9997	0.9997
Secuencia 6	C1	0/29	5.74 ± 3.11	0.9997	0.9987
	C2	0/52	17.54 ± 2.81	0.9995	0.9995
Secuencia 7	C1	1/25	6.78 ± 3.38	0.9983	0.9993
	C2	0/24	5.05 ± 3.77	0.9994	0.9996
Secuencia 8	C1	0/76	2.21 ± 2.38	0.9999	0.9998
	C2	9/95	3.12 ± 1.03	0.9997	0.9999
Promedio		16/674	7.27 ± 5.88	0.9987	0.9976

Con estos resultados podemos concluir que **MODET** tiene un buen funcionamiento frente a Mean Shift. Queda, por tanto, validado el principal objetivo de esta prueba, la comparación de resultados contra un algoritmo reconocido en este ámbito. Por otra parte, **MODET** Mejorado consigue un porcentaje mayor de detecciones del objeto por *frame*, perdiendo precisión en la estimación de la posición del objeto.

Una vez comparados los resultados de **MODET** con los resultados de otro algoritmo de tracking como es Mean Shift, nos enfocaremos en evaluar **MODET** y **MODET** Mejorado en pruebas con una mayor dificultad.

5.4 Prueba 2: Escenario interior, cámara fija y múltiples objetos

En esta segunda parte vamos a grabar vídeos en los que intervienen varios objetos sobre el mismo escenario interior. Para esta prueba hemos elaborado 5 secuencias, cada una grabada con las dos cámaras. En todas las secuencias vamos a utilizar varios de los objetos enumerados en la tabla 5.4, a diferencia del apartado anterior donde sólo se utilizaba un objeto por secuencia. Estos vídeos se pueden visualizar en la página web del proyecto[11]. En la tabla 5.11 mostramos las características de las secuencias.

Tabla 5.11. Características de las secuencias con varios objetos en movimiento.

Secuencia	Objeto	Duración (s)	Velocidad en cámara 1 (fps)	Velocidad en cámara 2 (fps)	Tamaño de imagen (píxeles)	# de objetos
1	4-3-4-2-2	16	15	25	640x480	5
2	4-3-2-4-2	12	15	25	640x480	5
3	4-3-2	10	15	25	640x480	3
4	4-3-2-1	10	15	25	640x480	4
5	4-3-2-1-2	12	15	25	640x480	5

Cómo se realizó en el apartado anterior, los resultados y gráficas que se mostrarán a lo largo de esta sección, corresponden a la secuencia 1 de la Prueba 2.

En la secuencia 1 de esta prueba intervienen 5 objetos. A cada uno de los objetos se le asigna un color de rectángulo identificativo que lo define durante todo el vídeo. En las figuras 5.12 y 5.13 se muestran los colores asignados a cada objeto, desde la cámara 1 y la cámara 2, respectivamente.

Figura 5.12. Objetos vistos desde la cámara 1.

[11] http://modet.clegua.es/prueba2.html

Figura 5.13. Objetos vistos desde la cámara 2.

En la figura 5.14, se muestra la trayectoria real que siguen los distintos objetos que intervienen en la secuencia 1, representando cada uno de ellos por el color asignado anteriormente.

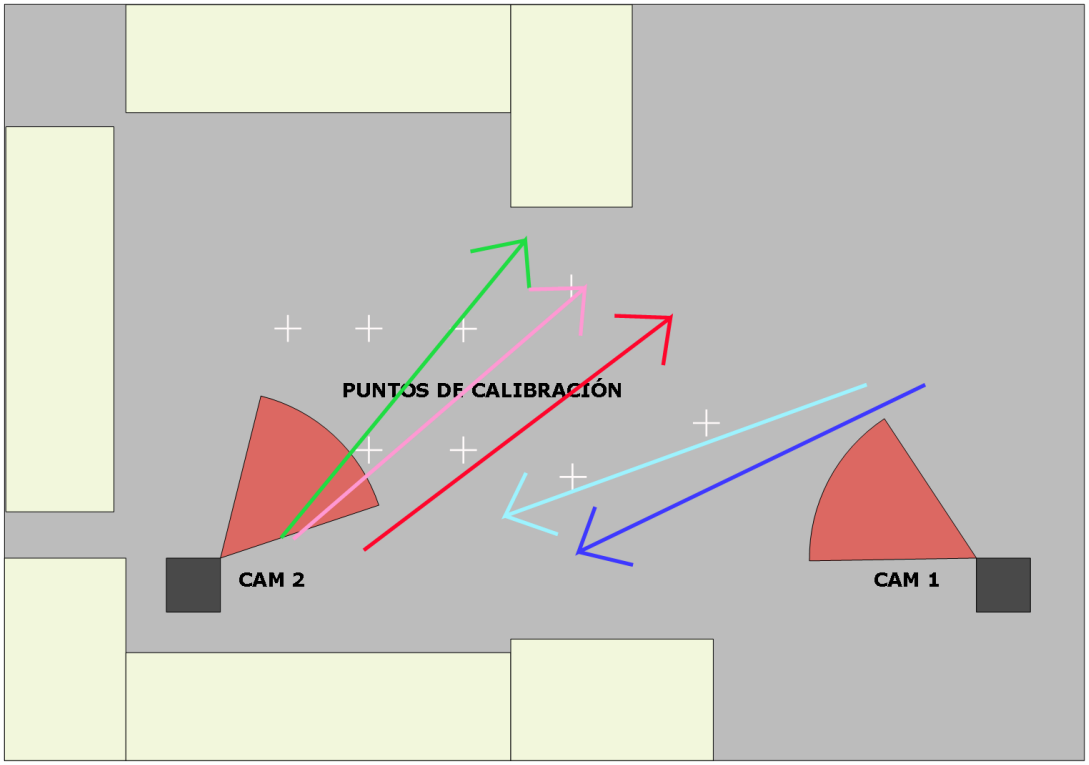

Figura 5.14. Trayectoria de los objetos sobre el escenario de la secuencia 1.

A continuación, en la figura 5.15 se aprecian las trayectorias real, de **MODET** y de **MODET** Mejorado de los objetos.

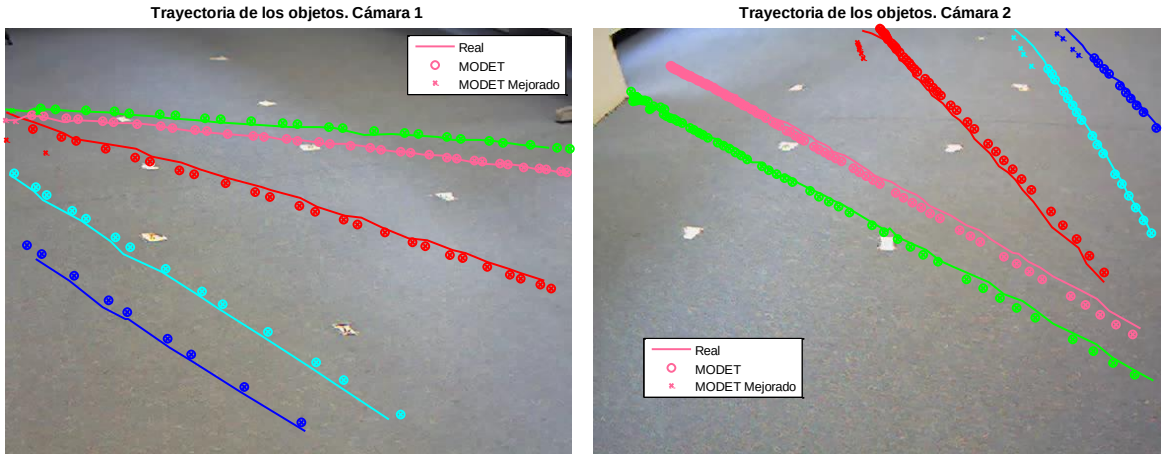

Figura 5.15. Trayectorias de los objetos a lo largo de la secuencia 1.

Según la figura, en la cámara 2 queda de manifiesto cómo para los objetos representados por el color turquesa y azul, al tener poca altura, tienen un pequeño error en los *frames* proyectados por **MODET** Mejorado, sin embargo, para el objeto representado por el color rojo, al tener una mayor altura, **MODET** Mejorado proyecta la posición del objeto con menor precisión. Este hecho también se puede observar en las figuras 5.18 y 5.19, mostradas a continuación.

Para evaluar el algoritmo vamos a usar el mismo procedimiento que en la Prueba 1. Se compara el centro de masas del objeto estimado por los algoritmos **MODET** y **MODET** Mejorado con el punto central de la posición real del objeto.

En las figuras 5.16, 5.17, 5.18 y 5.19, mostramos la comparativa de las trayectorias de cada objeto que intervienen en la secuencia. Las figuras 5.16 y 5.17 corresponden a las trayectorias vistas desde la cámara 1 en el eje x y eje y, respectivamente. Las figuras 5.18 y 5.19, son las vistas desde la cámara 2.

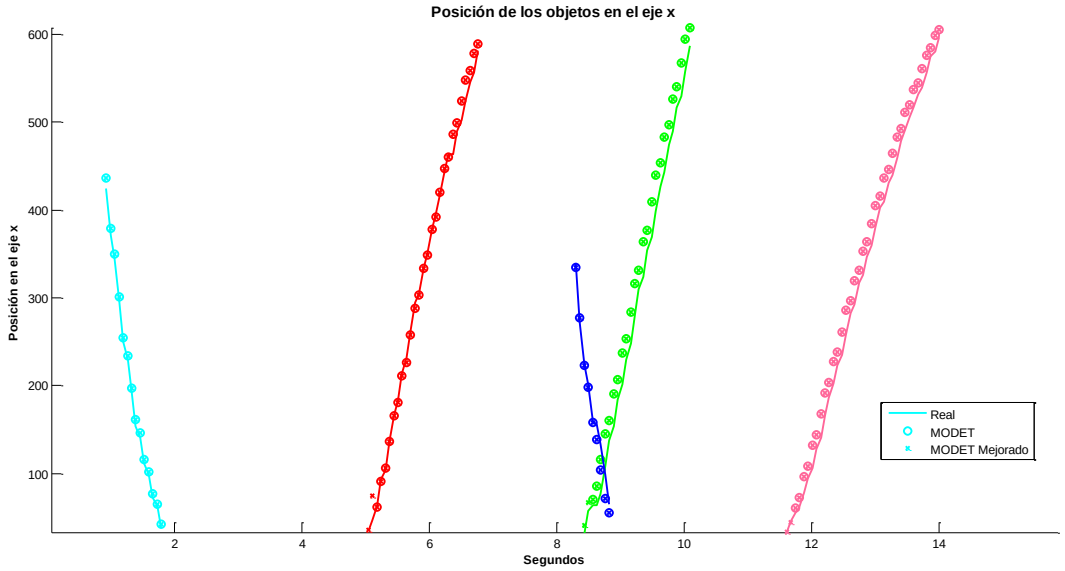

Figura 5.16. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 1.

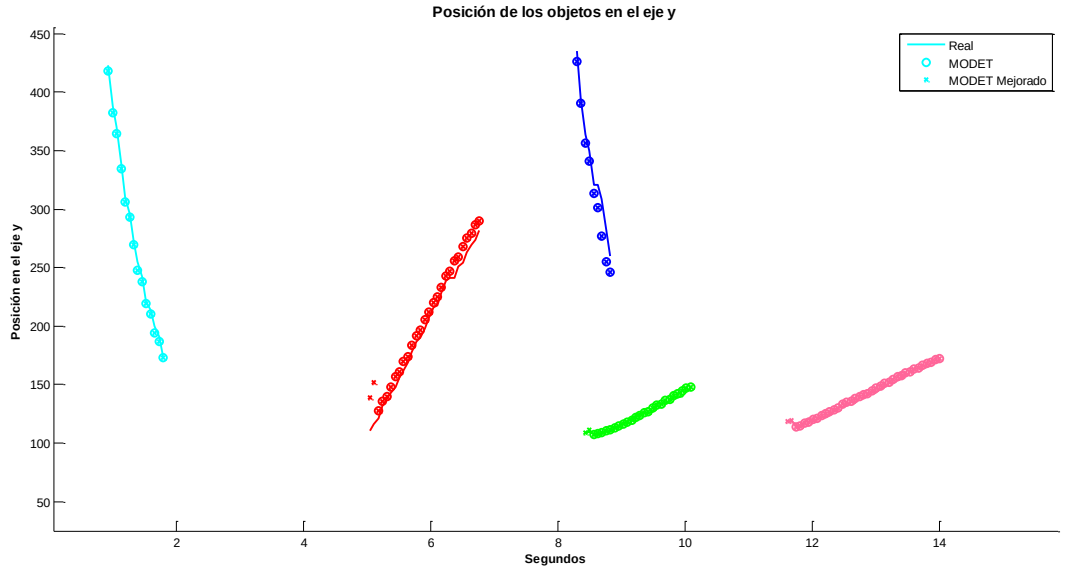

Figura 5.17. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje y desde la cámara 1.

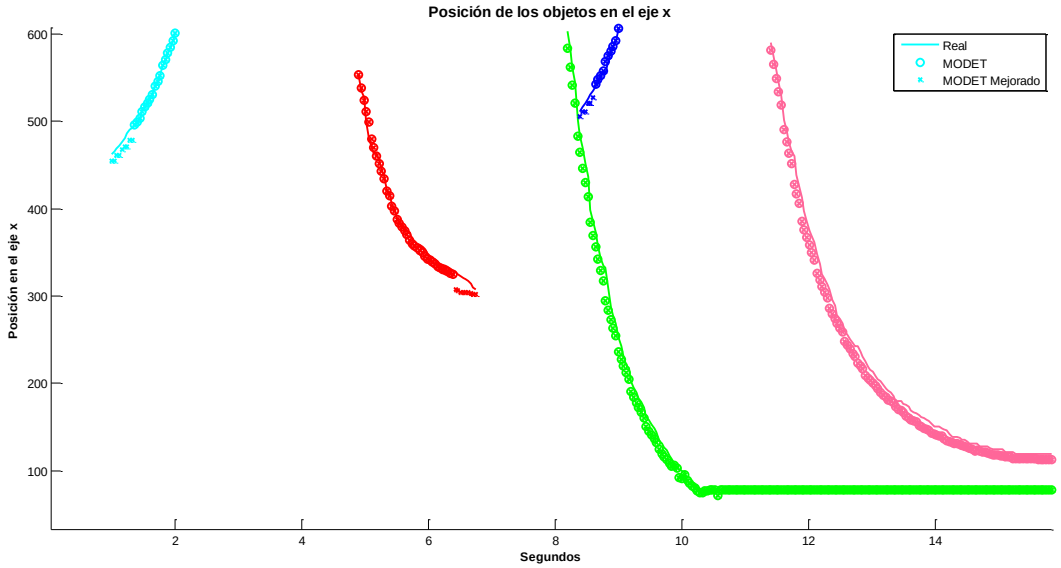

Figura 5.18. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 2.

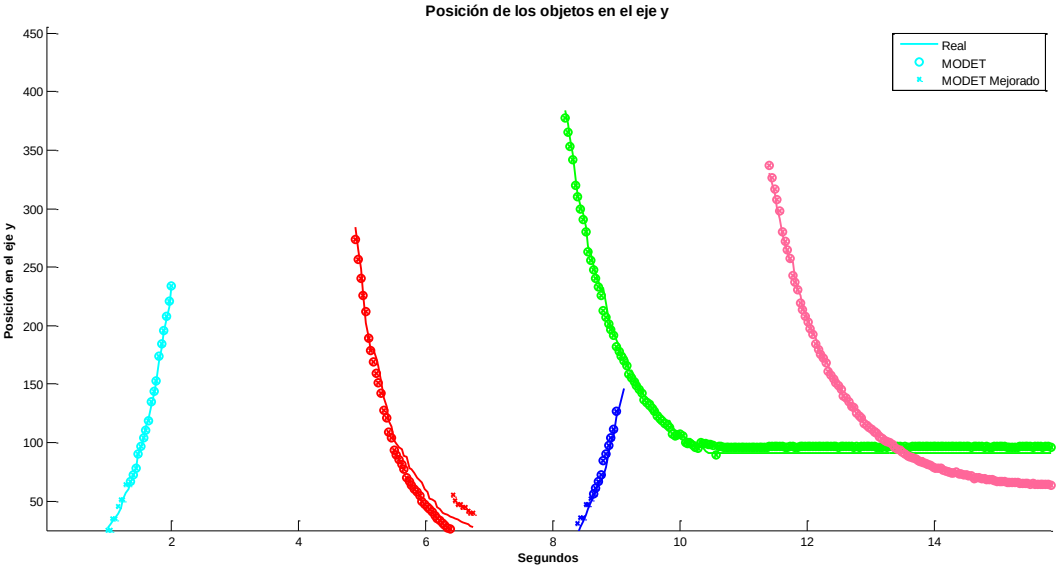

Figura 5.19. Posición real de los objeto frente a posición MODET y MODET Mejorado en el eje y desde la cámara 2.

En las figuras se comprueba cómo el algoritmo también obtiene muy buenos resultados de seguimiento en secuencias con múltiples objeto, ya que se aprecia como la trayectoria real y la trayectoria del algoritmo prácticamente se solapa. Por otro lado, en los casos para los que **MODET** no detecta el objeto, **MODET** Mejorado proyecta correctamente su trayectoria.

A continuación, en las tablas 5.12 y 5.13, se muestra una comparación entre los algoritmos **MODET** y **MODET** Mejorado en las cámaras 1 y 2, respectivamente.

Tabla 5.12. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	6/113	0/113
O1	Error medio	6.87 ± 2.42	6.87 ± 2.42
	Correlación en eje x	0.9999	0.9999
	Correlación en eje y	0.9998	0.9998
O2	Error medio	10.30 ± 7.56	12.27 ± 10.45
	Correlación en eje x	0.9991	0.9986
	Correlación en eje y	0.9986	0.9902
O3	Error medio	30.52 ± 9.21	28.96 ± 10.42
	Correlación en eje x	0.9986	0.9986
	Correlación en eje y	0.9944	0.9944
O4	Error medio	18.34 ± 14.33	18.34 ± 14.33
	Correlación en eje x	0.9946	0.9946
	Correlación en eje y	0.9917	0.9917
O5	Error medio	19.21 ± 6.42	18.21 ± 7.55
	Correlación en eje x	0.9993	0.9991
	Correlación en eje y	0.9965	0.9967
Total	Error medio	17.98 ± 11.27	17.87 ± 11.60
	Correlación en eje x	0.9965	0.9968
	Correlación en eje y	0.9968	0.9956

Tabla 5.13. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	27/394	3/394
O1	Error medio	2.19 ± 1.29	5.55 ± 5.06
	Correlación en eje x	0.9996	0. 9965
	Correlación en eje y	0.9997	0. 9990
O2	Error medio	12.09 ± 3.04	13.60 ± 4.62
	Correlación en eje x	0.9981	0. 9974
	Correlación en eje y	0.9982	0. 9889
O3	Error medio	7.87 ± 5.38	7.87 ± 5.38
	Correlación en eje x	0.9995	0.9995
	Correlación en eje y	0.9984	0.9984
O4	Error medio	6.92 ± 5.44	8.05 ± 4.49
	Correlación en eje x	0.9811	0. 9849
	Correlación en eje y	0.9861	0. 9934
O5	Error medio	11.17 ± 5.23	11.17 ± 5.23
	Correlación en eje x	0.9994	0.9994
	Correlación en eje y	0.9994	0.9994
Total	Error medio	9.01 ± 5.51	9.35 ± 5.66
	Correlación en eje x	0.9991	0. 9992
	Correlación en eje y	0.9961	0. 9960

Siguiendo la misma línea de los resultados obtenidos en el apartado anterior, para **MODET** se obtiene un error medio y desviación típica bajos, consiguiendo para este caso un porcentaje de detección del 93,49% en las dos cámaras. En cambio, con **MODET** Mejorado se consigue aumentar hasta el 99,41% el porcentaje de detección perdiendo ligeramente precisión en la cámara 2 y contrarrestando esa pérdida de precisión en la cámara 1.

Seguidamente, se muestra el conjunto de resultados de las distintas secuencias que intervienen en esta prueba. Tanto los resultados cómo las gráficas de las distintas secuencias de esta prueba, se encuentran en el anexo *A.2*.

Tabla 5.14. Resultados del algoritmo MODET.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	6/113	17.98 ± 11.27	0.9965	0.9968
	C2	27/394	9.01 ± 5.51	0.9991	0.9961
Secuencia 2	C1	6/101	33.78 ± 14.60	0.9835	0.9889
	C2	27/459	8.56 ± 7.65	0.9993	0.9959
Secuencia 3	C1	2/241	5.48 ± 3.12	0.9997	0.9995
	C2	194/366	9.44 ± 2.42	0.9991	0.9974
Secuencia 4	C1	10/332	7.00 ± 5.58	0.9995	0.9998
	C2	6/113	13.25 ± 6.01	0.9990	0.9908
Secuencia 5	C1	10/177	12.27 ± 3.22	0.9981	0.9971
	C2	21/519	11.86 ± 8.17	0.9995	0.9929
Promedio		309/2815	10.83 ± 8.67	0.9977	0.9955

Tabla 5.15. Resultados del algoritmo MODET Mejorado.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	0/113	17.87 ± 11.60	0.9968	0.9956
	C2	3/394	9.35 ± 5.66	0.9960	0.9960
Secuencia 2	C1	0/101	33.15 ± 14.50	0.9850	0.9888
	C2	1/459	8.84 ± 7.54	0.9993	0.9951
Secuencia 3	C1	2/241	5.55 ± 3.29	0.9996	0.9995
	C2	0/366	13.32 ± 4.47	0.9962	0.9905
Secuencia 4	C1	0/332	7.91 ± 7.66	0.9990	0.9990
	C2	0/113	14.49 ± 8.71	0.9959	0.9906
Secuencia 5	C1	4/177	12.28 ± 3.46	0.9982	0.9970
	C2	3/519	12.01 ± 8.16	0.9995	0.9910
Promedio		13/2815	11.57 ± 8.90	0.9974	0.9942

Observando las tablas se llega a la conclusión de que con **MODET** Mejorado se tiene una eficacia casi del 100%, tanto en la detección como en el seguimiento de los objetos. Por otro lado, pierde eficiencia si los objetos tienen una mayor altura respecto al plano del suelo (z=0).

5.5 Escenario Exterior

Una vez que se ha comprobado el buen funcionamiento de los algoritmos **MODET** y **MODET** Mejorado, en vídeos grabados por nosotros mismos bajo un escenario interior controlado, vamos a aumentar la dificultad del escenario. En esta sección se van a implementar dos nuevas pruebas, la primera de ellas corresponde a un escenario exterior formado por dos cámaras fijas (**Prueba 3**) y la siguiente a un escenario exterior formado por una cámara móvil (**Prueba 4**).

En los siguientes apartados se explica en detalle tanto el escenario cómo los resultados de cada una de las pruebas.

5.6 Prueba 3: Escenario exterior, cámara fija y múltiples objetos

En esta sección se va a describir el escenario y los objetos que intervienen en esta prueba, así como los parámetros de **MODET** escogidos y las conclusiones en función de los resultados obtenidos.

La **Prueba 3** se ha realizado con vídeos tomados de la base de datos de PETS 2001[12]. Estos vídeos se han grabado con dos cámaras fijas, en un escenario exterior formado por una carretera situada entre un conjunto de viviendas con un aparcamiento y jardines. Los objetos que aparecen en esta prueba son varios coches, una furgoneta, y diferentes grupos de personas. Aquellas personas que no estén definidas por el número de puntos de interés suficientes no se tomarán en cuenta en esta prueba.

Los parámetros escogidos para este escenario los mostramos en la tabla 5.16. Igual que en el escenario interior, hemos fijado 7 puntos en el plano del suelo de la escena para calcular la transformación proyectiva, y así poder mapear puntos de una cámara a otra. Todos los parámetros han sido calibrados con los primeros fotogramas de la secuencia número 3.

Tabla 5.16. Parámetros del algoritmo MODET, escogidos en la fase de entrenamiento para vídeos grabados en un escenario exterior con cámara fija.

Fase	Parámetro	Valor
Extracción de puntos SURF	SURFThreshold	500
Detección	Models	1
	dist_thresh	3
	kCluster	45
	numMinPoints	15
Seguimiento	minScore	-20
	numMaxPointsTracking	2000
Fusión de datos	disObj	250

[12] http://www.cvg.rdg.ac.uk/slides/pets.html

En esta prueba vamos a utilizar 4 secuencias, cada una grabada desde dos cámaras. Estas secuencias se pueden ver en la página web del proyecto[13]. En la tabla 5.17 se han enumerado las características de las distintas secuencias de ésta prueba.

Tabla 5.17. Características de las secuencias con varios objetos en movimiento en un escenario exterior.

Secuencia	Fuente	Duración (s)	Velocidad en cámara 1 (fps)	Velocidad en cámara 2 (fps)	Tamaño de imagen (píxeles)	# de objetos
1	PETS 2001	14	25	25	768x576	2
2	PETS 2001	26	25	25	768x576	3
3	PETS 2001	5.5	25	25	768x576	2
4	PETS 2001	6	25	25	768x576	1

Los objetos que intervienen en la secuencia 1 de la Prueba 3 consisten en una furgoneta blanca y un grupo de tres personas. En este caso, se representa a la furgoneta con el rectángulo turquesa y al grupo de personas con el rectángulo rojo. A continuación, se muestran los distintos objetos vistos desde la cámara 1 y 2 en figuras 5.20 y 5.21, respectivamente.

Figura 5.20. Objetos que intervienen en la secuencia 1 vistos desde la cámara 1.

Figura 5.21. Objetos que intervienen en la secuencia 1 vistos desde la cámara 2.

[13] http://modet.clegua.es/prueba3.html

A continuación, en la figura 5.22 se representan las trayectorias sobre la imagen del escenario para la secuencia 1.

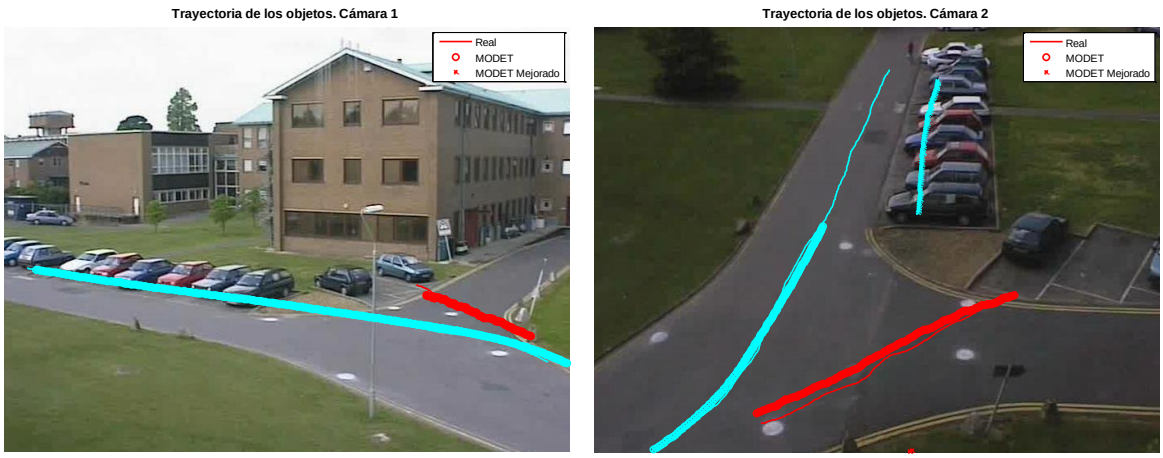

Figura 5.22. Trayectorias de los objetos a lo largo de la secuencia 1.

En ambas figuras vemos cómo **MODET** detecta el objeto prácticamente sobre la trayectoria real del objeto, reiterando los buenos resultados de este algoritmo en escenarios exteriores. En la segunda figura, se observa claramente cómo **MODET** tarda más en detectar el objeto y se representa la proyección de la otra cámara con un error mayor que en las pruebas anteriores, debido a la altura de la furgoneta con respecto al plano del suelo (ver figura 5.20 izquierda).

Las trayectorias reales, **MODET** y **MODET** Mejorado de estos objetos se representan en las siguientes figuras (5.23 y 5.24).

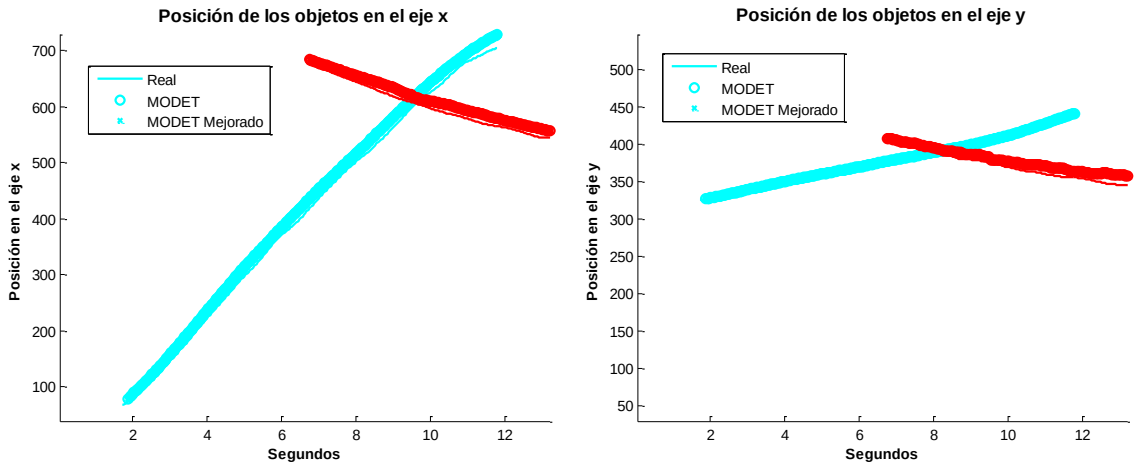

Figura 5.23. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x y en el eje y para la cámara 1.

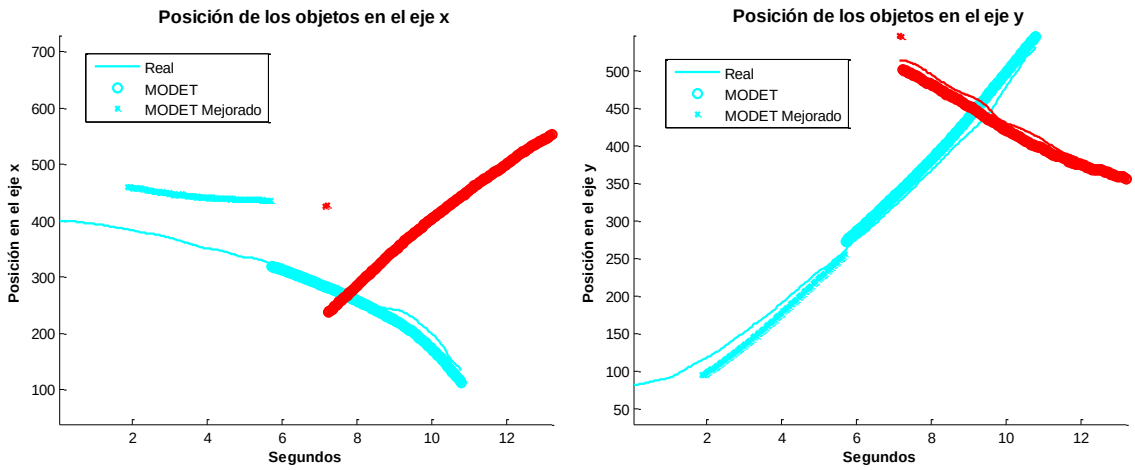

Figura 5.24. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x y en el eje y para la cámara 2.

En la cámara 1 de esta secuencia existe una oclusión parcial entre los segundos 8 y 10, puesto que la furgoneta (rectángulo turquesa) se superpone en gran parte al grupo de personas (rectángulo rojo). En la figura 5.23 se puede ver que, a pesar de esta oclusión parcial, **MODET** sigue correctamente la trayectoria real del objeto que es ocultado. En la cámara 2, **MODET** detecta los objetos en segundos posteriores a que el objeto haya aparecido en la escena. Con **MODET** Mejorado podemos proyectar la posición debido a que en la cámara 1 los objetos se detectan en *frames* anteriores. En la figura 5.24, podemos observar que esta proyección arrastra un error más elevado que en las secciones anteriores, debido a que la perspectiva con la que la cámara 1 graba este escenario, hace que los objetos tengan una altura respecto al plano del suelo muy grande.

En las tablas 5.18 y 5.19 se comparan los resultados de **MODET** y **MODET** Mejorado para la secuencia 1.

Tabla 5.18. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	6/417	6/417
O1	Error medio	15.75 ± 2.95	15.75 ± 2.95
	Correlación en eje x	0.9999	0.9999
	Correlación en eje y	0.9988	0.9988
O2	Error medio	15.00 ± 3.60	15.00 ± 3.60
	Correlación en eje x	0.9988	0.9988
	Correlación en eje y	0.9967	0.9967
Total	Error medio	15.45 ± 3.24	15.45 ± 3.24
	Correlación en eje x	0.9998	0.9998
	Correlación en eje y	0.9918	0.9918

Tabla 5.19. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	144/422	46/422
O1	Error medio	20.51 ± 9.03	50.38 ± 35.89
	Correlación en eje x	0.9941	0.9684
	Correlación en eje y	0.9990	0.9989
O2	Error medio	11.07 ± 4.37	13.36 ± 20.39
	Correlación en eje x	0.9998	0.9729
	Correlación en eje y	0.9980	0.9907
Total	Error medio	15.41 ± 8.36	35.42 ± 35.56
	Correlación en eje x	0.9985	0.9201
	Correlación en eje y	0.9791	0.9940

En estas tablas se refleja lo que anteriormente se ha explicado. El algoritmo **MODET** produce un error medio bajo, ajustándose a la trayectoria real del objeto, pero en la cámara 2 se tarda más tiempo en detectar los objetos. En **MODET** Mejorado se corrige este problema, aumentando el error medio debido a la altura de los objetos con respecto al plano del suelo. Este hecho se evidencia en los resultados del objeto 1 para la cámara 2.

En las tablas 5.20 y 5.21, mostramos el resumen de todos los resultados de las secuencias que aparecen en el anexo *A.3*.

Tabla 5.20. Resultados del algoritmo MODET.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	6/417	15.45 ± 3.24	0.9998	0.9918
	C2	144/422	15.41 ± 8.36	0.9985	0.9791
Secuencia 2	C1	155/951	15.35 ± 7.79	0.9927	0.9850
	C2	340/1056	12.15 ± 5.23	0.9968	0.9986
Secuencia 3	C1	2/218	11.31 ± 3.24	0.9971	0.9673
	C2	5/280	17.31 ± 8.35	0.9600	0.9981
Secuencia 4	C1	6/83	23.64 ± 9.34	0.9995	0.9992
	C2	78/126	12.17 ± 5.50	0.9989	0.9993
Promedio		736/3553	14.61 ± 6.97	0.9951	0.9974

Tabla 5.21. Resultados del algoritmo MODET Mejorado.

		# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	C1	6/417	15.45 ± 3.24	0.9998	0.9918
	C2	46/422	35.42 ± 35.56	0.9201	0.9940
Secuencia 2	C1	65/951	17.66 ± 10.34	0.9900	0.9836
	C2	130/1056	30.62 ± 38.26	0.9203	0.9953
Secuencia 3	C1	2/218	11.31 ± 3.24	0.9971	0.9673
	C2	5/280	17.31 ± 8.35	0.9600	0.9981
Secuencia 4	C1	6/83	23.64 ± 9.34	0.9995	0.9992
	C2	56/126	23.01 ± 17.07	0.9942	0.9994
Promedio		316/3553	22.95 ± 25.92	0.9712	0.9964

Podemos concluir que para escenarios más complejos, **MODET** obtiene muy buenos resultados incluso con oclusiones parciales, aumentando el error en la posición del objeto proyectado por **MODET** Mejorado según aumenta la altura de los objetos sobre el plano del suelo. Dicho aumento de la altura puede deberse a que los objetos posean unas dimensiones mayores o a la posición de las cámaras.

5.7 Prueba 4: Escenario exterior, cámara móvil y un objeto

En esta prueba vamos a evaluar el algoritmo en secuencias de vídeo de alta resolución con una cámara móvil. En este caso, el escenario utilizado corresponde a un circuito de carreras grabado por un aficionado en el que aparece un único coche. El vídeo ha sido tomado de la plataforma de vídeos YouTube[14].

Los parámetros fijados para este escenario son los representados en la tabla 5.22. Se han ajustado con los primeros fotogramas de la secuencia número 3.

Tabla 5.22. Parámetros del algoritmo MODET, escogidos en la fase de entrenamiento para vídeos grabados en un escenario exterior con cámara móvil.

Fase	Parámetro	Valor
Extracción de puntos SURF	`SURFThreshold`	2000
Detección	`Models`	[1 2 3]
	`dist_thresh`	3
	`kCluster`	30
	`numMinPoints`	25
Seguimiento	`minScore`	-2
	`numMaxPointsTracking`	1000

[14] http://www.youtube.com/watch?v=aWPKUUFYl4E

Para esta prueba hemos utilizado 3 secuencias que pueden visualizarse en la página web del proyecto[15]. Mostramos las características de estas secuencias en la tabla 5.23.

Tabla 5.23. Características de las secuencias con un objeto en movimiento sobre un escenario exterior grabado con una cámara móvil.

Secuencia	Fuente	Duración (s)	Velocidad de grabación (fps)	Tamaño de imagen (píxeles)	# de objetos
1	YouTube	18.5	25	1920x1080	1
2	YouTube	4.5	25	1920x1080	1
3	YouTube	2.5	25	1920x1080	1

Para representar los resultados de esta prueba, análogamente a como lo hemos hecho en las pruebas anteriores, vamos a describir los datos obtenidos para la secuencia 1. En esta secuencia aparece un único coche, el aficionado con la cámara sigue el movimiento del coche a través de su recorrido sobre el escenario. Cuando el coche se encuentra a la altura de la cámara, éste sobrepasa los límites del área de detección lo que provoca que el algoritmo detenga el seguimiento del objeto. Cuando el área del objeto vuelve a ser menor que el área de detección, **MODET** lo capta de nuevo y sigue su movimiento correctamente. En la figura 5.25 mostramos la primera y la segunda captura del objeto, con un rectángulo rosa y turquesa, respectivamente.

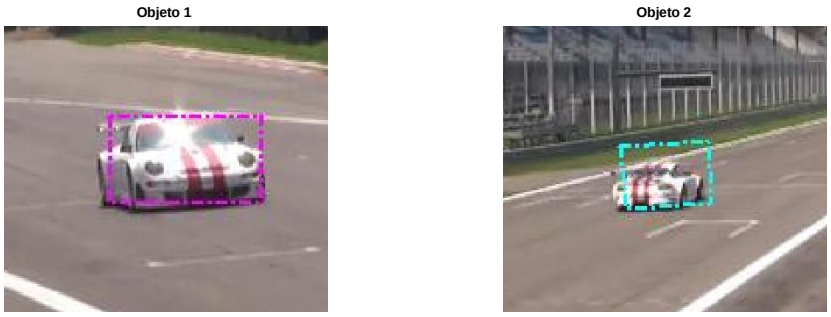

Figura 5.25. Objetos que intervienen en la secuencia 1.

En la figura 5.26 se representa la secuencia de la Prueba 4. La flecha azul indica la dirección del movimiento del objeto, que además coincide con la dirección en la que la cámara se desplaza.

[15] http://modet.clegua.es/prueba4.html

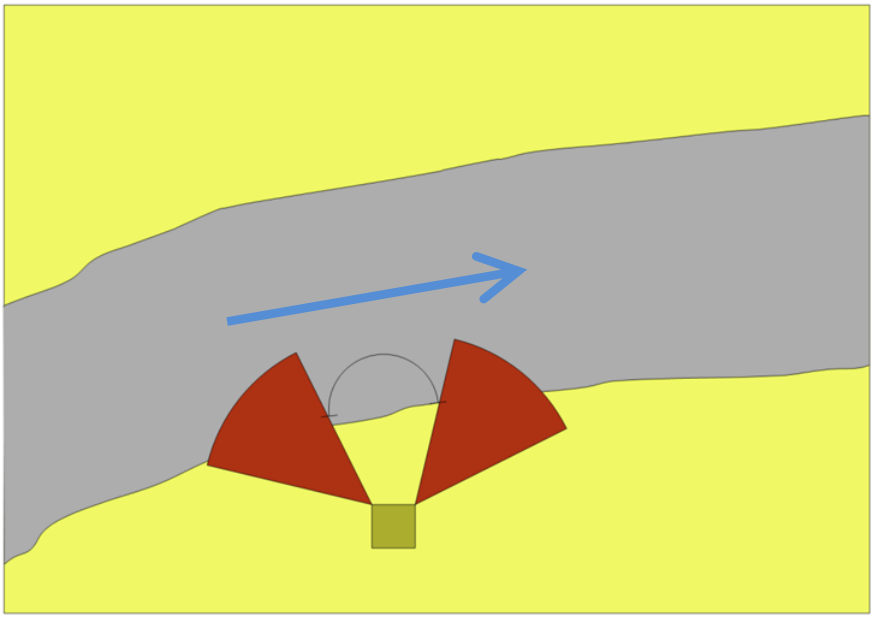

Figura 5.26. Recreación de la trayectoria del objeto en la secuencia 1 sobre el escenario exterior grabado con una cámara móvil.

En la figura 5.27 se muestran las trayectorias real y **MODET** del objeto en la primera captación y en la segunda captación.

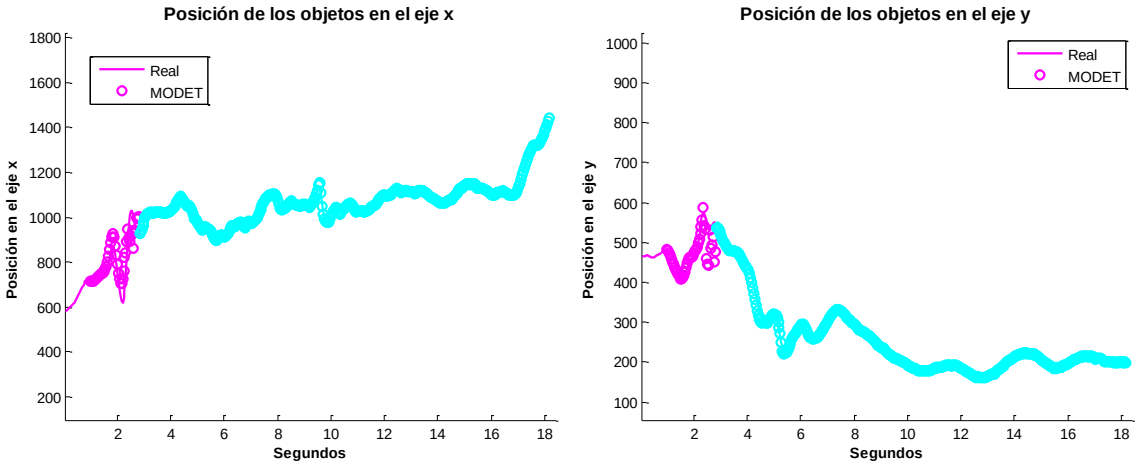

Figura 5.27. Posición real de los objetos frente a posición MODET en el eje x y en el eje y.

En la anterior figura observamos cómo el algoritmo detecta y sigue al objeto correctamente a pesar del movimiento de la cámara. Además, vemos cómo en el instante en el que el área del objeto es mayor que el área de detección, **MODET** detiene el seguimiento. En la gráfica observamos que este instante es mínimo, sin que ello afecte a los buenos resultados que se están obteniendo de **MODET**.

La tabla 5.24 muestra los resultados de la secuencia 1.

Tabla 5.24. Resultados del algoritmo MODET.

		MODET
	Número de frames en los que no se detecta el objeto	23/453
O1	Error medio	56.67 ± 44.81
	Correlación en eje x	0.8351
	Correlación en eje y	0.8133
O2	Error medio	4.98 ± 7.26
	Correlación en eje x	0.9975
	Correlación en eje y	0.9989
Total	Error medio	10.51 ± 22.66
	Correlación en eje x	0.9856
	Correlación en eje y	0.9956

En la tabla podemos comprobar que, para esta secuencia con la cámara en movimiento, se mantienen las prestaciones de **MODET** vistas en pruebas anteriores. Vemos que, el objeto ha sido detectado en un 94,9% de los *frames* en los que aparece, con un bajo error medio. Cabe destacar que el error medio, medido en píxeles, será mayor cuanto mayor sea el tamaño de la imagen. Para las secuencias de esta prueba, aunque se mantenga la proporción del error medio, este será mayor debido a que la imagen generada tiene un tamaño en el eje x de 1920 píxeles y en el eje y de 1080 píxeles.

Por último mostramos la totalidad de resultados de las secuencias recogidas en el anexo *A.4* en la tabla 5.25.

Tabla 5.25. Resultados del algoritmo MODET.

	# de frames no detecta objeto	Error Medio	Correlación en eje x	Correlación en eje y
Secuencia 1	23/453	10.51 ± 22.66	0.9856	0.9956
Secuencia 2	0/100	28.12 ± 39.98	0.9266	0.9408
Secuencia 3	5/59	3.37 ± 3.64	0.9996	0.9967
Promedio	28/612	12.87 ± 26.51	0.9910	0.9941

Con esta prueba podemos concluir que el algoritmo **MODET** también tiene un buen funcionamiento en vídeos en los que la cámara sigue al objeto en movimiento. Como se ha explicado antes, el instante en el que el objeto no se detecta, no afecta a los resultados de la trayectoria, ya que la trayectoria **MODET** y la real están prácticamente en la misma posición.

5.8 Prueba 5: Otros escenarios

Por último, sin efectuar un análisis exhaustivo, hemos comprobado el funcionamiento de **MODET** en escenarios de distinta naturaleza, con la finalidad de corroborar los resultados obtenidos con los anteriores escenarios. Vamos a mostrar un ejemplo que corresponde a una secuencia de un partido de tenis grabado con una cámara fija. En los resultados se observa que **MODET** detecta y sigue correctamente a ambos jugadores que intervienen en el partido, afianzando los buenos resultados de **MODET** en escenarios complejos. El tamaño de imagen de esta secuencia es de 1280x720 píxeles, con una velocidad de grabación de 25 fps.

Para esta secuencia hemos fijado los parámetros que mostramos en la tabla 5.26.

Tabla 5.26. Parámetros del algoritmo MODET, escogidos en la fase de entrenamiento para vídeos grabados en un escenario complejo con cámara fija.

Fase	Parámetro	Valor mínimo
Extracción de puntos SURF	SURFThreshold	500
Detección	Models	1
	dist_thresh	3
	kCluster	40
	numMinPoints	15
Seguimiento	minScore	-20
	numMaxPointsTracking	2000

En la figura 5.28 mostramos a los dos jugadores que intervienen en la secuencia y son detectados por **MODET**.

Figura 5.28. Objetos que intervienen en la secuencia.

A continuación, en la figura 5.29 se representan las trayectorias sobre la imagen del escenario para la secuencia.

Figura 5.29. Trayectorias de los objetos a lo largo de la secuencia.

En la figura 5.30 se muestran las trayectorias real y **MODET** de los objetos.

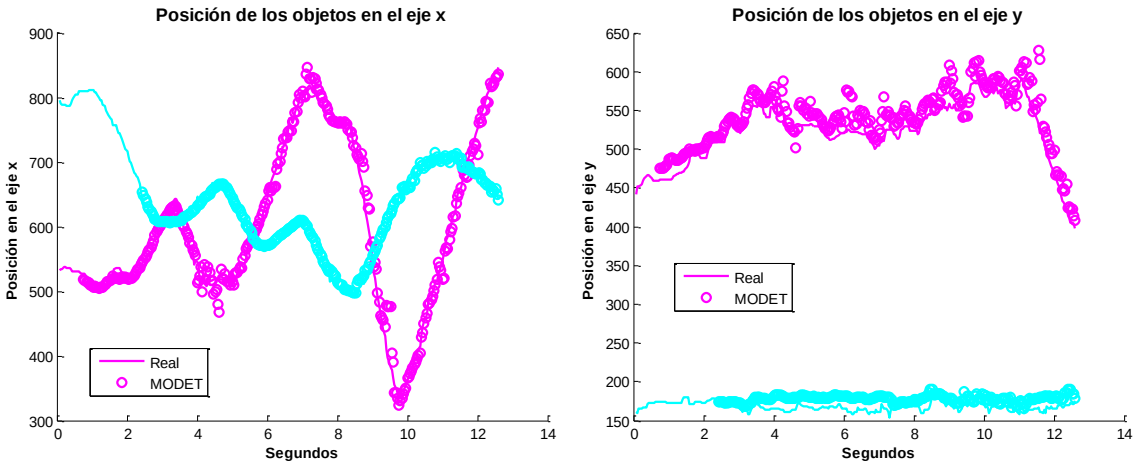

Figura 5.30. Posición real de los objetos frente a posición MODET en el eje x y en el eje y.

En la anterior figura observamos cómo el algoritmo detecta y sigue a los objetos correctamente. Tarda más en detectar al objeto turquesa debido a que está situado en el fondo de la imagen, lo que conlleva un menor número de puntos de interés.

La tabla 5.27 muestra los resultados de la secuencia.

Tabla 5.27. Resultados del algoritmo MODET.

		MODET
	Número de frames en los que no se detecta el objeto	76/628
O1	**Error medio**	21.98 ± 14.40
	Correlación en eje x	0.9921
	Correlación en eje y	0.9395
O2	**Error medio**	13.02 ± 4.99
	Correlación en eje x	0.9963
	Correlación en eje y	0.1566
Total	**Error medio**	17.84 ± 11.96
	Correlación en eje x	0.9927
	Correlación en eje y	0.9984

En la tabla se puede comprobar que para un escenario complejo como el presentado, en el que los jugadores de tenis cambian rápidamente de dirección de movimiento, se mantienen las prestaciones de **MODET**.

5.9 Tiempo de ejecución

Aunque no es el objetivo del trabajo, se han realizado algunas pruebas de tiempo de ejecución para valorar su posible implementación en un sistema real.

En las tablas 5.28, 5.29, 5.30 y 5.31 vamos a mostrar las velocidades de ejecución en *frames* por segundo (fps) para el algoritmo **MODET** y **MODET** Mejorado en las pruebas 1, 2, 3 y 4. Los cálculos se han realizado desde un ordenador personal con un procesador de 4 núcleos a 2.00 GHz y una memoria RAM de 4 GB.

Tabla 5.28. Velocidad de procesamiento de los algoritmos MODET y MODET Mejorado para la Prueba 1.

		MODET (fps)	MODET Mejorado (fps)
Secuencia 1	C1	11.6599	13.8352
	C2	15.3345	
Secuencia 2	C1	13.6530	12.4997
	C2	11.9967	
Secuencia 3	C1	11.6271	12.2295
	C2	12.5914	
Secuencia 4	C1	11.9981	13.2823
	C2	14.0417	
Secuencia 5	C1	13.2499	14.4750
	C2	15.1913	
Secuencia 6	C1	13.1125	13.7927
	C2	14.2145	
Secuencia 7	C1	14.5522	14.9817
	C2	15.3644	
Secuencia 8	C1	11.7402	12.8415
	C2	13.9878	
Promedio		13.5663	13.5659

Tabla 5.29. Velocidad de procesamiento de los algoritmos MODET y MODET Mejorado para la Prueba 2.

		MODET (fps)	MODET Mejorado (fps)
Secuencia 1	C1	12.7910	12.0094
	C2	11.6820	
Secuencia 2	C1	13.3424	11.4674
	C2	10.7896	
Secuencia 3	C1	9.6157	11.7817
	C2	13.5092	
Secuencia 4	C1	8.6478	9.8605
	C2	10.7989	
Secuencia 5	C1	11.6212	9.2216
	C2	8.3346	
Promedio		11.1133	10.8960

Tabla 5.30. Velocidad de procesamiento de los algoritmos MODET y MODET Mejorado para la Prueba 3.

		MODET (fps)	MODET Mejorado (fps)
Secuencia 1	C1	5.2906	6.5311
	C2	8.5881	
Secuencia 2	C1	4.8999	5.6174
	C2	6.6700	
Secuencia 3	C1	5.5792	6.5743
	C2	8.0313	
Secuencia 4	C1	6.5561	7.9095
	C2	9.9977	
Promedio		6.9164	6.6294

Tabla 5.31. Velocidad de procesamiento del algoritmo MODET para la Prueba 4.

	MODET (fps)
Secuencia 1	3.1939
Secuencia 2	3.3371
Secuencia 3	2.9184
Promedio	3.2664

En las tablas anteriores podemos observar cómo varía la velocidad de procesamiento en función de los tamaños de imagen de cada vídeo. Las pruebas 1 y 2, realizadas sobre vídeos con tamaño de imagen de 640x480 píxeles, tienen una velocidad de procesamiento de 13.56 fps y 11.11 fps para **MODET** y 13.5659 fps y 10.8960 fps para **MODET** Mejorado. Estas velocidades podrían ser aceptables para un sistema en tiempo real, teniendo en cuenta que, pierde rendimiento cuanto mayor sea el número de objetos seguidos en la escena. La Prueba 3, realizada sobre vídeos con tamaño de imagen de 768x576 píxeles, tiene una velocidad de procesamiento de 6.91 fps para **MODET** y 6.62 fps para **MODET** Mejorado. La velocidad se ve muy afectada por el aumento del tamaño de la imagen de los vídeos. Por último, la Prueba 4, realizada sobre vídeos con tamaño de imagen de 1920x1080 píxeles, tiene una velocidad de procesamiento de 3.26 fps para **MODET**. Utilizando vídeos de HD observamos que, a pesar de disminuir la velocidad, sería posible llegar a procesar 3 *frames* por segundo.

Sin llegar a profundizar en la posible implementación de **MODET** para un sistema de tiempo real, se puede valorar su desarrollo en sistemas con tamaño de imagen de vídeo estándar (640x480 píxeles). Además, en el capítulo de trabajo futuro propondremos algunas mejoras de eficiencia.

6 CONCLUSIONES Y TRABAJO FUTURO

A lo largo de este proyecto se han estudiado diferentes algoritmos con la finalidad de detectar y seguir objetos en movimiento. Además, se ha analizado cómo mejorar el seguimiento de objetos utilizando información de múltiples cámaras. Para ello se realizó una investigación previa de los distintos sistemas de *tracking* que existen en la literatura. En el capítulo del estado del arte, se analizó en mayor profundidad los métodos que basan su estrategia en la correspondencia de puntos característicos. Como consecuencia de este estudio previo, se decidió qué técnicas utilizar en los diferentes módulos de la arquitectura: detección de objetos en movimiento, seguimiento de los objetos detectados y alineación de información entre las cámaras. Para la identificación de regiones en movimiento de la escena se ha utilizado un método de flujo óptico, que agrupa correspondencias de puntos en movimiento. Los grupos de puntos en movimiento localizados, se representan por los puntos de interés que contienen estos conjuntos, formando objetos. Para el seguimiento de objetos, se ha utilizado un método estadístico de tracking por puntos, que busca correspondencia entre puntos de cada objeto en distintos instantes de tiempo, dando lugar a la transformación del objeto. En la última fase se ha fusionado la información de las cámaras utilizando una transformación 2D para alinear los datos.

Tanto en la fase de detección como en la de seguimiento de objetos, se ha decidido utilizar una estrategia basada en puntos de interés para buscar correspondencias entre puntos en diferentes instantes de tiempo. El algoritmo elegido para detectar y describir los puntos característicos ha sido SURF. La elección de SURF frente a otros algoritmos como SIFT ha sido motivada por su gran balance entre precisión y eficiencia, siendo un algoritmo muy respaldado en diversos artículos de la literatura. Para minimizar errores entre correspondencia de puntos, se ha escogido un algoritmo de alineamiento que permite seleccionar los puntos que siguen una transformación predominante. La técnica de alineamiento empleada para la eliminación de correspondencias espurias, ha sido RAMOSAC, que es una evolución de RANSAC. En la fase de detección, gracias a esta cadena de algoritmos, logramos segmentar los puntos estáticos en la escena y los puntos en movimiento, incluso en vídeos grabados con cámara móvil. En la fase de seguimiento, se consigue modelar transformaciones de objetos ya detectados aunque éstos estén definidos por muy pocos puntos. En la última fase, con la finalidad de mejorar el algoritmo de detección y seguimiento de objetos, se ha utilizado homografía proyectiva fusionando la información de las diferentes cámaras. La mejora implementada consiste en la utilización de los datos recogidos en cada cámara para emparejar objetos observados desde distintas vistas, y así recuperar objetos en todas las cámaras que sólo han sido detectados en algunas.

Para evaluar el sistema desarrollado se han utilizado distintas bases de datos, desde vídeos grabados por nosotros hasta vídeos utilizados en la valoración de otros sistemas de tracking. Las diferentes pruebas planteadas han sido las siguientes:

- Ocho secuencias en un escenario interior, grabado con dos cámaras fijas en el que interviene un objeto en movimiento.

- Cinco secuencias en un escenario interior, grabado con dos cámaras fijas en el que intervienen varios objetos en movimiento.
- Cuatro secuencias en un escenario exterior, de la base de vídeos de PETS 2001, grabado con dos cámaras fijas donde existen distintos objetivos como vehículos o personas en movimiento.
- Tres secuencias en un escenario exterior, grabado por una cámara móvil en el que interviene un objeto en movimiento.

A lo largo de estas pruebas se ha comparado **MODET** frente a otro algoritmo de tracking como es Mean Shift, obteniendo mejores resultados que éste para algunas de las secuencias evaluadas. También, se han obtenido buenos resultados en secuencias de mayor dificultad en las que intervienen varios objetos y existen oclusiones parciales. Por último, a pesar de no ser el objetivo de este proyecto, se ha demostrado la eficacia de **MODET** en secuencias de alta resolución grabadas con una cámara móvil. En todos los casos, se comprueba cómo la trayectoria real del objeto y la de **MODET** es prácticamente la misma, siendo el error medio mínimo. **MODET** Mejorado establece un incremento en el porcentaje de detección de objetos en movimiento con respecto a **MODET**, con una leve pérdida de eficiencia causada por el error de proyección entre cámaras. Finalmente, sin realizar un estudio exhaustivo, se ha demostrado la viabilidad del desarrollo tanto de **MODET** como de **MODET** Mejorado en aplicaciones en tiempo real para tamaños de imagen de 640x480 píxeles.

El objetivo de este PFC, detectar y seguir objetos en movimiento de manera automática en sistemas con múltiples cámaras, queda validado consiguiendo un algoritmo robusto en diferentes escenarios, pudiendo detectar y seguir objetos en sistemas grabados por dos cámaras fijas o por una cámara móvil. El único requisito necesario para el buen funcionamiento de **MODET** es el ajuste de los parámetros para cada escenario. Además, no es necesaria información previa sobre las cámaras. Para las pruebas realizadas en este proyecto, se ha calibrado el algoritmo utilizando una de las secuencias de cada escenario, y se ha evaluado con el resto.

Como primera idea acerca de realizar un futuro estudio se propone la utilización de algoritmos de detección y descripción de puntos característicos, estudiados en el estado del arte, que tengan menor carga computacional que SURF, como ORB o FREAK, pudiendo aumentar la velocidad de **MODET** para una posible implementación en sistemas que funcionen en tiempo real.

Otra mejora posible, sería la reducción del error de proyección para **MODET** Mejorado, aprendiendo de manera automática la transformación homográfica, usando para ello *frames* en los que aparezca el objeto en las dos vistas.

Por otro lado, se plantea la representación de trayectorias 3D del plano suelo de los objetos sobre una recreación del escenario visto desde arriba, disminuyendo el error producido en la proyección, de una cámara a otra, del centro de masas del objeto en **MODET** Mejorado.

También, se propone la utilización de correspondencias automáticas entre cámaras para encontrar relaciones geométricas, sustituyendo las fijadas manualmente, haciendo extensible la utilización de **MODET** Mejorado en secuencias grabadas con sistemas con cámaras móviles.

Por último, se sugiere extender la implementación de **MODET** Mejorado para su utilización en sistemas con más de dos cámaras. Una aproximación puede ser establecer relaciones trifocales para escenarios grabados con tres cámaras.

REFERENCIAS

[1] A. Elgammal, R. Duraiswami, D. Harwood y L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. PIEEE, 90, p 1151-1163, 2002.

[2] R. Mathew, Z. Yu y J. Zhang. Detecting new stable objects in surveillance video. MSP, p 1-4, 2005.

[3] H. Liao, J.Chang y L. Chen. A localized Approach to abandoned luggage detection with Foreground-Mask sampling. AVSS, p 132-139, 2008.

[4] I. Haritaoglu, D. Harwood y L.S. Davis. W4: A real time system for detecting and tracking people. CVPR, p 962-967, 1998.

[5] C. Stauffer y W. Grimson. Adaptive background mixture models for realtime tracking. CVPR, p 246-252, 1999.

[6] A. Lipton, H. Fujiyoshi y R. Patil. Moving target classification and tracking from real-time video. WACV, p 8-14, 1998.

[7] H. Deng, B. Xiong, O. Qiaofeng. Moving Target Detection Based on Discontinuous Frame Difference Regional Optical Flow Method. OE, p 300-307, 2009.

[8] A. Wedel, A. Meißner, C. Rabe, U. Franke y D. Cremers. Detection and Segmentation of Moving Objects from Dense Scene Flow Independently. EMMCVPR, p 14-27, 2009.

[9] T. Horprasert, D. Harwood y L.S. Davis. A statistical approach for realtime robust background subtraction and shadow detection. FRW, p 1-19, 1999.

[10] A. Yilmaz, O. Javed y M. Shah. Object Tracking: A Survey. ACM CS, 38, 4, 13, 2006.

[11] C. Veenman, M. Reinders y E. Backer. Motion tracking as a constrained optimization problem. PR, 36, 9, p 2049-2067, 2003.

[12] D. Serby, E. Koller-Meier y L. Gool. Probabilistic object tracking using multiple features. ICPR, 2004.

[13] D. Comaniciu, D. Ramesh y V. Andmeer. Kernel-based object tracking. PAMI, 25, p 564-575, 2003.

[14] A. Yilmaz, X. Li y M. Shah. Contour based object tracking with occlusion handling in video acquired using mobile cameras. PAMI, 26, 11, p 1531-1536, 2004.

[15] A. Sundaresan y R. Chellappa. Multi-camera Tracking of Articulated Human Motion Using Motion and Shape Cues. ITIP, 18, 9, p 2114-2126, 2009.

[16] A. Ali y J. Aggarwal. Segmentation and recognition of continuous human activity. DRWV, p 28-35, 2001.

[17] S. Zhu y A. Yuille. Region competition: unifying snakes, region growing, and Bayes/mdl for multiband image segmentation. PAMI, 18, 9, p 884-900, 1996.

[18] N. Paragios y R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. JCV, 46, 3, p 223-247, 2002.

[19] P. Fieguth, D. Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. CVPR, p 21-27, 1997.

[20] G. Edwards, C. Taylor y T. Cootes. Interpreting face images using active appearance models. FGR, p 300-305, 1998.

[21] T. Cootes, G. Edwards y C. Taylor. Robust real-time periodic motion detection, analysis, and applications. PAMI, 23, 6, p 681-685, 2001.

[22] B. Moghaddam y A. Pentland. Probabilistic visual learning for object representation. TPAMI, 19, 7, p 696-710, 1997.

[23] M. Black y A. Jepson. Eigentracking: robust matching and tracking of articulated objects using a view-based representation. IJCV, 26, 1, p 63-84, 1998.

[24] K. Mikolajczyk y C. Schmid. Scale and Affine Invariant Interest Point Detectors. IJCV, 1, 60, p 63-86, 2004.

[25] V. Salari y I. Sethi. Feature point correspondence in the presence of occlusion. TPAMI, 12, 1, p 87-91, 1990.
[26] C. Veenman, M. Reinders y E. Backer. Resolving motion correspondence for densely moving points. PAMI, 23, 1, p 54-72, 2001.
[27] K. Shafique y M. Shah. A non-iterative greedy algorithm for multi-frame point correspondence. ICCV, p 110-115, 2003.
[28] T. Broida y R. Chellappa. Estimation of object motion parameters from noisy images. ITPAMI, 8, 1, p 90-99, 1986.
[29] L. Lu, X. Dai y G. Hager. Efficient particle filtering using RANSAC with application to 3D face tracking. IVC, 24, 6, p 581-592, 2006.
[30] P. Strandmark y I. Gu. Joint Random Sample Consensus and Multiple Motion Models for Robust Video Tracking. SCIA, 2009.
[31] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. CVPR, p 232-237, 1998.
[32] D. Comaniciu y P. Meer. Mean Shift: A robust approach toward feature space analysis. TPAMI, 24, 5, p 603-619, 2002.
[33] S. Avidan. Support vector tracking. CVPR, p 184-191, 2001.
[34] D. Huttenlocher, J. Noh y W. Rucklidge. Tracking nonrigid objects in complex scenes. ICCV, p 93-101, 1993.
[35] N. PeterFreund. Robust tracking of position and velocity with Kalman snakes. TPAMI, 21, 6, p 564-569, 1999.
[36] M. Isard y A. Blake. Condensation - conditional density propagation for visual tracking. IJCV, 29, 1, p 5-28, 1998.
[37] M. Fischler y R Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. ACM, 24, p 381-395, 1981.
[38] J. Shi y C. Tomasi. Good features to track. CVPR, p 593-600, 2004.
[39] C. Harris y M. Stephens. A combined corner and edge detector. AVC, p 147-151, 1988.
[40] D. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60, p. 91-110, 2004.
[41] H. Bay, T. Tuytelaars y L. Gool. Speeded-Up Robust Features (SURF). CVIU, 110(3), 2008.
[42] E. Rosten y T. Drummond. Machine learning for high speed corner detection. ECCV, 1, 2006.
[43] P. Jensfelt, D. Kragic, J. Folkesson y M. Björkman. A framework for vision based bearing only 3D SLAM. ICRA, 2006.
[44] S. Smith. A new class of corner finder. MVC, p 139-148, 1992.
[45] S. Leutenegger, M. Chli y R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. ICCV, p 2548-2555, 2011.
[46] M. Calonder, V. Lepetit, C. Strecha y P. Fua. BRIEF: Binary Robust Independent Elementary Features. ECCV, 2010.
[47] E. Rublee, V. Rabaud, K. Konolige y G. Bradski. ORB: an efficient alternative to SIFT or SURF. 2011.
[48] A. Alahi, R. Ortiz y P. Vandergheynst. FREAK: Fast Retina Keypoint. CVPR, 2012.
[49] J. Kang, I. Cohen y G. Medioni. Tracking people in crowded scenes across multiple cameras. ACCV, 2004.
[50] R. Hartley y A. Zisserman. Multiple View Geometry in Computer Vision. CUP, 2000.
[51] W. Du, J. Hayet, J. Piater y J. Verly. Collaborative multi-camera tracking of athletes in team sports. CVBASE, 2006.
[52] M. Taj y A. Cavallaro. Multi-camera track-before-detect. ICDSC, 2009.
[53] N. Anjum y A. Cavallaro. Trajectory association and fusion across partially overlapping cameras. AVSS, 2009.
[54] V. Kettnaker y R. Zabih. Bayesian multi-camera surveillance. CVPR, 1999.
[55] T. Huang y S. Russell. Object identification in a Bayesian context. IJCAI, 1997.

[56] A. Dick y M. Brooks. A stochastic approach to tracking objects across multiple cameras. BSLNCS, 3339, 2005, p 160-170, 2004.
[57] Y. Sheikh y M. Shah. Trajectory association across multiple airborne cameras. PAMI, 30, 2, p 361-367, 2008.
[58] X. Wang, K. Tieu y W. Grimson. Correspondence-Free multi-camera activity analysis and scene modeling. CVPR, 2008.

ANEXO DE PRUEBAS Y RESULTADOS

A.1 Escenario interior, cámara fija y un objeto

A continuación recogemos los resultados de las pruebas realizadas en vídeos con sólo un objeto en movimiento grabados en un escenario interior.

Secuencia 2

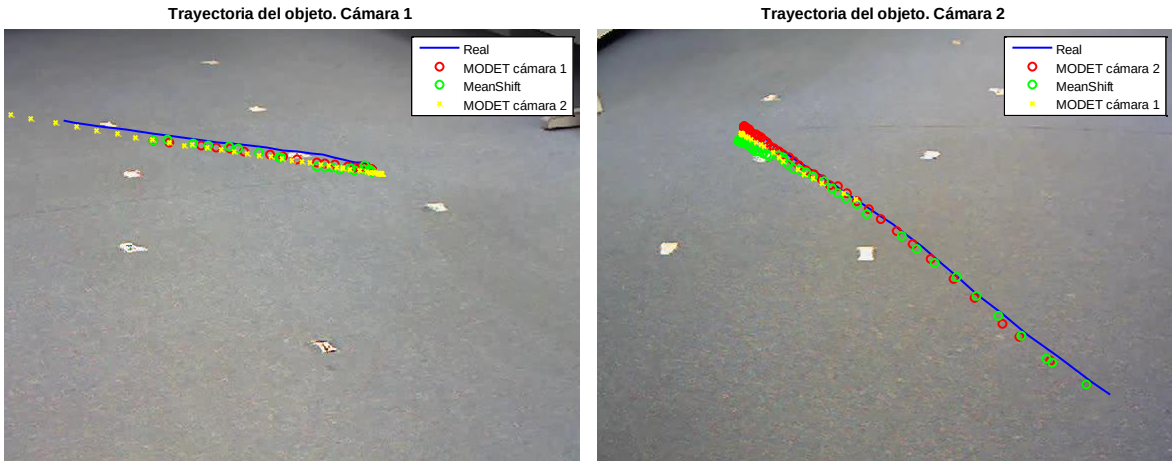

Figura A.1. Trayectorias del objeto a lo largo de la secuencia 2.

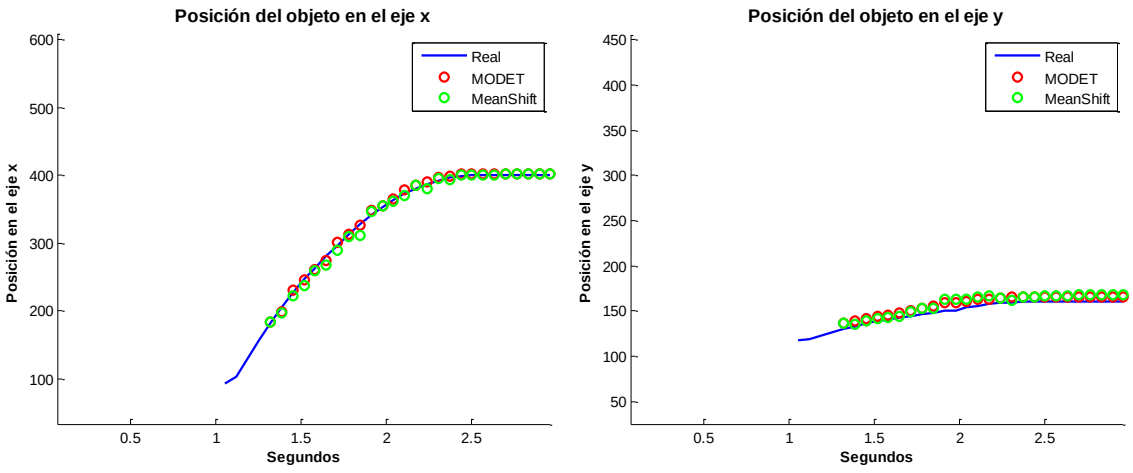

Figura A.2. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

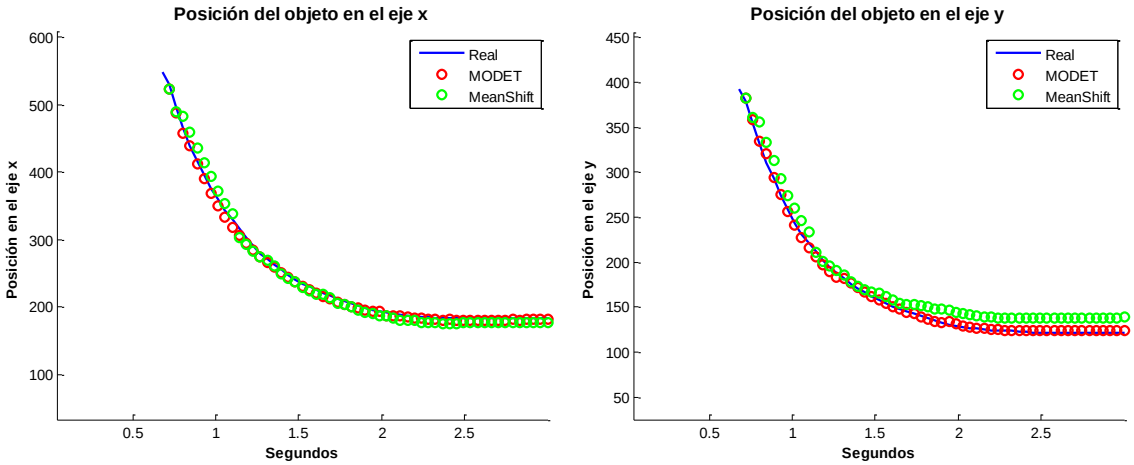

Figura A.3. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

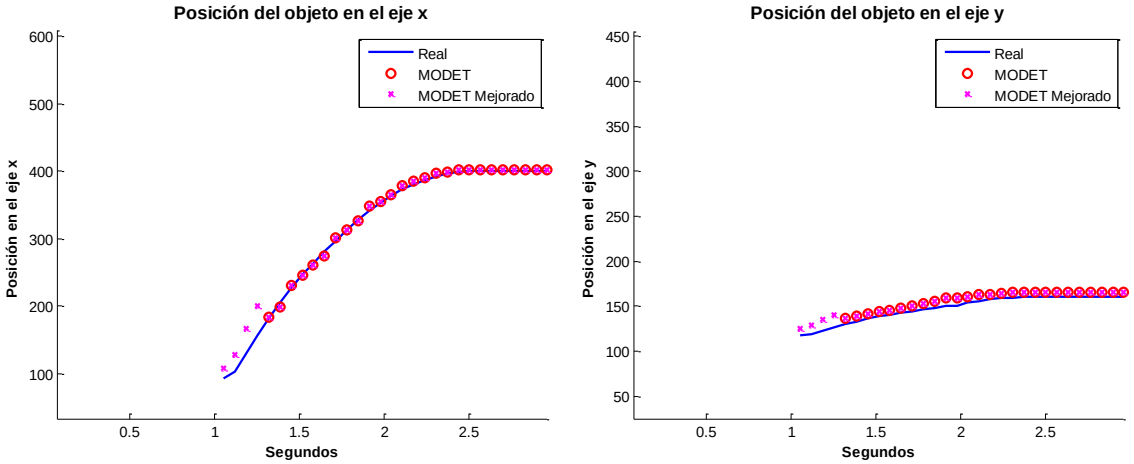

Figura A.4. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

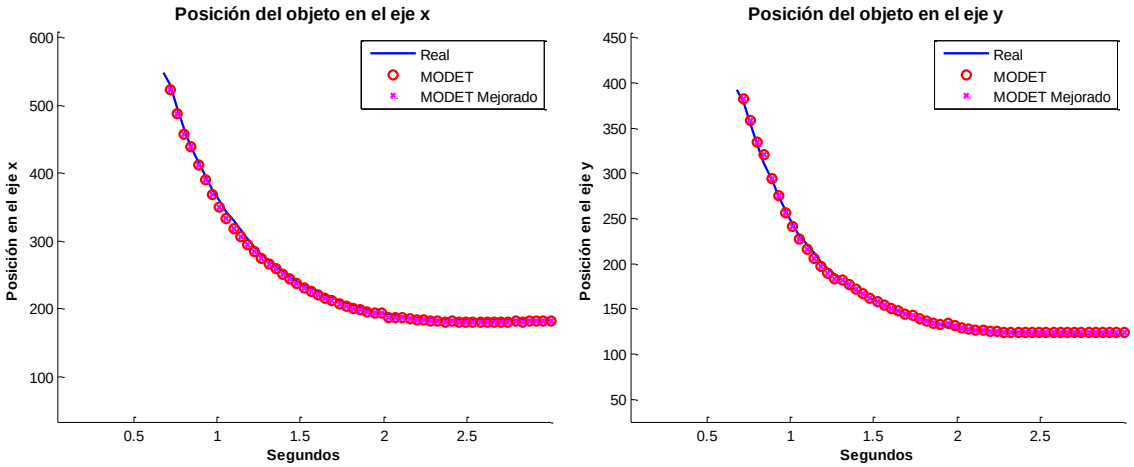

Figura A.5. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.1. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	4/30	4/30	0/30
Error medio	6.96 ± 1.25	8.49 ± 3.07	10.34 ± 9.78
Correlación en eje x	0.9993	0.9979	0.9953
Correlación en eje y	0.9952	0.9727	0.9913

Tabla A.2. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	1/56	1/56	1/56
Error medio	5.14 ± 2.57	15.31 ± 6.26	5.14 ± 2.57
Correlación en eje x	0.9997	0.9980	0.9997
Correlación en eje y	0.9992	0.9956	0.9992

Secuencia 3

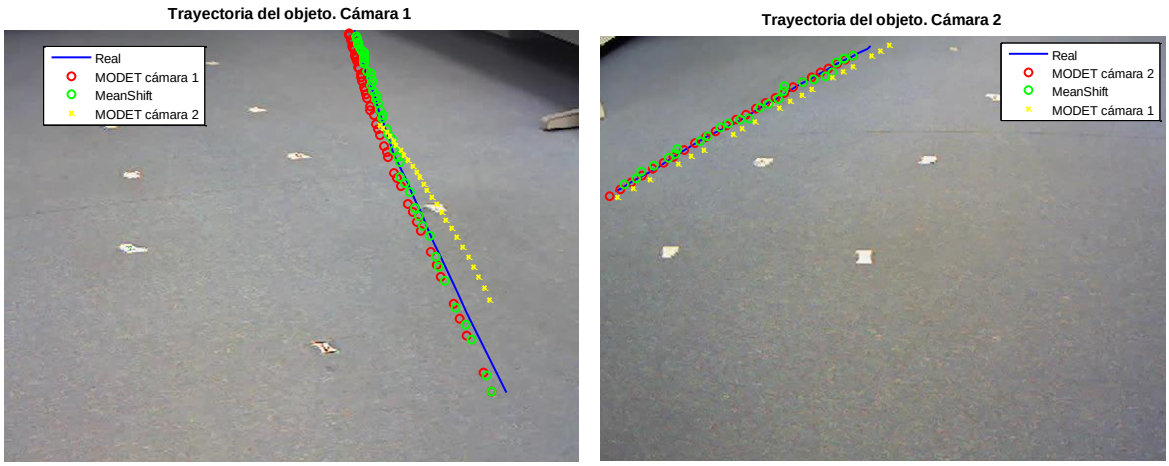

Figura A.6. Trayectorias del objeto a lo largo de la secuencia 3.

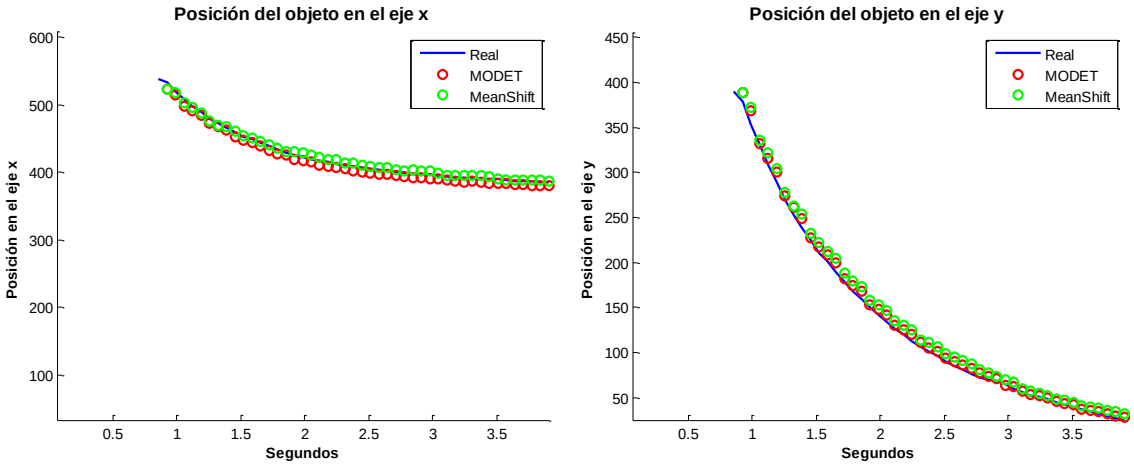

Figura A.7. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

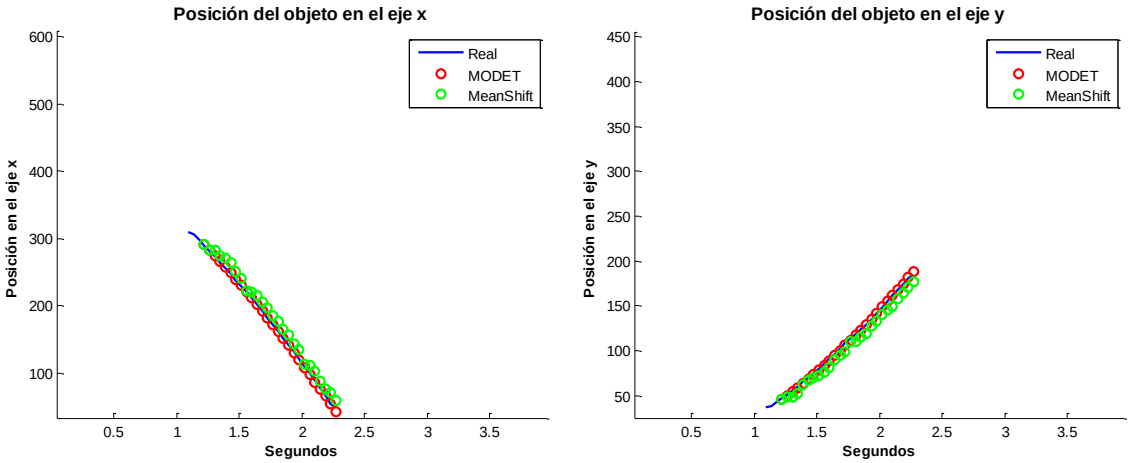

Figura A.8. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

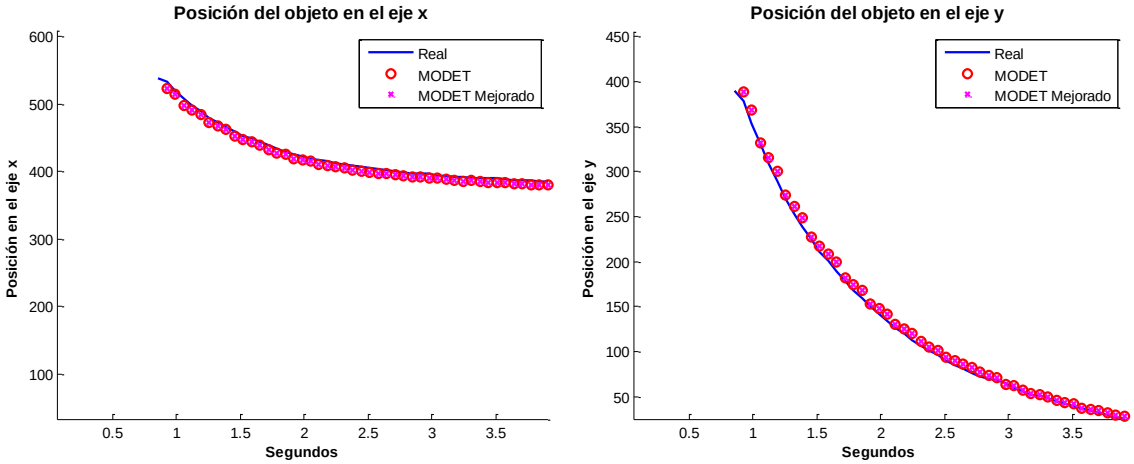

Figura A.9. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

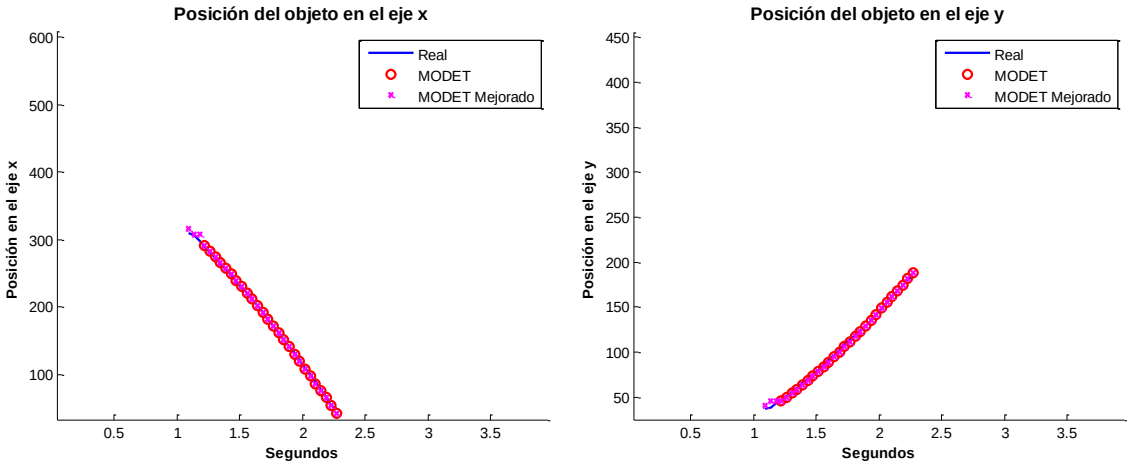

Figura A.10. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.3. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	1/47	0/47	1/47
Error medio	7.72 ± 2.35	9.65 ± 3.46	7.72 ± 2.35
Correlación en eje x	0.9994	0.9982	0.9994
Correlación en eje y	0.9997	0.9996	0.9997

Tabla A.4. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	3/29	3/29	0/29
Error medio	2.14 ± 1.26	13.74 ± 4.20	2.88 ± 2.55
Correlación en eje x	0.9999	0.9982	0.9998
Correlación en eje y	0.9997	0.9989	0.9986

Secuencia 4

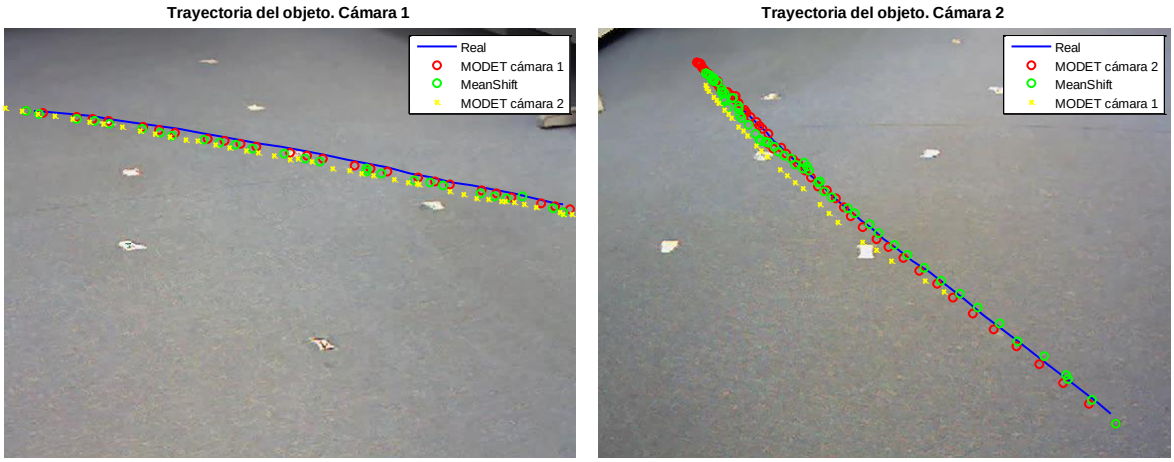

Figura A.11. Trayectorias del objeto a lo largo de la secuencia 4.

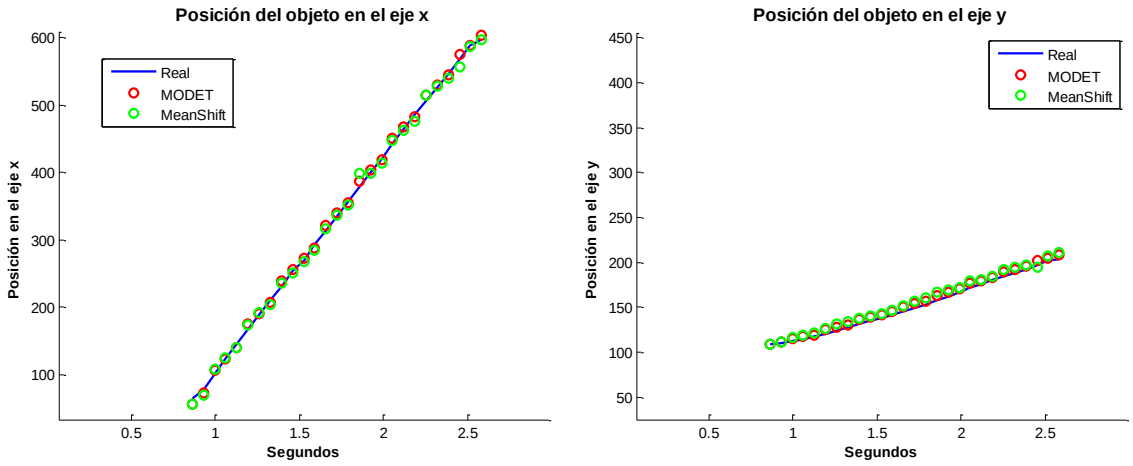

Figura A.12. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

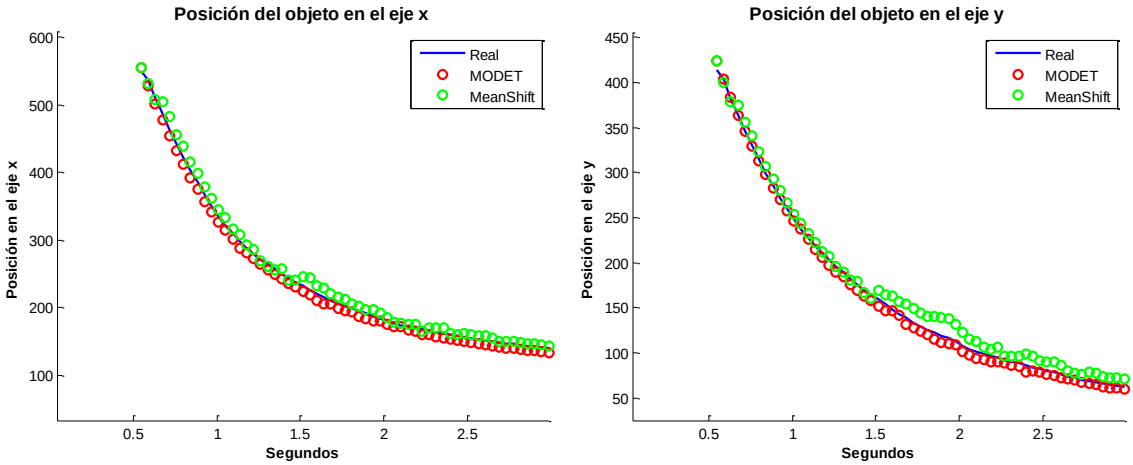

Figura A.13. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

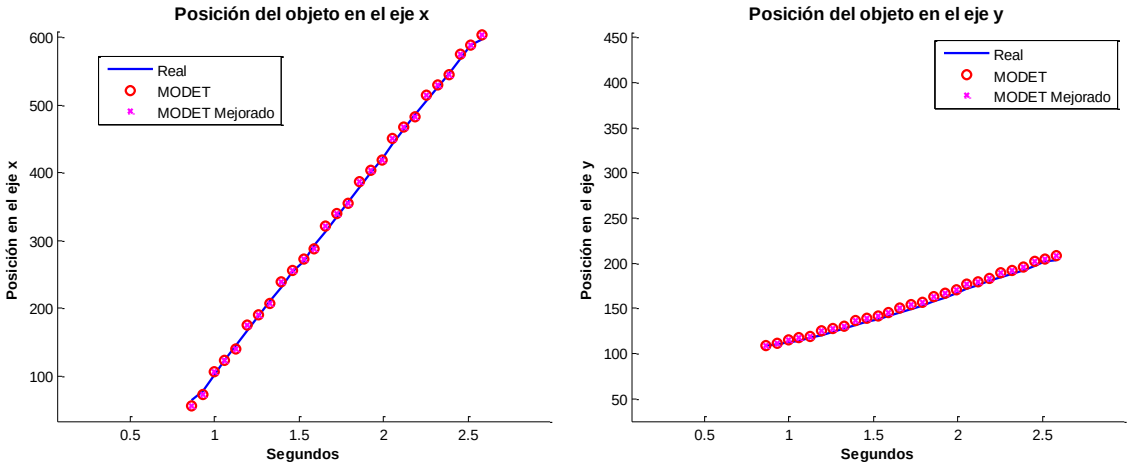

Figura A.14. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

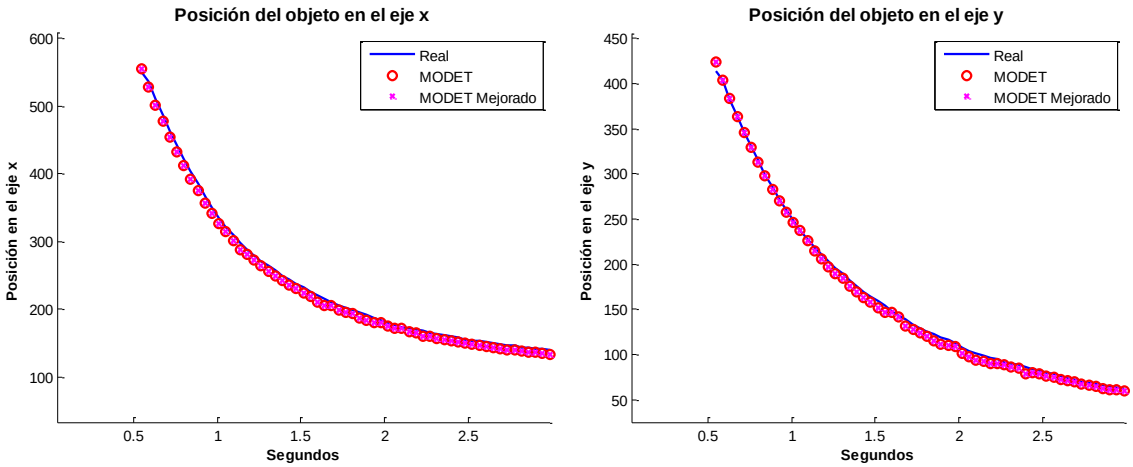

Figura A.15. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.5. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/27	0/27	0/27
Error medio	6.29 ± 2.58	8.48 ± 3.24	6.29 ± 2.58
Correlación en eje x	0.9996	0.9991	0.9996
Correlación en eje y	0.9992	0.9970	0.9992

Tabla A.6. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/59	0/59	0/59
Error medio	8.65 ± 1.57	12.16 ± 5.64	8.65 ± 1.57
Correlación en eje x	0.9998	0.9988	0.9998
Correlación en eje y	0.9997	0.9984	0.9997

Secuencia 5

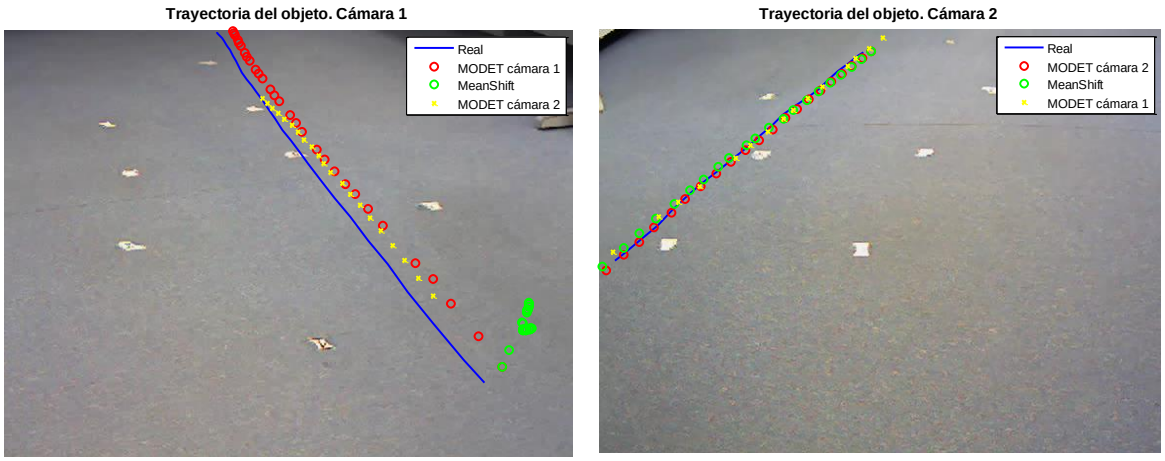

Figura A.16. Trayectorias del objeto a lo largo de la secuencia 5.

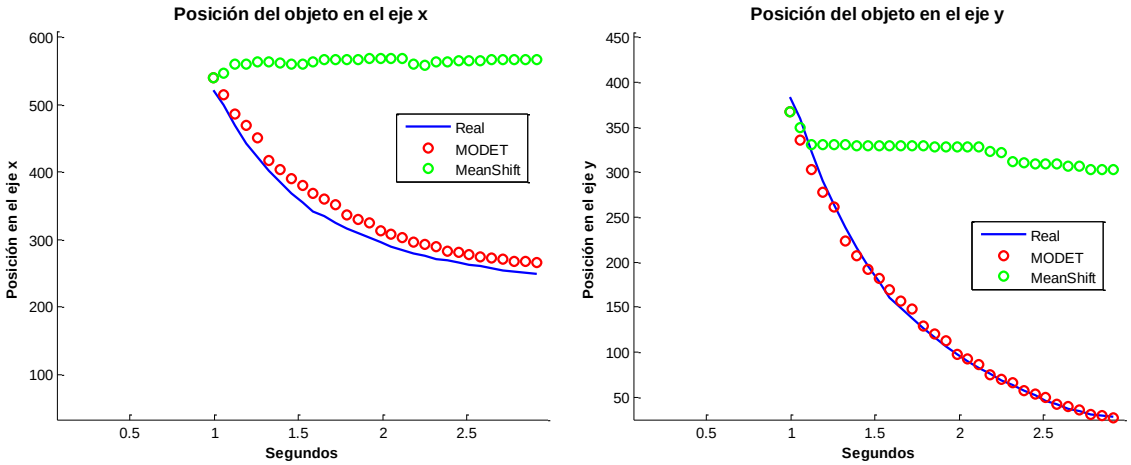

Figura A.17. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

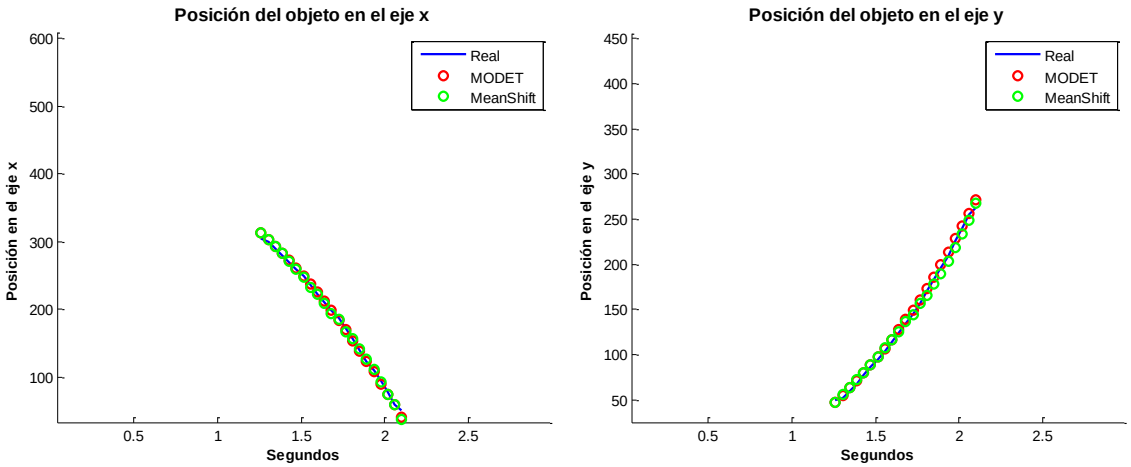

Figura A.18. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

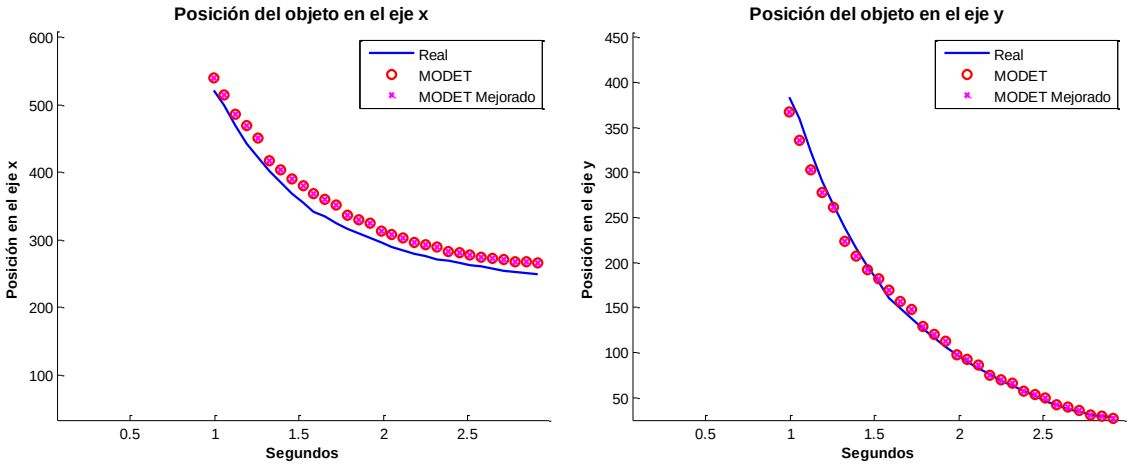

Figura A.19. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

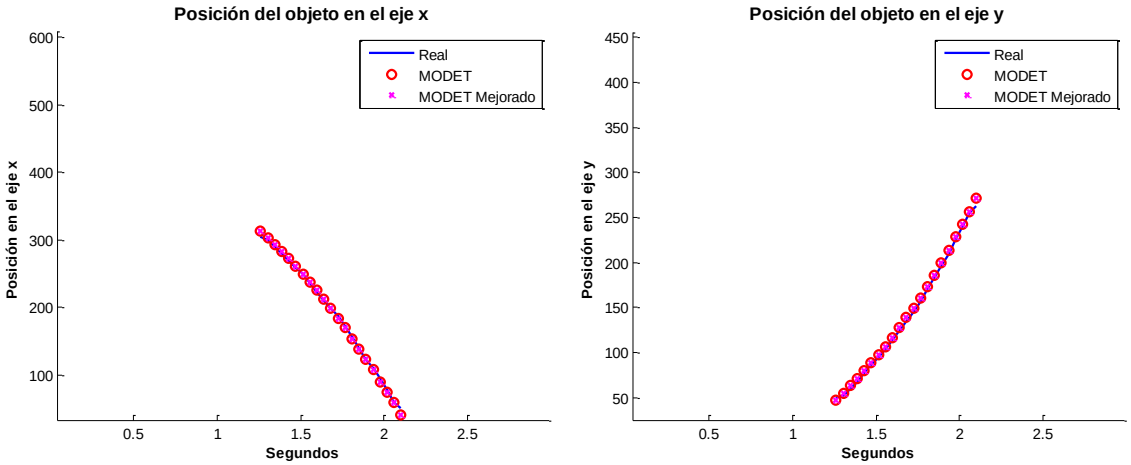

Figura A.20. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.7. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/30	0/30	0/30
Error medio	20.45 ± 4.98	303.73 ± 118.07	20.45 ± 4.98
Correlación en eje x	0.9988	-0.7423	0.9988
Correlación en eje y	0.9985	0.8517	0.9985

Tabla A.8. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/21	0/21	0/21
Error medio	4.49 ± 2.52	5.49 ± 2.57	4.49 ± 2.52
Correlación en eje x	0.9997	0.9990	0.9997
Correlación en eje y	0.9997	0.9987	0.9997

Secuencia 6

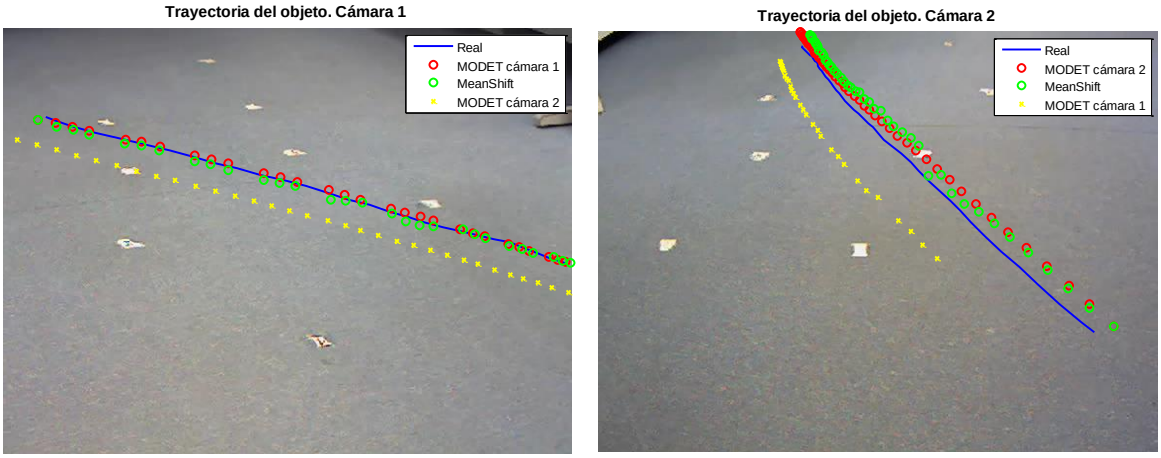

Figura A.21. Trayectorias del objeto a lo largo de la secuencia 6.

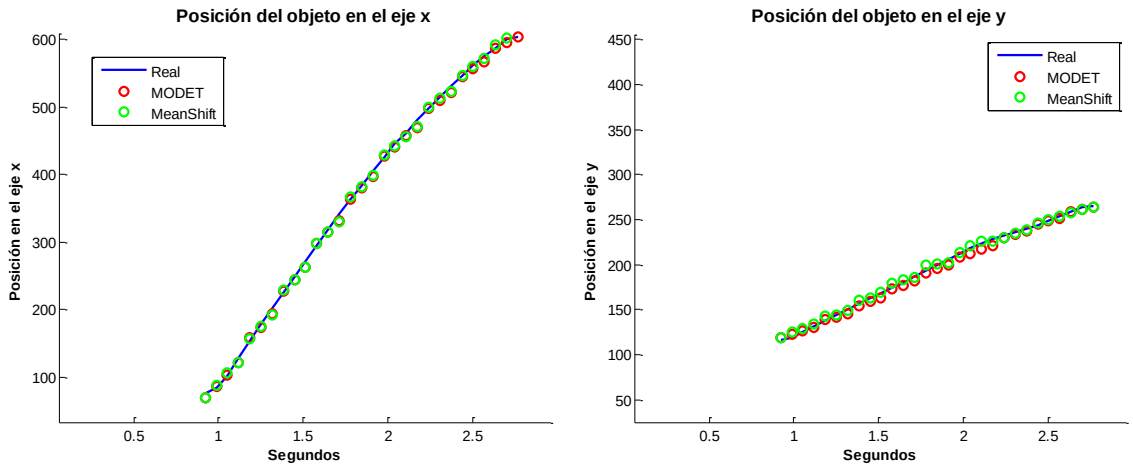

Figura A.22. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

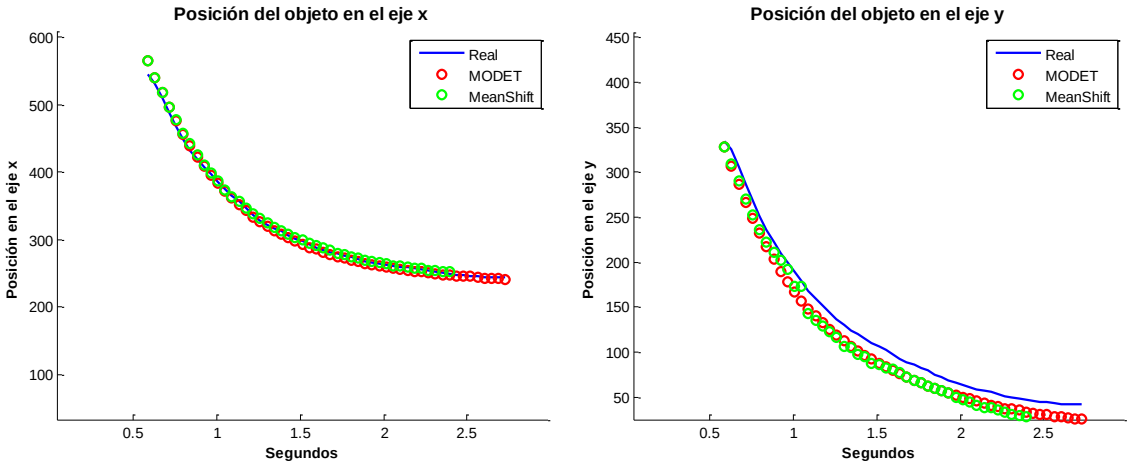

Figura A.23. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

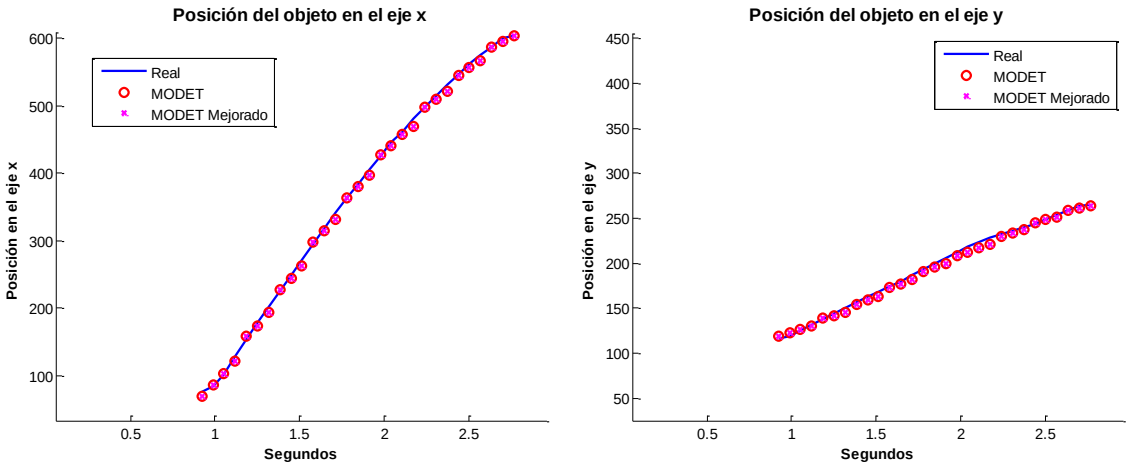

Figura A.24. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

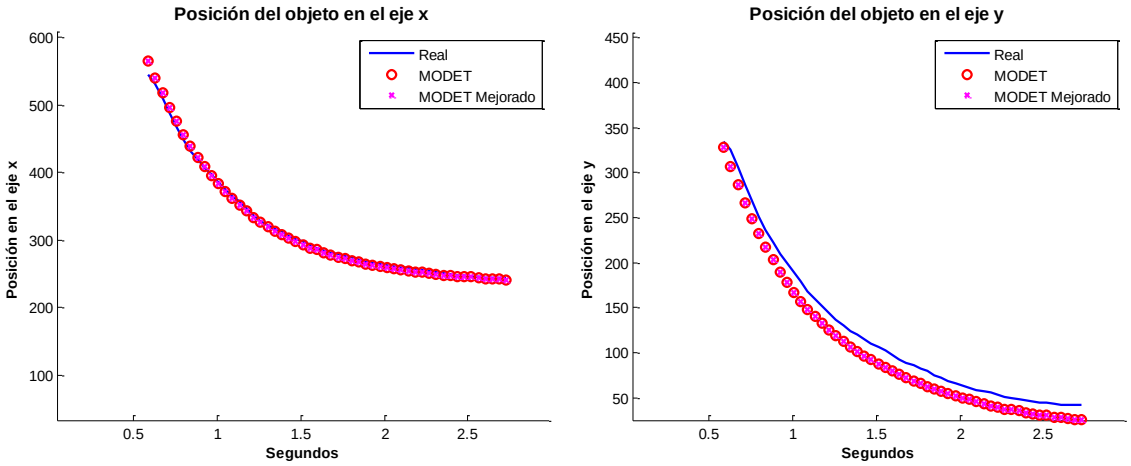

Figura A.25. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.9. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/29	0/29	0/29
Error medio	5.74 ± 3.11	5.62 ± 2.40	5.74 ± 3.11
Correlación en eje x	0.9997	0.9996	0.9997
Correlación en eje y	0.9987	0.9988	0.9987

Tabla A.10. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/52	8/52	0/52
Error medio	17.54 ± 2.81	17.90 ± 3.83	17.54 ± 2.81
Correlación en eje x	0.9995	0.9996	0.9995
Correlación en eje y	0.9995	0.9989	0.9995

Secuencia 7

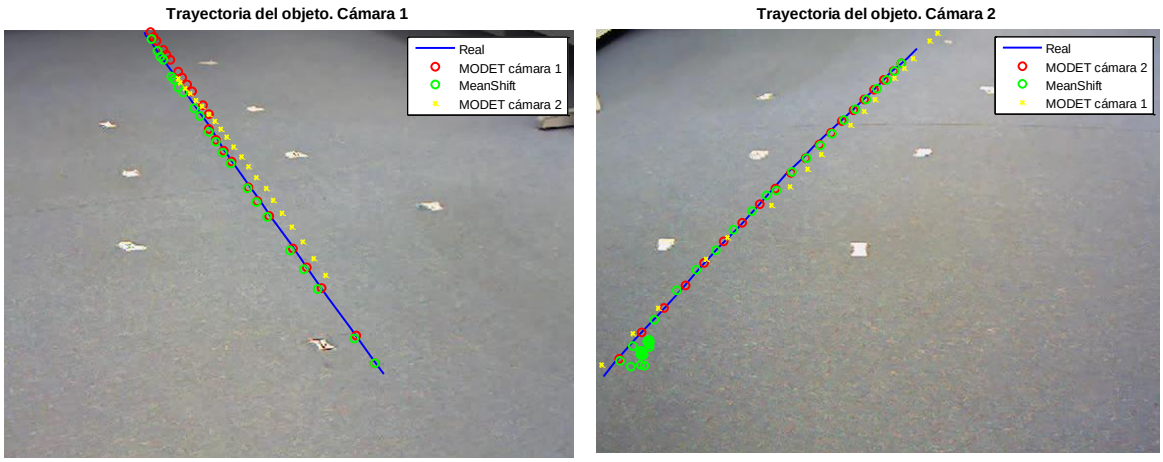

Figura A.26. Trayectorias del objeto a lo largo de la secuencia 7.

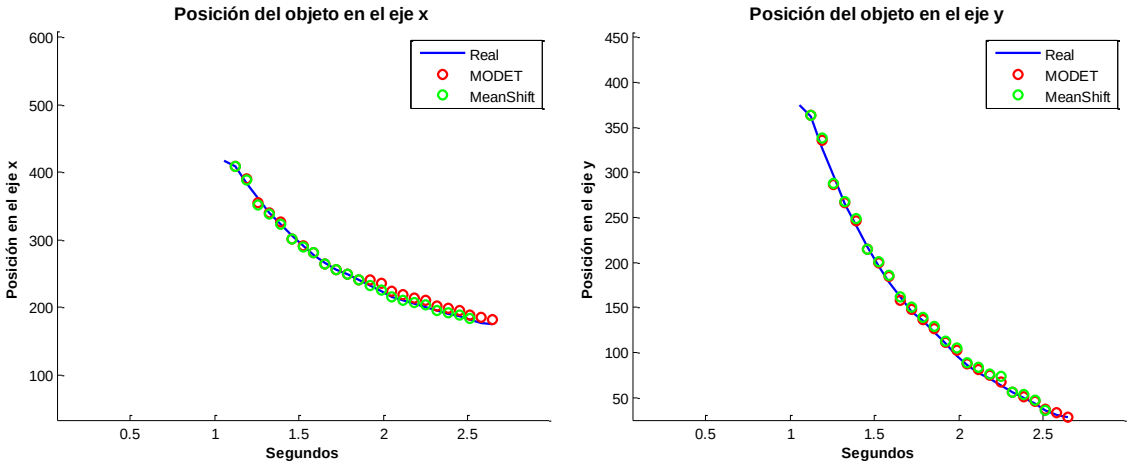

Figura A.27. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

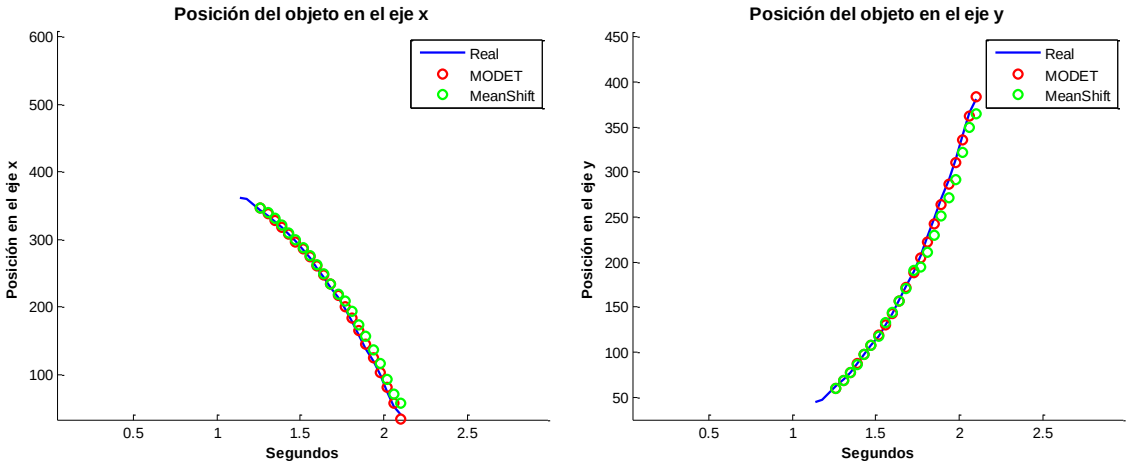

Figura A.28. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

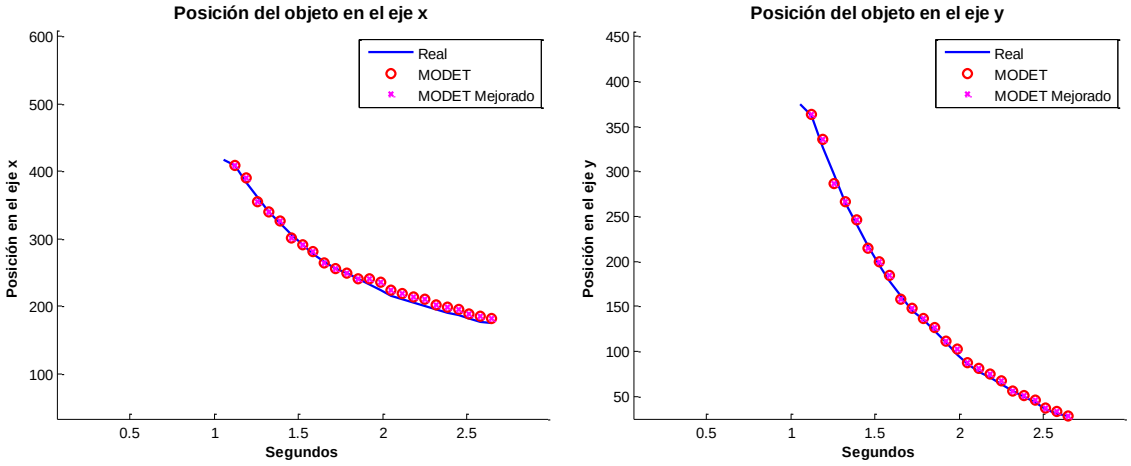

Figura A.29. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

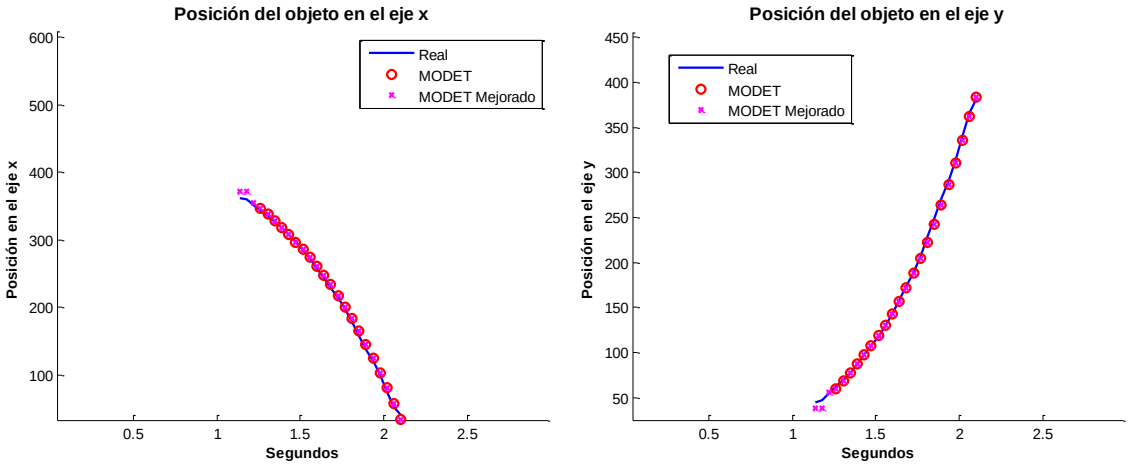

Figura A.30. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.11. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	1/25	3/25	1/25
Error medio	6.78 ± 3.38	5.72 ± 3.35	6.78 ± 3.38
Correlación en eje x	0.9983	0.9990	0.9983
Correlación en eje y	0.9993	0.9990	0.9993

Tabla A.12. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	3/24	3/24	0/24
Error medio	4.31 ± 2.81	12.58 ± 10.62	5.05 ± 3.77
Correlación en eje x	0.9995	0.9997	0.9994
Correlación en eje y	0.9997	0.9988	0.9996

Secuencia 8

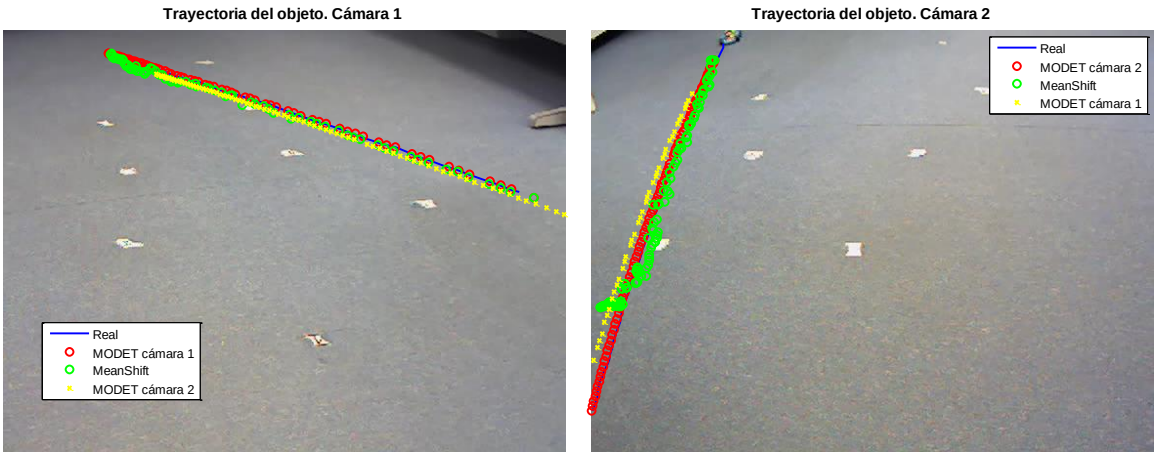

Figura A.31. Trayectorias del objeto a lo largo de la secuencia 8.

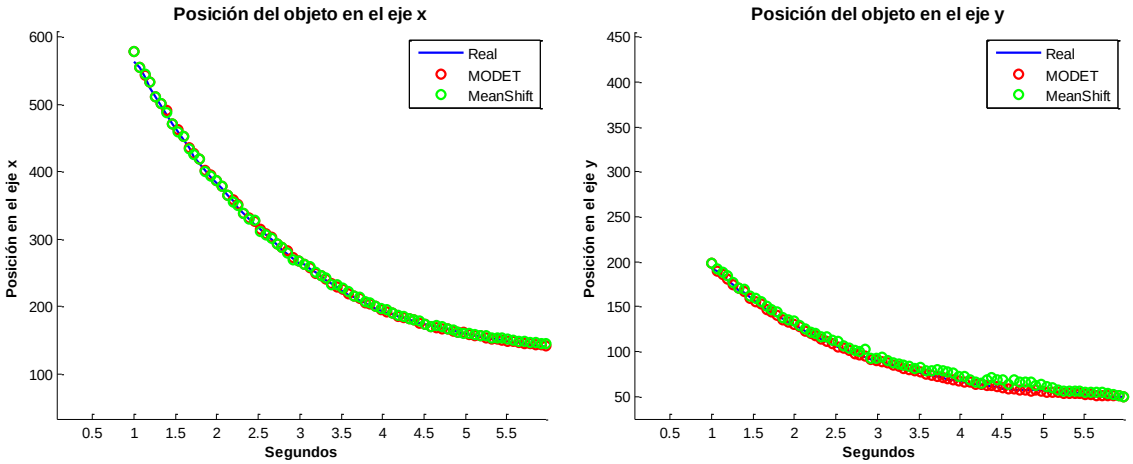

Figura A.32. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 1.

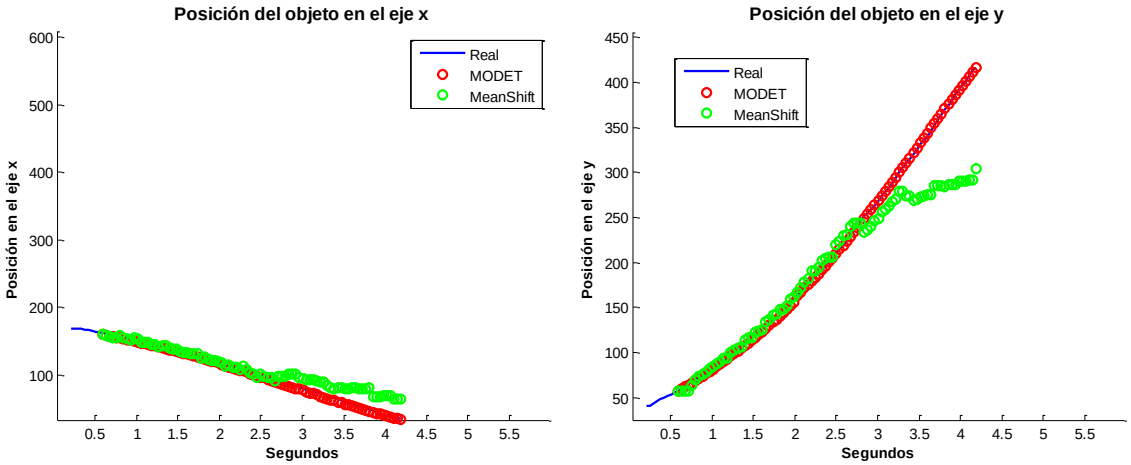

Figura A.33. Posición real del objeto frente a posición MODET y Mean Shift desde la cámara 2.

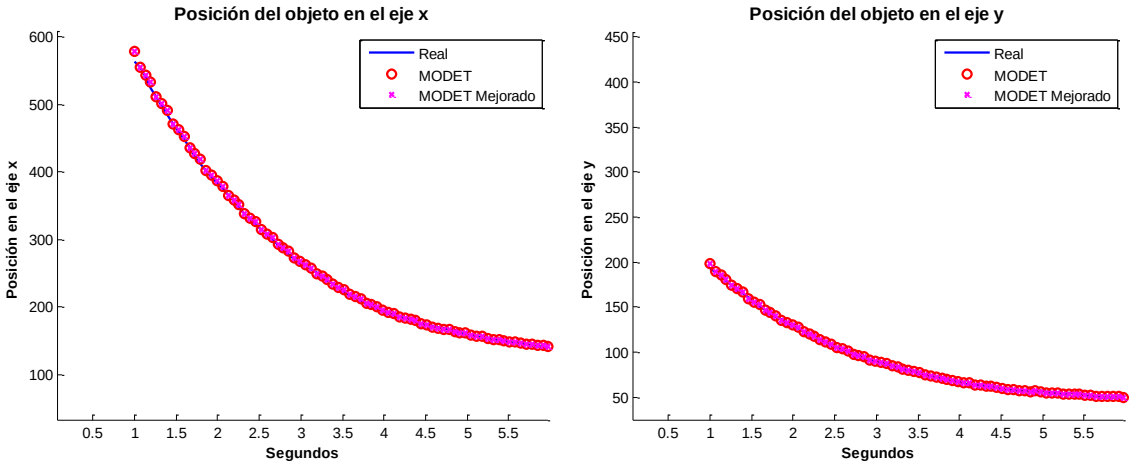

Figura A.34. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 1

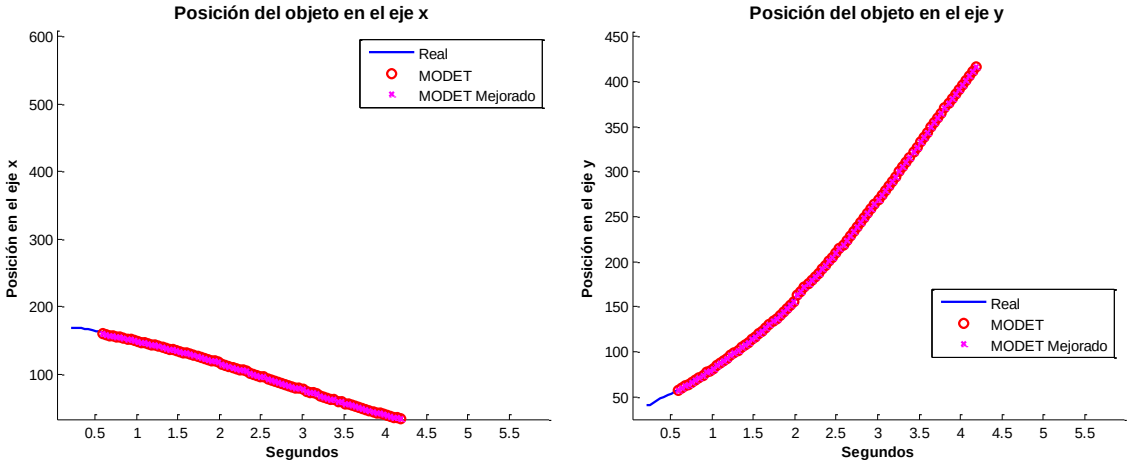

Figura A.35. Posición real del objeto frente a posición MODET y MODET Mejorado desde la cámara 2

Tabla A.13. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 1

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	0/76	0/76	0/76
Error medio	2.21 ± 2.38	5.47 ± 2.54	2.21 ± 2.38
Correlación en eje x	0.9999	0.9998	0.9999
Correlación en eje y	0.9998	0.9984	0.9998

Tabla A.14. Comparativa entre los algoritmos MODET, Mean Shift y MODET Mejorado. Cámara 2.

	MODET	Mean Shift	MODET Mejorado
Número de frames en los que no se detecta el objeto	9/95	9/95	9/95
Error medio	3.12 ± 1.03	27.33 ± 35.71	3.12 ± 1.03
Correlación en eje x	0.9997	0.9861	0.9997
Correlación en eje y	0.9999	0.9676	0.9999

A.2 Escenario interior, cámara fija y múltiples objetos

A continuación recogemos los resultados de las pruebas realizadas en vídeos con varios objetos en movimiento grabados en un escenario interior.

Secuencia 2

Figura A.36. Objetos vistos desde la cámara 1.

Figura A.37. Objetos vistos desde la cámara 2.

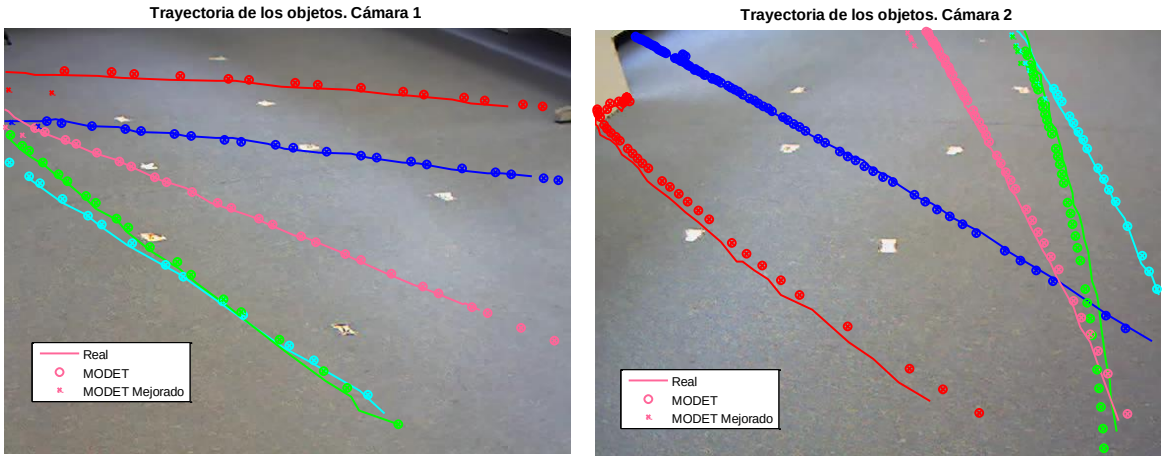

Figura A.38. Trayectorias de los objetos a lo largo de la secuencia 2.

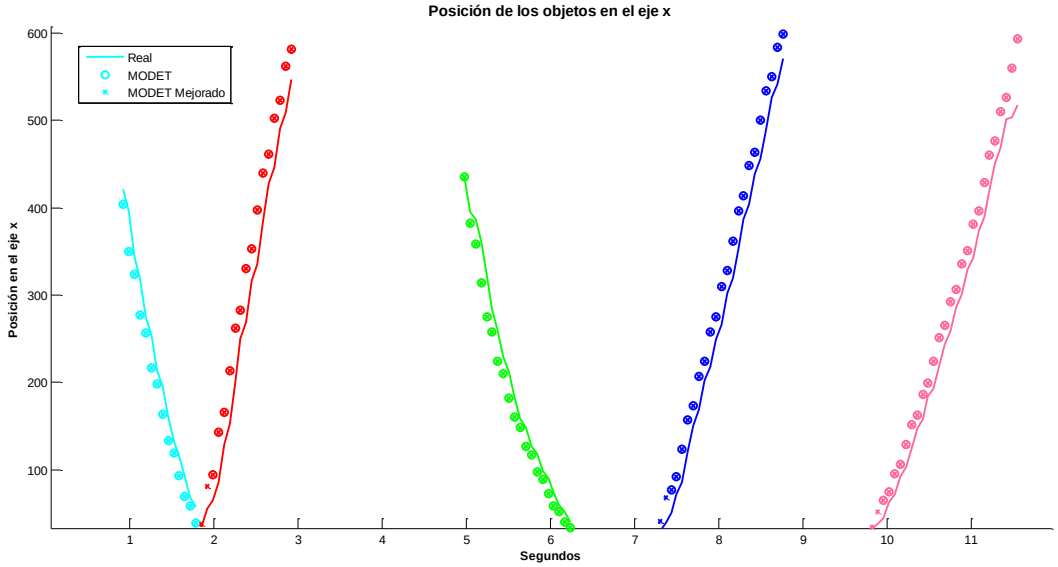

Figura A.39. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 1.

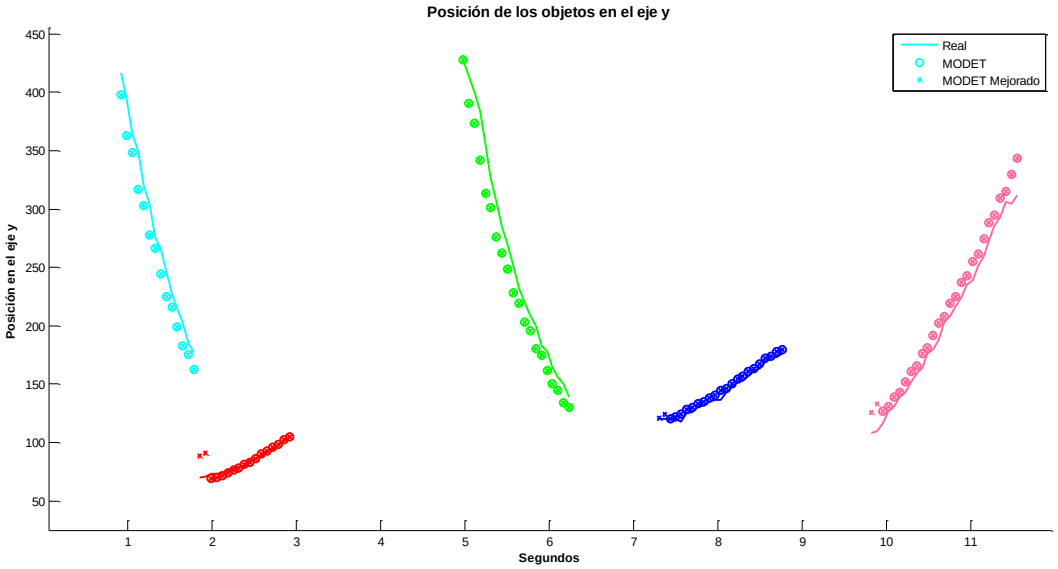

Figura A.40. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje y desde la cámara 1.

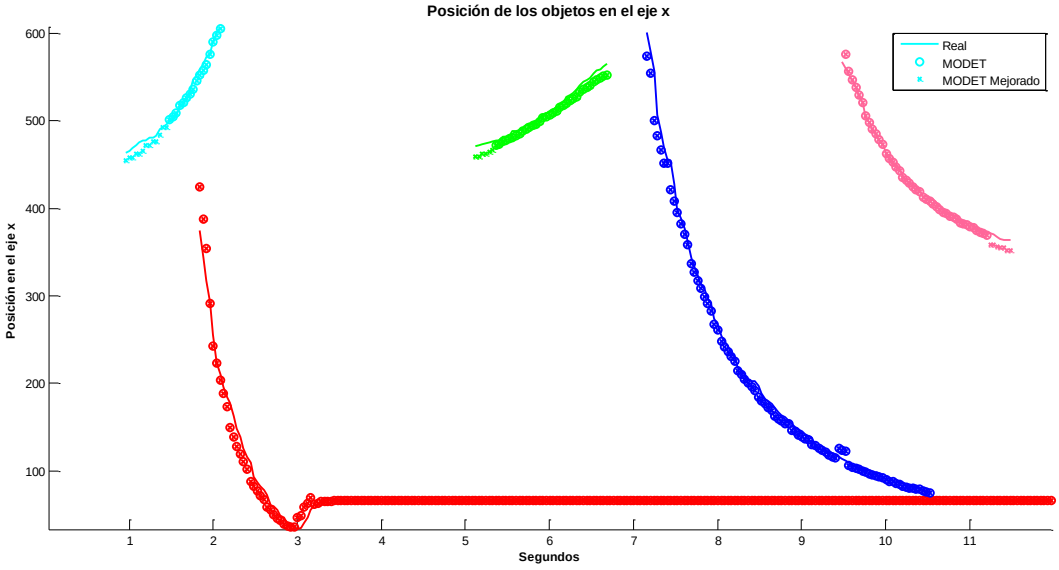

Figura A.41. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 2.

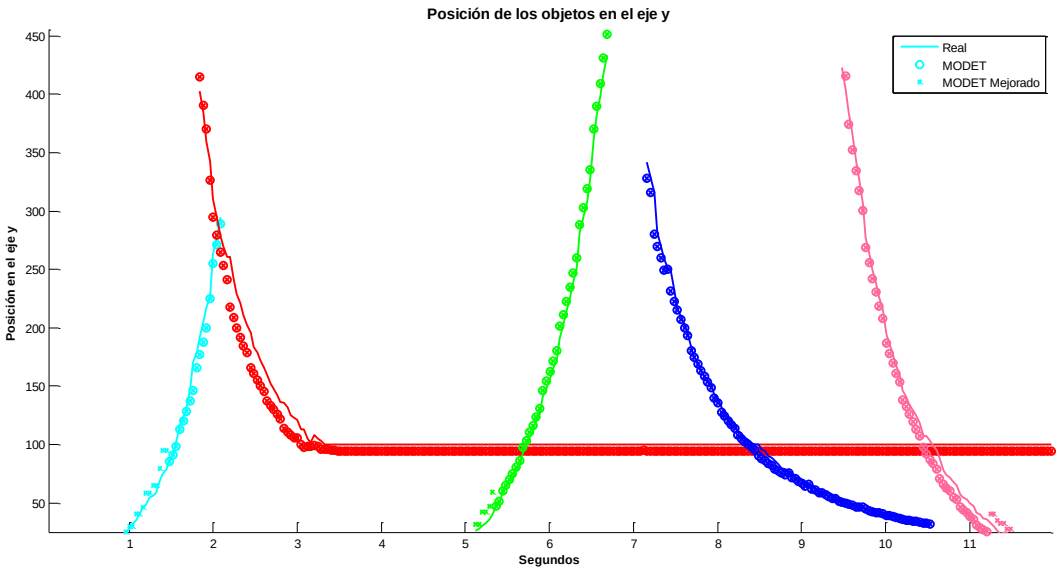

Figura A.42. Posición real de los objeto frente a posición MODET y MODET Mejorado en el eje y desde la cámara 2.

Tabla A.15. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	6/101	0/101
Total	Error medio	33.78 ± 14.60	33.15 ± 14.50
	Correlación en eje x	0.9835	0.9850
	Correlación en eje y	0.9889	0.9888

Tabla A.16. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	27/459	1/459
Total	Error medio	8.56 ± 7.65	8.84 ± 7.54
	Correlación en eje x	0.9993	0.9993
	Correlación en eje y	0.9959	0.9951

Secuencia 3

Figura A.43. Objetos vistos desde la cámara 1.

Figura A.44. Objetos vistos desde la cámara 2.

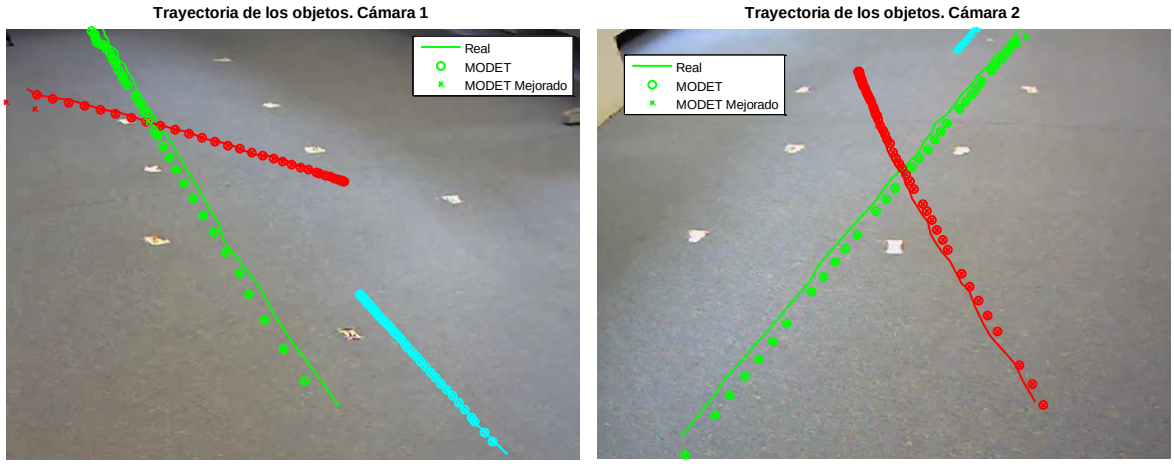

Figura A.45. Trayectorias de los objetos a lo largo de la secuencia 3.

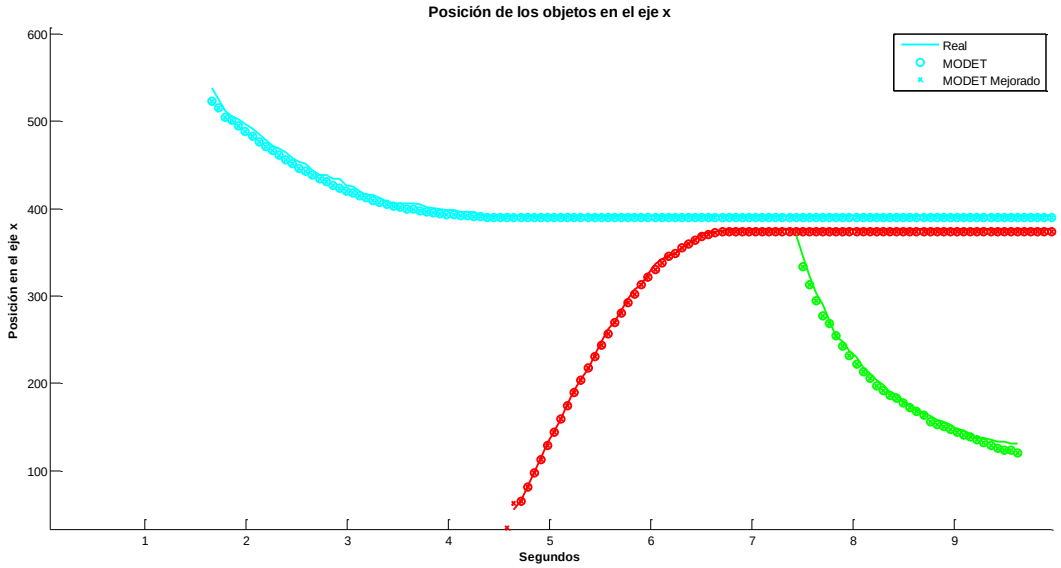

Figura A.46. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 1.

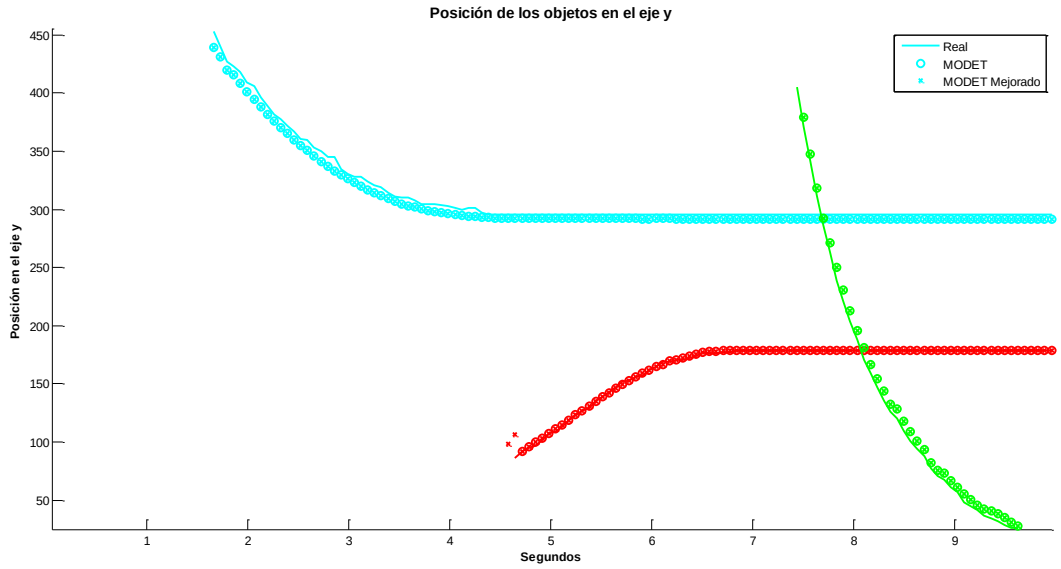

Figura A.47. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje y desde la cámara 1.

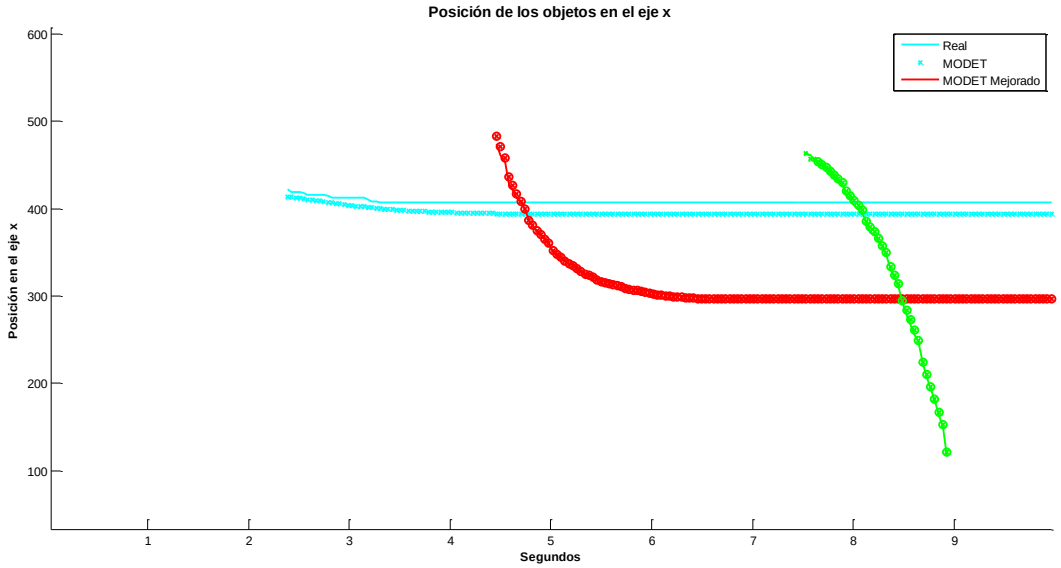

Figura A.48. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 2.

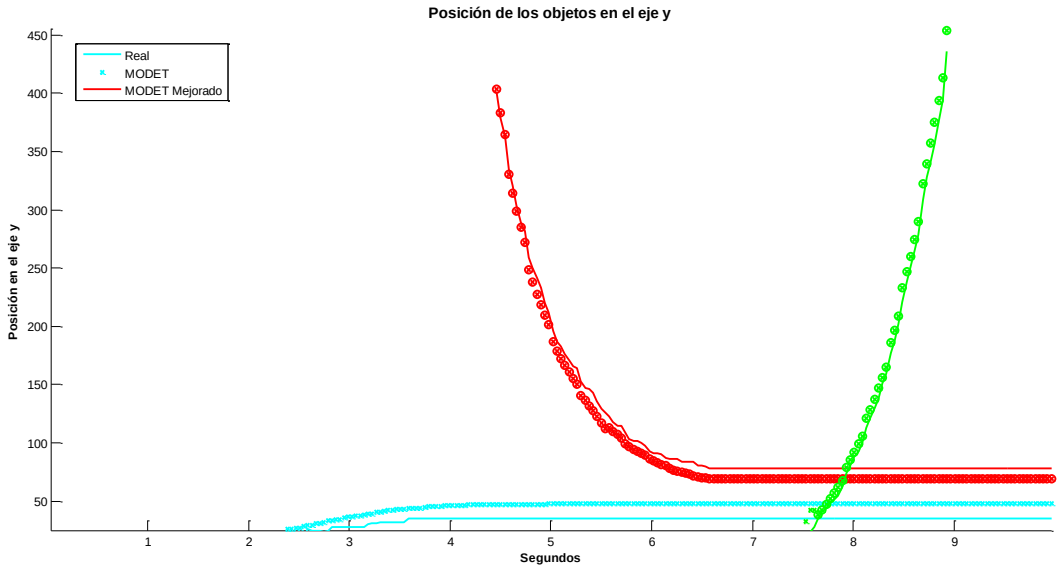

Figura A.49. Posición real de los objeto frente a posición MODET y MODET Mejorado en el eje y desde la cámara 2.

Tabla A.17. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
Total	Número de frames en los que no se detecta el objeto	2/241	1/241
	Error medio	5.48 ± 3.12	5.55 ± 3.29
	Correlación en eje x	0.9997	0.9996
	Correlación en eje y	0.9995	0.9995

Tabla A.18. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	194/366	0/366
Total	Error medio	9.44 ± 2.42	13.32 ± 4.47
	Correlación en eje x	0.9991	0.9962
	Correlación en eje y	0.9974	0.9905

Secuencia 4

Figura A.50. Objetos vistos desde la cámara 1.

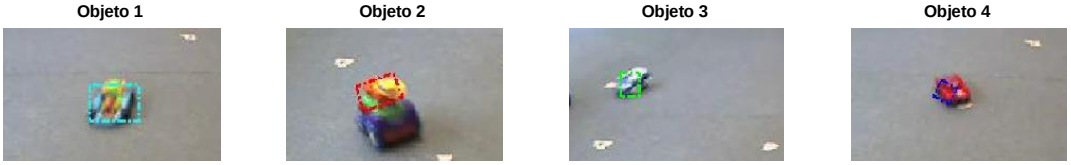

Figura A.51. Objetos vistos desde la cámara 2.

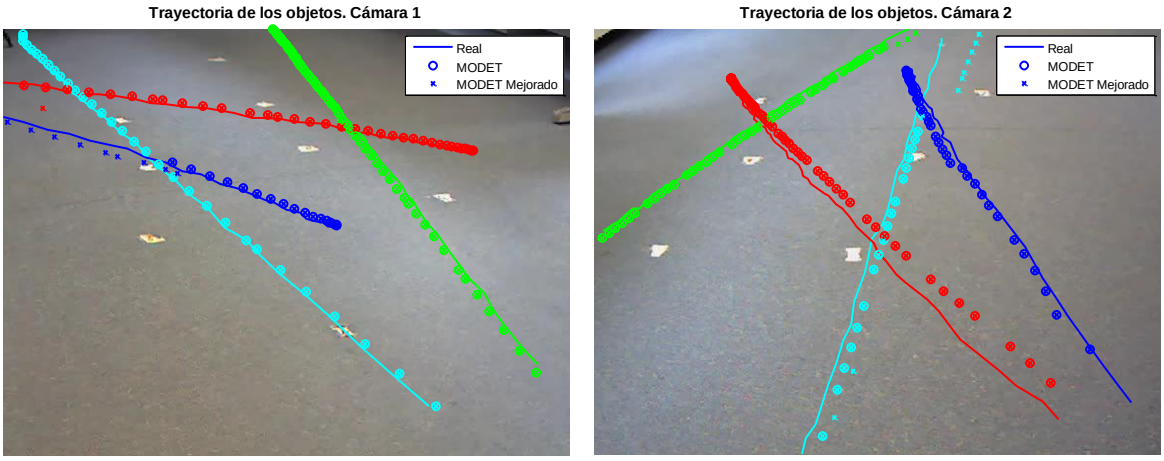

Figura A.52. Trayectorias de los objetos a lo largo de la secuencia 4.

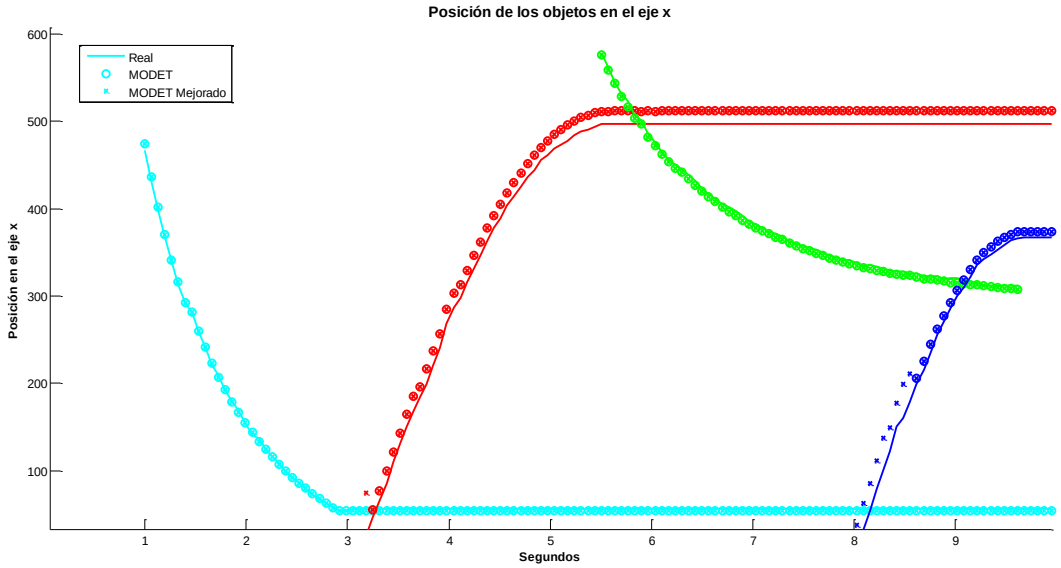

Figura A.53. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 1.

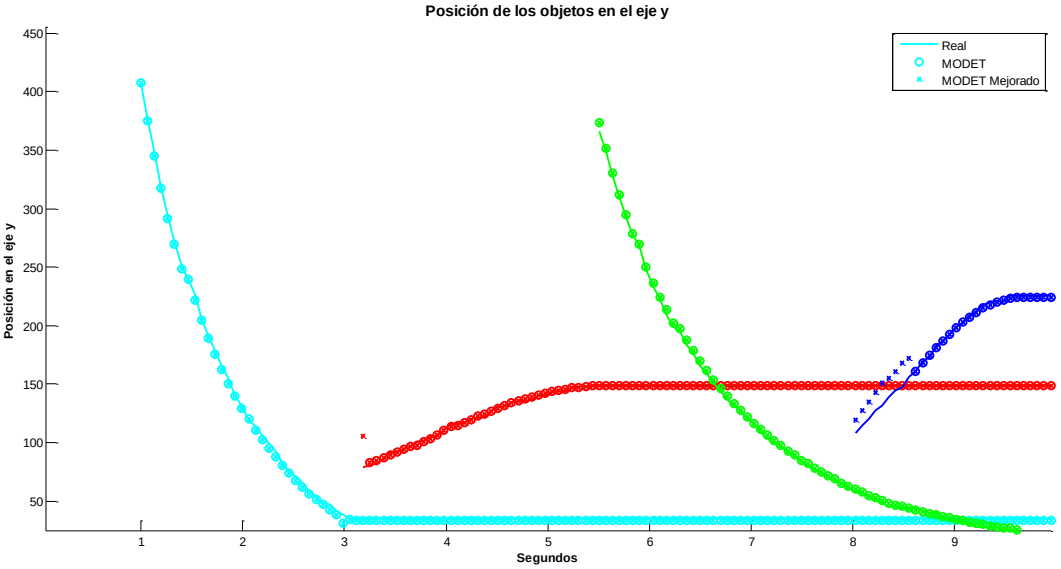

Figura A.54. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje y desde la cámara 1.

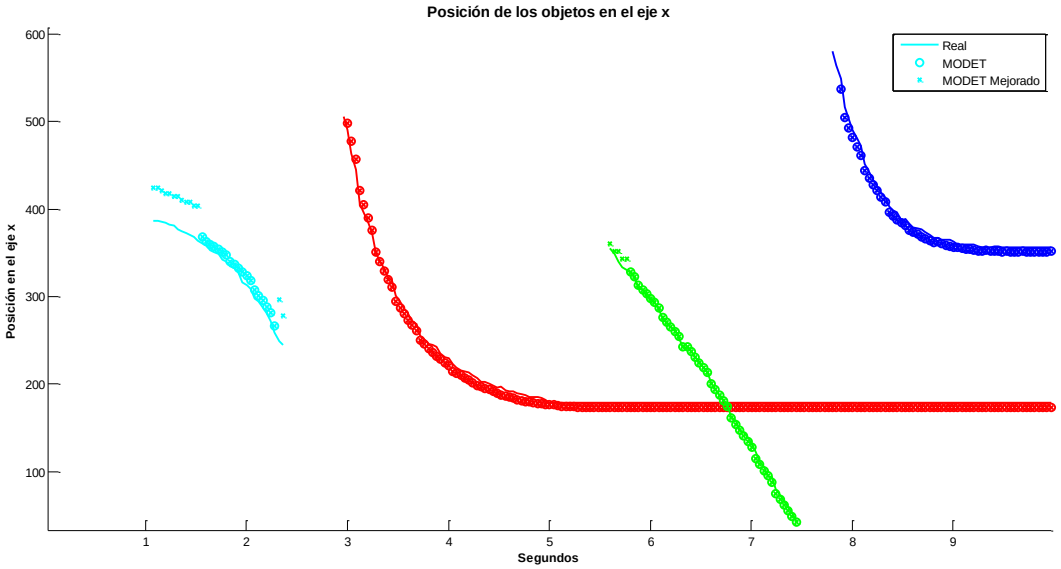

Figura A.55. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 2.

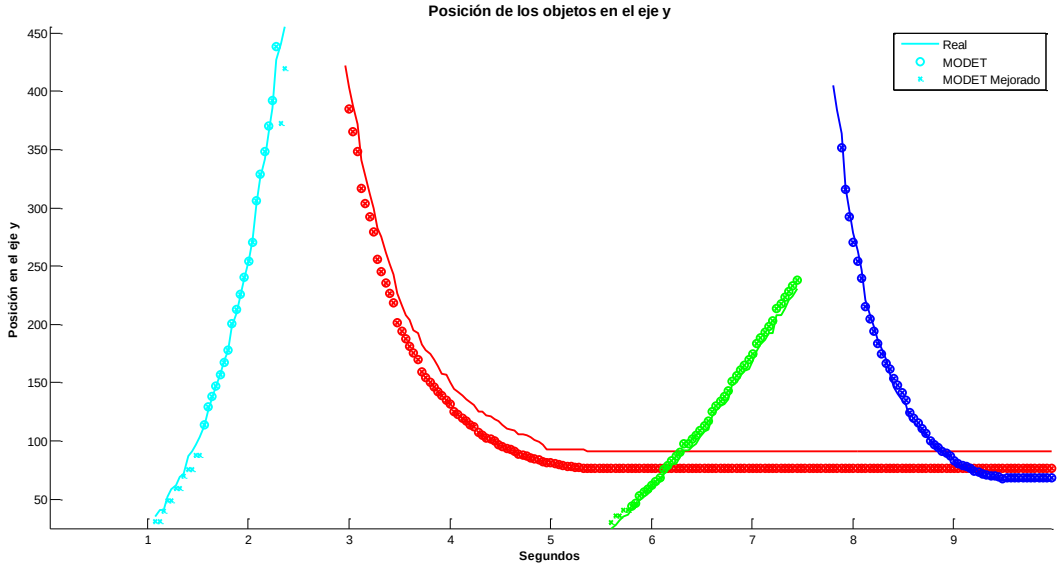

Figura A.56. Posición real de los objeto frente a posición MODET y MODET Mejorado en el eje y desde la cámara 2.

Tabla A.19. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
Total	Número de frames en los que no se detecta el objeto	10/332	0/332
	Error medio	7.00 ± 5.58	7.91 ± 7.66
	Correlación en eje x	0.9995	0.9990
	Correlación en eje y	0.9998	0.9990

Tabla A.20. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	6/113	0/113
Total	Error medio	13.25 ± 6.01	14.49 ± 8.71
	Correlación en eje x	0.9990	0.9959
	Correlación en eje y	0.9908	0.9906

Secuencia 5

Figura A.57. Objetos vistos desde la cámara 1.

Figura A.58. Objetos vistos desde la cámara 2.

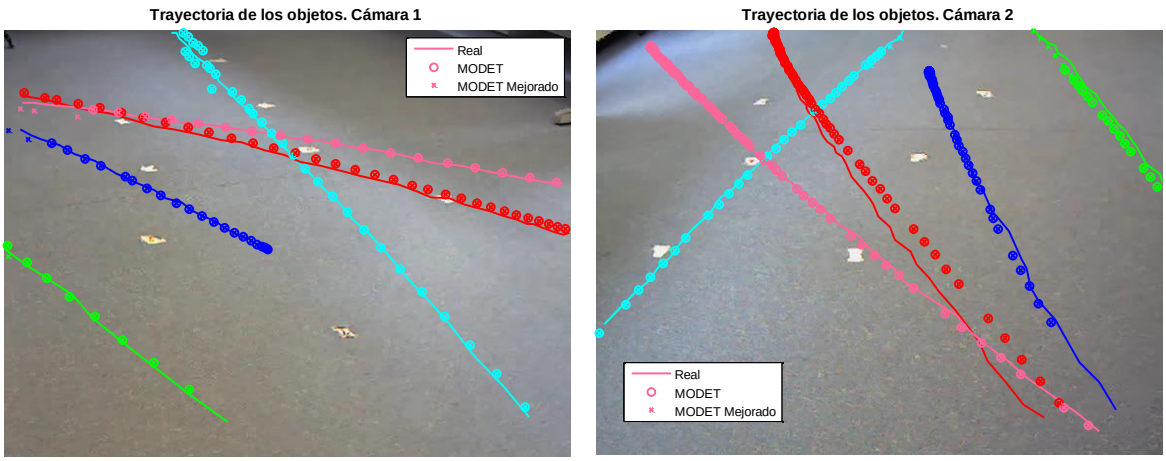

Figura A.59. Trayectorias de los objetos a lo largo de la secuencia 5.

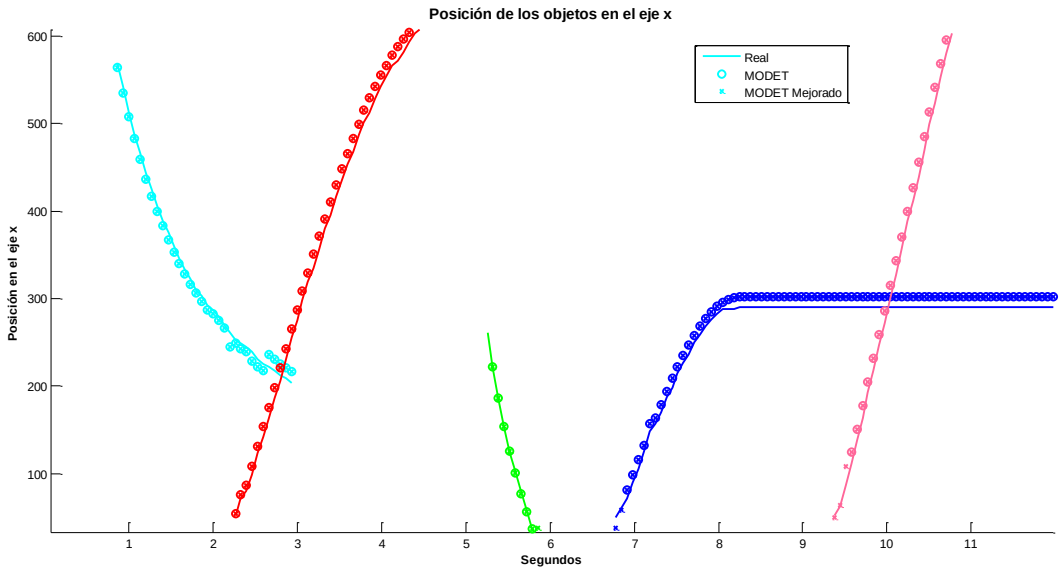

Figura A.60. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 1.

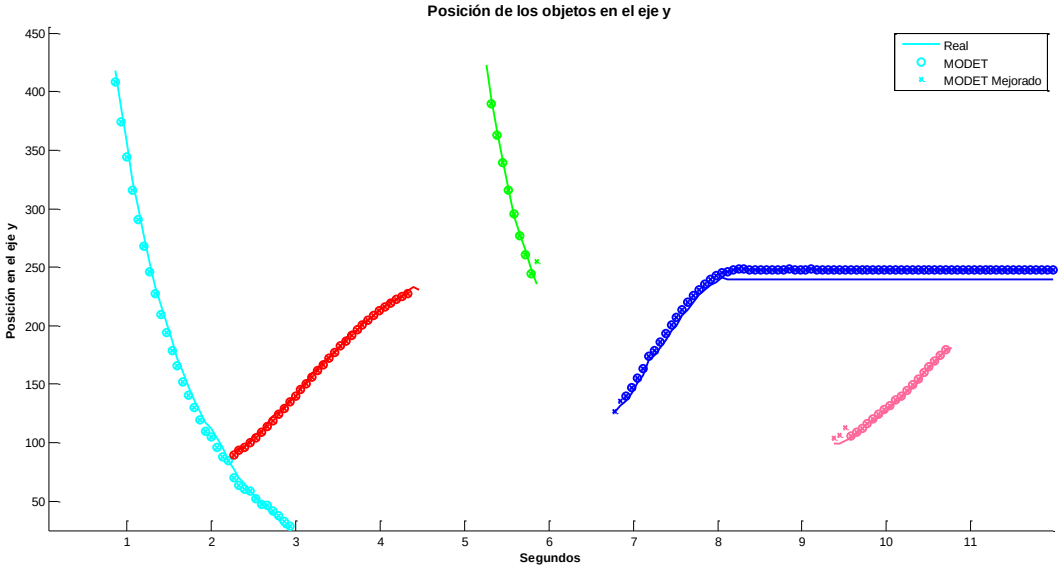

Figura A.61. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje y desde la cámara 1.

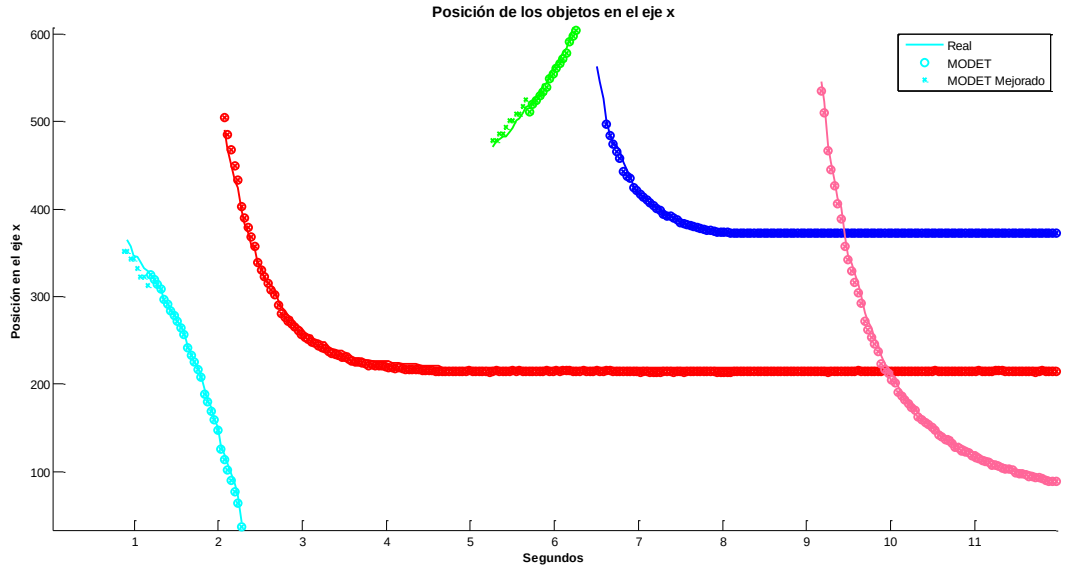

Figura A.62. Posición real de los objetos frente a posición MODET y MODET Mejorado en el eje x desde la cámara 2.

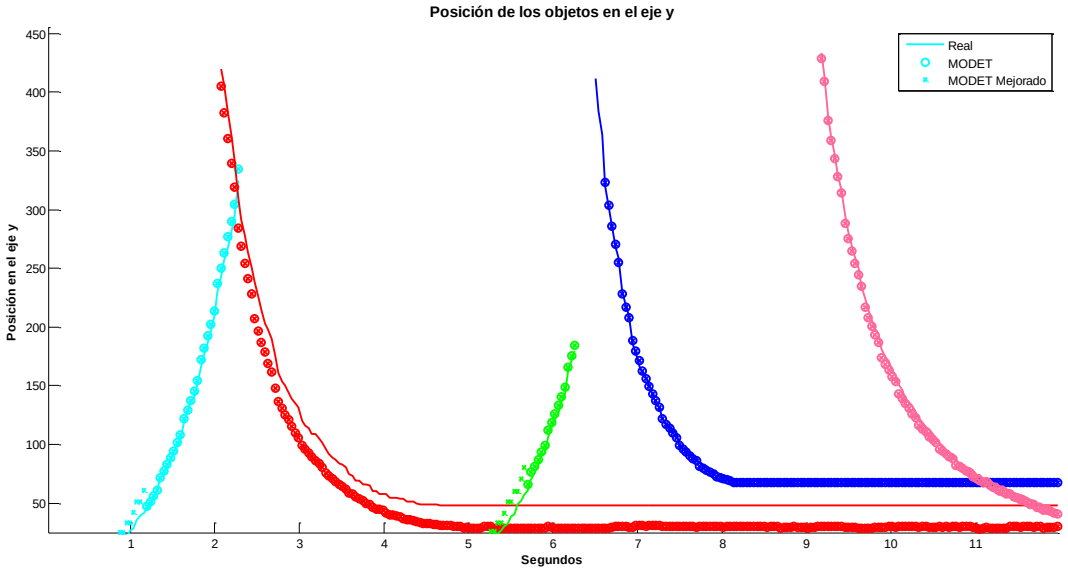

Figura A.63. Posición real de los objeto frente a posición MODET y MODET Mejorado en el eje y desde la cámara 2.

Tabla A.21. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
Total	Número de frames en los que no se detecta el objeto	10/177	4/177
	Error medio	12.27 ± 3.22	12.28 ± 3.46
	Correlación en eje x	0.9981	0.9982
	Correlación en eje y	0.9971	0.9970

Tabla A.22. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	21/519	3/519
Total	Error medio	11.86 ± 8.17	12.01 ± 8.16
	Correlación en eje x	0.9995	0.9995
	Correlación en eje y	0.9929	0.9910

A.3 Escenario exterior, cámara fija y múltiples objetos

A continuación recogemos los resultados de las pruebas realizadas en vídeos con varios objetos en movimiento grabados con dos cámaras fijas en un escenario exterior.

Secuencia 2

Figura A.64. Objetos vistos desde la cámara 1.

Figura A.65. Objetos vistos desde la cámara 2.

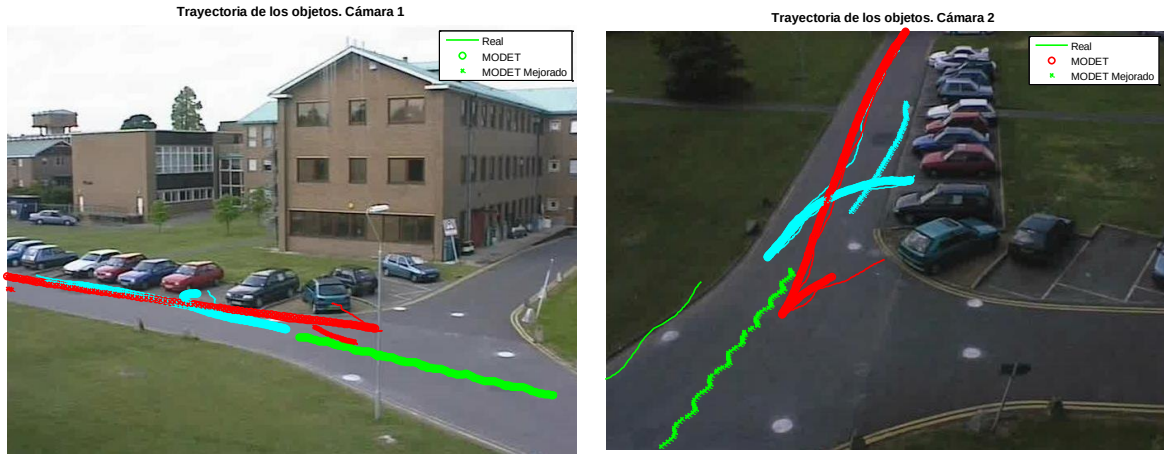

Figura A.66. Trayectorias de los objetos a lo largo de la secuencia 2.

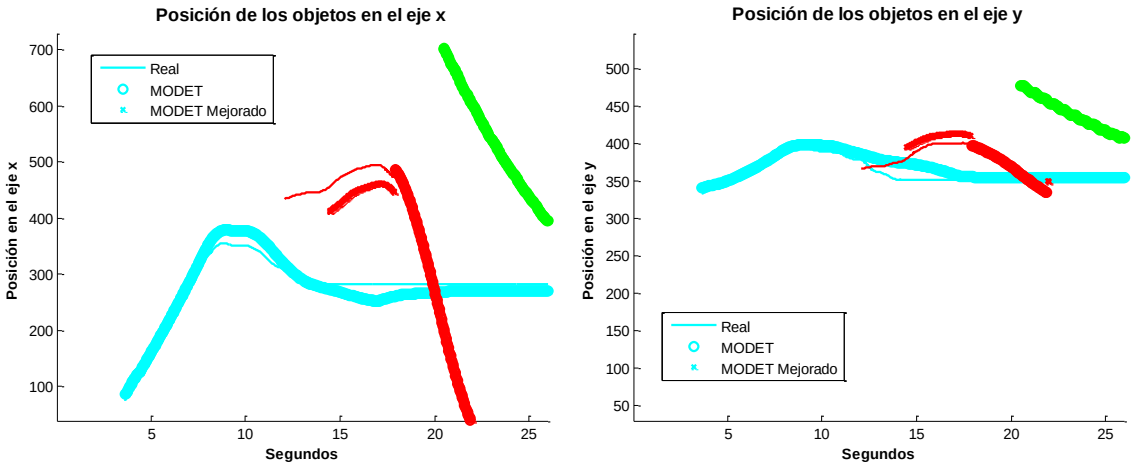

Figura A.67. Posición real de los objetos frente a posición MODET y MODET Mejorado desde la cámara 1.

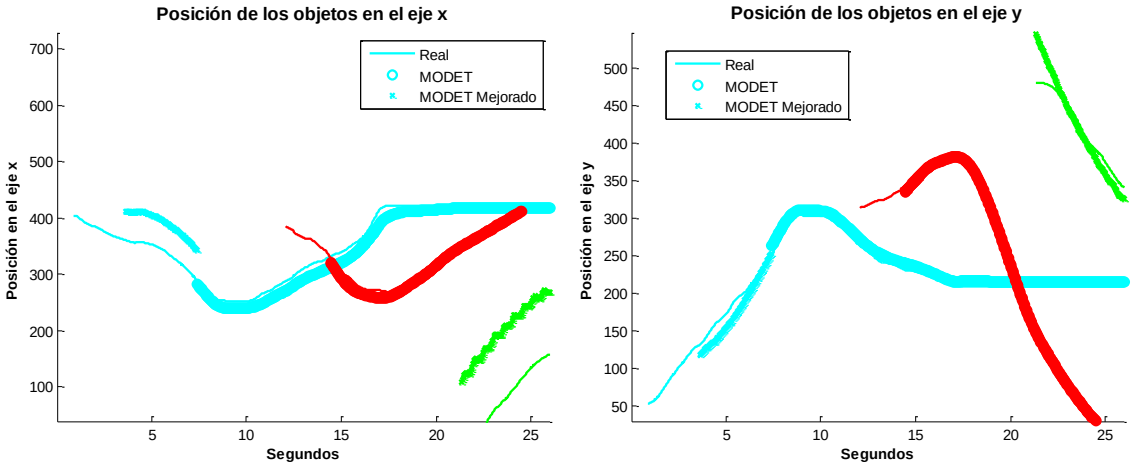

Figura A.68. Posición real de los objeto frente a posición MODET y MODET Mejorado desde la cámara 2.

Tabla A.23. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	155/951	65/951
Total	Error medio	15.35 ± 7.79	17.66 ± 10.34
	Correlación en eje x	0.9927	0.9900
	Correlación en eje y	0.9850	0.9836

Tabla A.24. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	340/1056	130/1056
Total	Error medio	12.15 ± 5.23	30.62 ± 38.26
	Correlación en eje x	0.9968	0.9203
	Correlación en eje y	0.9986	0.9953

Secuencia 3

Figura A.69. Objetos vistos desde la cámara 1.

Figura A.70. Objetos vistos desde la cámara 2.

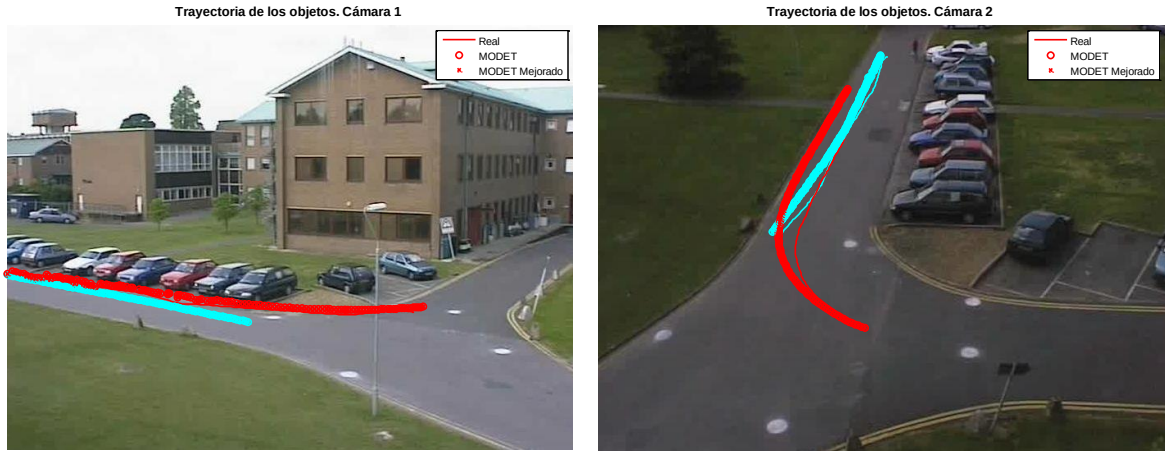

Figura A.71. Trayectorias de los objetos a lo largo de la secuencia 3.

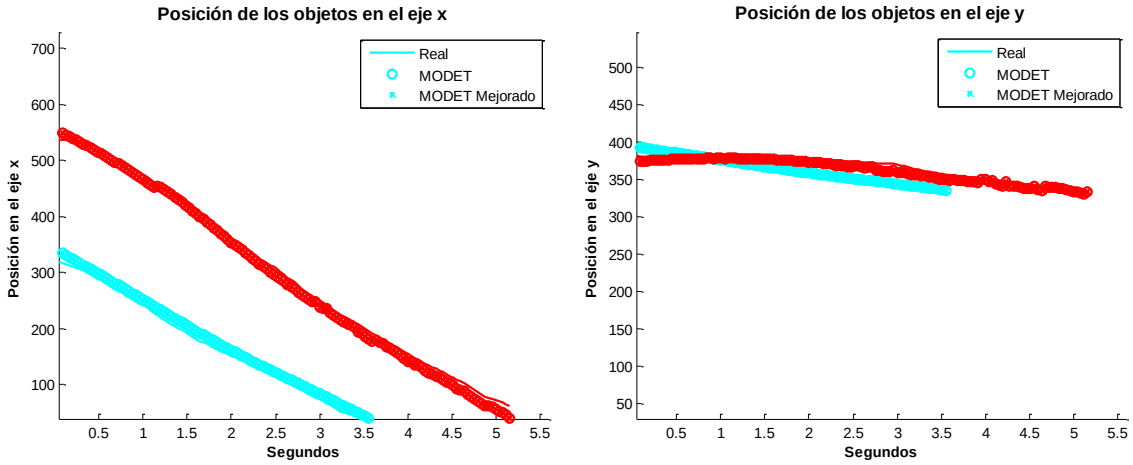

Figura A.72. Posición real de los objetos frente a posición MODET y MODET Mejorado desde la cámara 1.

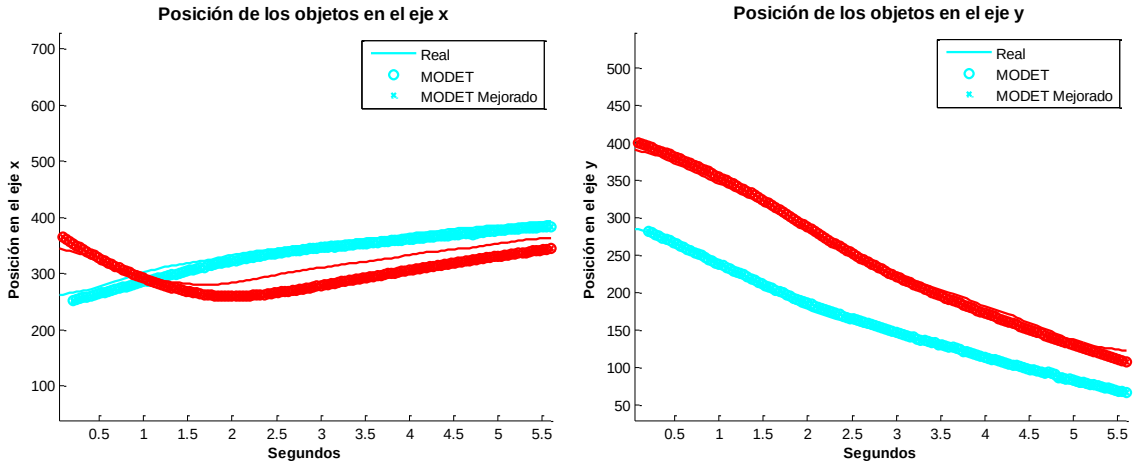

Figura A.73. Posición real de los objeto frente a posición MODET y MODET Mejorado desde la cámara 2.

Tabla A.25. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	2/218	2/218
Total	Error medio	11.31 ± 3.24	11.31 ± 3.24
	Correlación en eje x	0.9971	0.9971
	Correlación en eje y	0.9673	0.9673

Tabla A.26. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	5/280	5/280
Total	Error medio	17.31 ± 8.35	17.31 ± 8.35
	Correlación en eje x	0.9600	0.9600
	Correlación en eje y	0.9981	0.9981

Secuencia 4

Figura A.74. Objetos vistos desde la cámara 1.

Figura A.75. Objetos vistos desde la cámara 2.

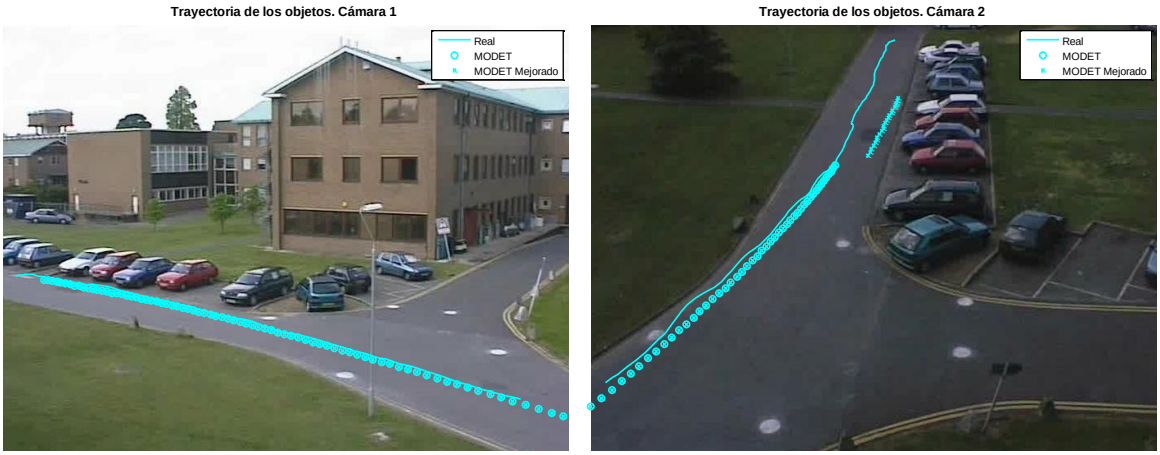

Figura A.76. Trayectorias de los objetos a lo largo de la secuencia 4.

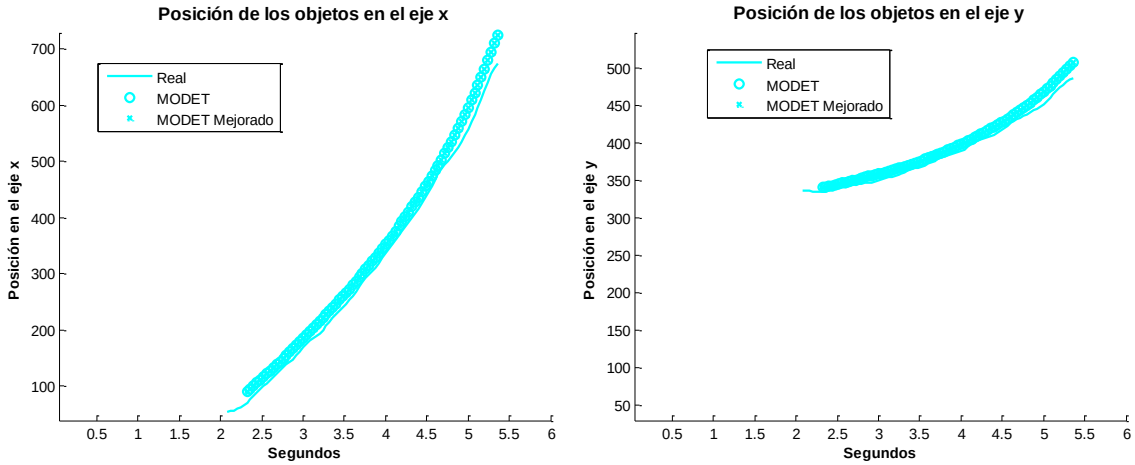

Figura A.77. Posición real de los objetos frente a posición MODET y MODET Mejorado desde la cámara 1.

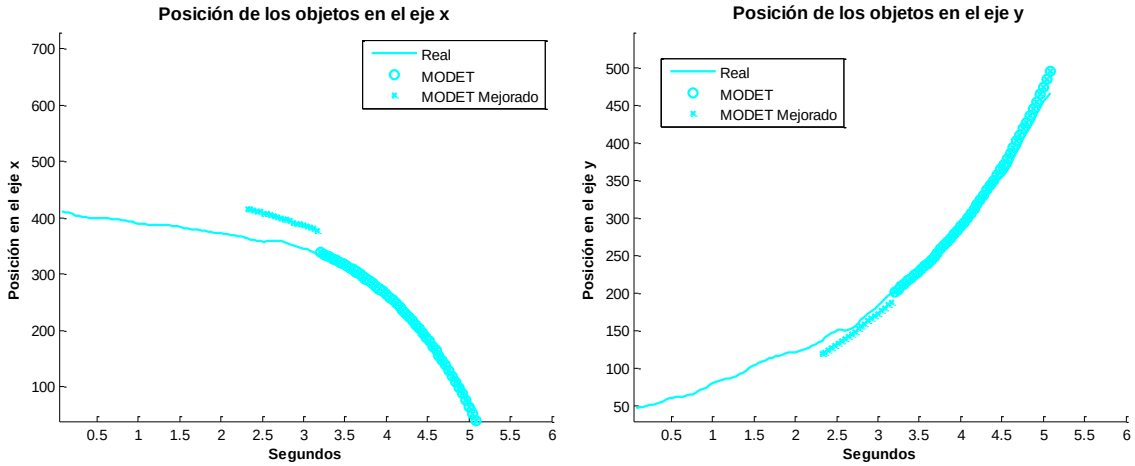

Figura A.78. Posición real de los objeto frente a posición MODET y MODET Mejorado desde la cámara 2.

Tabla A.27. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 1.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	6/83	6/83
Total	Error medio	23.64 ± 9.34	23.64 ± 9.34
	Correlación en eje x	0.9995	0.9995
	Correlación en eje y	0.9992	0.9992

Tabla A.28. Comparativa entre los algoritmos MODET y MODET Mejorado. Cámara 2.

		MODET	MODET Mejorado
	Número de frames en los que no se detecta el objeto	78/126	56/126
Total	Error medio	12.17 ± 5.50	23.01 ± 17.07
	Correlación en eje x	0.9989	0.9942
	Correlación en eje y	0.9993	0.9994

A.4 Escenario exterior, cámara móvil y un objeto

A continuación recogemos los resultados de las pruebas realizadas en vídeos con un objeto en movimiento grabados con una cámara móvil en un escenario exterior.

Secuencia 2

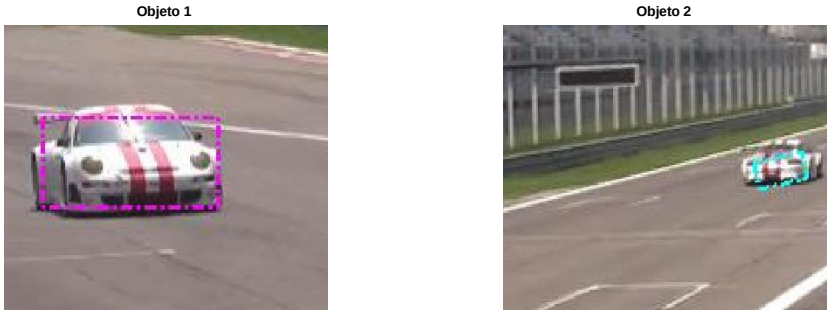

Figura A.79. Objetos que intervienen en la secuencia.

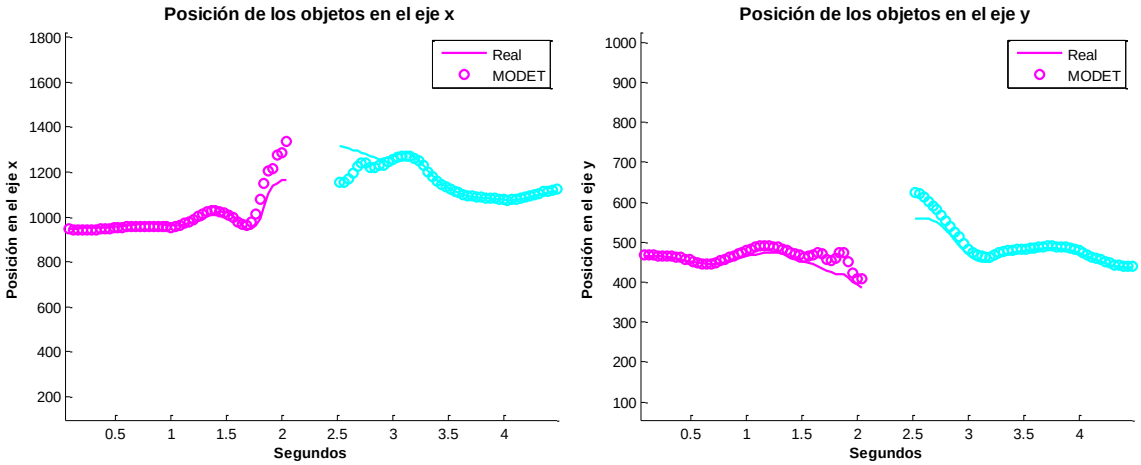

Figura A.80. Posición real de los objetos frente a posición MODET.

Tabla A.29. Resultados del algoritmo MODET.

		MODET
Total	Número de frames en los que no se detecta el objeto	0/100
	Error medio	28.12 ± 39.98
	Correlación en eje x	0.9266
	Correlación en eje y	0.9408

Secuencia 3

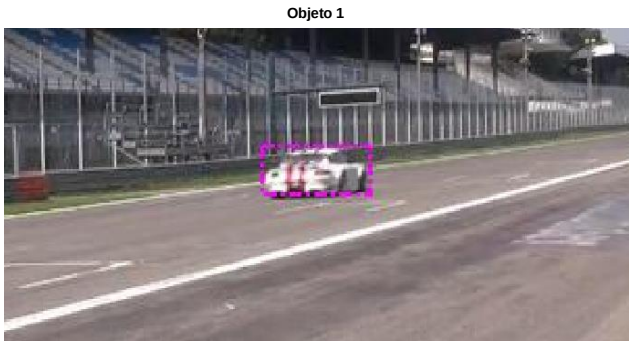

Figura A.81. Objeto que interviene en la secuencia.

Figura A.82. Trayectoria del objeto a lo largo de la secuencia.

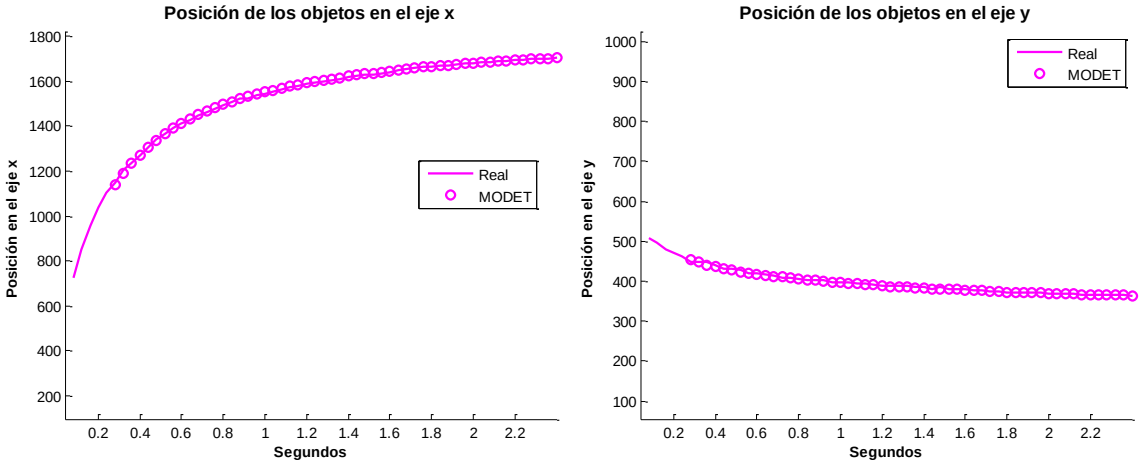

Figura A.83. Posición real de los objetos frente a posición MODET.

Tabla A.30. Resultados del algoritmo MODET.

		MODET
Total	Número de frames en los que no se detecta el objeto	5/59
	Error medio	3.37 ± 3.64
	Correlación en eje x	0.9996
	Correlación en eje y	0.9967

MANUAL DEL PROGRAMADOR

En este apéndice se incluyen una parte de los scripts implementados en MATLAB que intervienen en **MODET**.

A continuación mostramos un script ejemplo de la ejecución de **MODET** para una secuencia con grabada con dos vídeos.

```
clear all
close all

SURFThreshold = 100;
Models = 1;
dist_thresh = 1;
kCluster = 40;
numMinPoints = 6;
minScore = -2;
numMaxPointsTracking = 200;
disObj    = 250;

camara1 = 'videosEntrada\view1MultObj1.avi';
camara2 = 'videosEntrada\view2MultObj1.avi';
videoSalida = 'videosSalida\MultObj1.avi';
[pointDemotMultObjDos1_1, pointDemotMultObjDos2_1,...
pointDemotMultObj1_1, pointDemotMultObj2_1] =
MODET(camara1,camara2,videoSalida,disObj,numMinPoints,kCluster,dist_th
resh,Models,SURFThreshold,minScore,numMaxPointsTracking);
```

El fichero *MODET.m* implementa la mejora propuesta y genera el vídeo con los resultados.

```
function [pointAlgoView1, pointAlgoView2, pointAlgoView1_one,...
          pointAlgoView2_one] =
MODET(camara1,camara2,videoSalida,disObj,numMinPoints,kCluster,dist_th
resh,ModelsDetect,SURFThreshold,minScore,numMaxPointsTracking)

write_movie     = 1;
numObjetos      = 1;
start_frame     = 1;
start_frame2    = 1;
moviereader     = VideoReader(camara1);
moviereader2    = VideoReader(camara2);
endframe    = get(moviereader,'NumberOfFrames');
endframe2   = get(moviereader2,'NumberOfFrames');

[boxTotal, nobjectsTotal, numObjectoTotal, numObjetos] =
detectionTracking
(moviereader,numObjetos,start_frame,endframe,numMinPoints,kCluster,dis
t_thresh,ModelsDetect,SURFThreshold,minScore,numMaxPointsTracking);
[boxTotal2, nobjectsTotal2, numObjectoTotal2, numObjetos2] =
detectionTracking
(moviereader2,numObjetos,start_frame2,endframe2,numMinPoints,kCluster,
dist_thresh,ModelsDetect,SURFThreshold,minScore,numMaxPointsTracking);

pointAlgoView1_one      = cell(endframe,1);
pointAlgoView2_one      = cell(endframe2,1);
```

```
pointAlgoView1      = cell(endframe,1);
pointAlgoView2      = cell(endframe2,1);
hist        = zeros(1,2);
color        = cell(numObjetos2,1);

FRate1 = moviereader.FrameRate;
FRate2 = moviereader2.FrameRate;

instantTime = 1/FRate1;
time1 = FRate1 * instantTime;
time2 = FRate2 * instantTime;

corresObjecto = [];
corresObjecto2 = [];


% Homografía
[points1, points2] =
getpoints(read(moviereader,1),read(moviereader2,1));
Hom = homography2d(points1, points2);
HomInv = homography2d(points2, points1);

% Inicializamos la grabación
if write_movie
    try
        writerObj  = VideoWriter(videoSalida);
        writerObj.Quality = 75;
        writerObj.FrameRate = 15;
        open(writerObj);
    catch avierror
        disp(avierror)
        close(writerObj);
    end
end

figurewin = figure(1);
set(figurewin,'Position',[50 100 1300 400]);

for i = 1 + start_frame : min(floor((endframe - 1)*time2), (endframe2
- 1))

    MovieFrame = read(moviereader,ceil(i/time2));
    MovieFrame2 = read(moviereader2,i);

    subX1 = 1;
    subX2 = size(MovieFrame,2);
    subY1 = 1;
    subY2 = size(MovieFrame,1);

    cuadX1 = subX1 + subX2 * 0.05;
    cuadX2 = subX2 - subX2 * 0.05;
    cuadY1 = subY1 + subY2 * 0.05;
    cuadY2 = subY2 - subY2 * 0.05;

    if i==start_frame+1
        subplot(1,2,1)
        imshow(MovieFrame(cuadY1:cuadY2,cuadX1:cuadX2,:));
        subplot(1,2,2)
        imshow(MovieFrame2(cuadY1:cuadY2,cuadX1:cuadX2,:));
```

```
    else
        subplot(1,2,1)
        imshow(MovieFrame(cuadY1:cuadY2,cuadX1:cuadX2,:));
        subplot(1,2,2)
        imshow(MovieFrame2(cuadY1:cuadY2,cuadX1:cuadX2,:));
    end

    nobjects = nobjectsTotal{ceil(i/time2)};
    box = boxTotal{ceil(i/time2)};
    nobjects2 = nobjectsTotal2{i};
    box2 = boxTotal2{i};
    numObjecto = numObjectoTotal{ceil(i/time2)};
    numObjecto2 = numObjectoTotal2{i};

    %%Asignar Objetos

    if (nobjects || nobjects2)

        [numObj, numObj2] = asignaObj(box, box2, Hom, nobjects,
nobjects2, disObj);

        for j = 1:nobjects

            [posicion] = find(hist(:,1)==numObjecto{j});

            if isempty(posicion)%Si es nuevo

                inbox  = (j);
                inbox2 = find(numObj2==numObj(j));

                if numObj(j) > 0 && ~isempty(inbox2) &&
~isempty(find(hist(:,2)==numObjecto2{inbox2}, 1))
                    %Si tiene pareja asigno el color de la pareja
                    [posicion] = find(hist(:,2)==numObjecto2{inbox2});
                    hist(posicion,1) = [numObjecto{j}];
                    color{numObjecto{j}} = posicion;
                else
                    %Si la pareja no existe o no tinee correspondencia
                    hist(end+1,1) = [numObjecto{j}];
                    [posicion] = find(hist(:,1)==numObjecto{j});
                    color{numObjecto{j}} = posicion;
                end

            else%Si ya existe elijo color
                color{numObjecto{j}} = posicion;
            end

        end


        for j = 1:nobjects2

            [posicion] = find(hist(:,2)==numObjecto2{j});

            if isempty(posicion)

                inbox  = (j);
                inbox2 = find(numObj==numObj2(j), 1);
```

```
                if numObj2(j) > 0 && ~isempty(inbox2) &&
~isempty(find(hist(:,1)==numObjecto{inbox2}))

                    [posicion] = find(hist(:,1)==numObjecto{inbox2})
                    hist(posicion,2) = [numObjecto2{j}];
                    color{numObjecto2{j}} = posicion;

                else

                    hist(end+1,2) = [numObjecto2{j}];
                    [posicion] = find(hist(:,2)==numObjecto2{j});
                    color{numObjecto2{j}} = posicion;

                end

            else

                color{numObjecto2{j}} = posicion;

            end

        end

    end

    nobjetosTemp = nobjects;
    nobjetosTemp2 = nobjects2;

    %%Si los objetos que no correspondan busco su imagen en la otra
vista
    for j = 1:nobjects

        inbox  = (j);
        inbox2 = find(numObj2==numObj(j), 1);

        if numObj(j) == 0 || isempty(inbox2)

            boxTemp = Hom * box{j};

            mbox = zeros(2,size(boxTemp,2));
            mbox(1,:) = boxTemp(1,:) ./ boxTemp(3,:);
            mbox(2,:) = boxTemp(2,:) ./ boxTemp(3,:);
            meanbox = [ mean(mbox(1,:)); mean(mbox(2,:))];

            if (cuadX1 < meanbox(1) &&...
                meanbox(1) < cuadX2 &&...
                cuadY1 < meanbox(2) &&...
                meanbox(2) < cuadY2)

                box2{end+1} = boxTemp;
                nobjetosTemp2 = nobjetosTemp2 +1;
                numObjetos2 = numObjetos2 + 1;
                numObjecto2{end+1} = numObjetos2;
                color{numObjecto2{end}} = color{numObjecto{j}};

            end
```

```
        end

    end


    for j = 1:nobjects2

        inbox  = (j);
        inbox2 = find(numObj==numObj2(j), 1);

        if numObj2(j) == 0 || isempty(inbox2)

            boxTemp = HomInv * box2{j};

            mbox = zeros(2,size(boxTemp,2));
            mbox(1,:) = boxTemp(1,:) ./ boxTemp(3,:);
            mbox(2,:) = boxTemp(2,:) ./ boxTemp(3,:);
            meanbox = [ mean(mbox(1,:)); mean(mbox(2,:))];

            if (cuadX1 < meanbox(1) &&...
                meanbox(1) < cuadX2 &&...
                cuadY1 < meanbox(2) &&...
                meanbox(2) < cuadY2)

                box{end+1} = HomInv * box2{j};
                nobjetosTemp = nobjetosTemp +1;
                numObjetos2 = numObjetos2 + 1;
                numObjecto{end+1} = numObjetos2;
                color{numObjecto{end}} = color{numObjecto2{j}};

            end

        end
    end

    if nobjetosTemp

        subplot(1,2,1)
        hold on
        plottingTotal(MovieFrame, nobjetosTemp, box, color,
numObjecto);
        hold off;

        for obj = 1:nobjects
            plotbox = zeros(2,size(box{obj},2));
            plotbox(1,:) = box{obj}(1,:) ./ box{obj}(3,:);
            plotbox(2,:) = box{obj}(2,:) ./ box{obj}(3,:);

pointAlgoView1_one{ceil(i/time2)}(:,color{numObjecto{obj}}) = [
mean(plotbox(1,:)); mean(plotbox(2,:))];
        end

        for obj = 1:nobjetosTemp
            plotbox = zeros(2,size(box{obj},2));
            plotbox(1,:) = box{obj}(1,:) ./ box{obj}(3,:);
            plotbox(2,:) = box{obj}(2,:) ./ box{obj}(3,:);
            pointAlgoView1{ceil(i/time2)}(:,color{numObjecto{obj}}) =
[ mean(plotbox(1,:)); mean(plotbox(2,:))];
```

```
        end


    end

    if  nobjetosTemp2

        subplot(1,2,2)
        hold on
        plottingTotal(MovieFrame2, nobjetosTemp2, box2, color,
numObjecto2);
        hold off;

        for obj = 1:nobjects2
            plotbox = zeros(2,size(box2{obj},2));
            plotbox(1,:) = box2{obj}(1,:) ./ box2{obj}(3,:);
            plotbox(2,:) = box2{obj}(2,:) ./ box2{obj}(3,:);
            pointAlgoView2_one{i}(:,color{numObjecto2{obj}}) =
[mean(plotbox(1,:)); mean(plotbox(2,:))];
        end

        for obj = 1:nobjetosTemp2
            plotbox = zeros(2,size(box2{obj},2));
            plotbox(1,:) = box2{obj}(1,:) ./ box2{obj}(3,:);
            plotbox(2,:) = box2{obj}(2,:) ./ box2{obj}(3,:);
            pointAlgoView2{i}(:,color{numObjecto2{obj}}) =
[mean(plotbox(1,:)); mean(plotbox(2,:))];
        end

    end

    if write_movie
        frame = getframe(gcf);
        writeVideo(writerObj,frame);
    end

end

if write_movie
    close(writerObj)
end
```

El fichero *detectionTracking.m* se encarga llamar a los módulos de detección y seguimiento de objetos iterativamente.

```
function [boxTotal, nobjectsTotal, numObjectoTotal, numObjetos,
frametimeDetector, frametimeTracking] = detectionTracking(moviereader,
numObjetos, start_frame,
endframe,numMinPoints,kCluster,dist_thresh,ModelsDetect,SURFThreshold,
minScore,obj_points_to_use)

AllModels = [1 2];

%%Inicializo variables
nobjects = 0;

%%%
```

```
% Configuro el conjunto de variables de cada objeto
%%%
CurrObjDescr  = cell(nobjects,1);
CurrObjFrames = cell(nobjects,1);
CurrObjScore  = cell(nobjects,1);
Models        = cell(nobjects,1);
box           = cell(nobjects,1);
prevbox       = cell(nobjects,1);
cost          = cell(nobjects,1);
ntomatch      = cell(nobjects,1);
etiqueta      = cell(nobjects,1);
boxTotal      = cell(nobjects,1);
nobjectsTotal = cell(nobjects,1);
numObjectoTotal = cell(nobjects,1);
instante  = cell(nobjects,1);

%Vector que almacena el tiempo de cada iteración
frametimeDetector = zeros(endframe - 1,1);
frametimeTracking = zeros(endframe - 1,1);

%
%  START TRACKING
%
%%Detectamos y describimos los puntos característicos del frame i-1
MovieFrame = read(moviereader,start_frame);
if size(MovieFrame,3)>1
    MovieFrameGray = rgb2gray(MovieFrame);
end
points_temp =
detectSURFFeatures(MovieFrameGray,'MetricThreshold',SURFThreshold);
[features, valid_points] = extractFeatures(MovieFrameGray,
points_temp, 'Method', 'SURF');
descrPrev = features';
pointsPrev = valid_points.Location';


try
for i = 1 + start_frame:endframe

    %Obtener frame actual
    MovieFrame = read(moviereader,i);
    %Convertir a escala de grises
    if size(MovieFrame,3)>1
        MovieFrameGray = rgb2gray(MovieFrame);
    end

    points_temp =
detectSURFFeatures(MovieFrameGray,'MetricThreshold',SURFThreshold);
    [features, valid_points] = extractFeatures(MovieFrameGray,
points_temp, 'Method', 'SURF');
    descrActual = features';
    pointsActual = valid_points.Location';


    %Inicialización de variables
    nobjectsTotal{i} = 0;
    boxTotal{i} = [];
    numObjectoTotal{i} = [];
```

```
    if nobjects > 0

        tTracking = tic;
        [ntomatch, inliers, CurrObjScore, CurrObjFrames, CurrObjDescr,
cost,...
         matches, objinliers, objoutliers, box, prob, nobjects,
prevbox, etiqueta] = ...
         tracking(MovieFrame, box, nobjects, CurrObjScore,
CurrObjFrames,...
                  CurrObjDescr, obj_points_to_use, ntomatch, cost,...
                  Models, prevbox, etiqueta, pointsActual,
descrActual, i, instante, minScore);


        [CurrObjScore, CurrObjFrames, CurrObjDescr] = ...
         update(nobjects, CurrObjScore, objinliers, objoutliers, box,
pointsActual,...
                inliers, prob, matches, CurrObjFrames, CurrObjDescr,
ntomatch, descrActual);

        boxTotal{i} = box(1:nobjects);
        nobjectsTotal{i} = nobjects;
        numObjectoTotal{i} = etiqueta(1:nobjects);

        frametimeTracking(i) = toc(tTracking);
    end

    %%Fase de detección
    %%Almacen el frame actual y el siguiente
    tDetector = tic;

    %% Primera Cámara

    [orgboxNuevos] =
detection(MovieFrameGray,descrPrev,pointsPrev,descrActual,pointsActual
,numMinPoints,kCluster,dist_thresh,ModelsDetect);
    nobjectsNuevos = length(orgboxNuevos);

    %%Para cada objeto detectado comprobamos que tenga área de
solapamiento
    %%con otro objeto

    for n = 1:nobjectsNuevos

        subX1Nuevos = min(orgboxNuevos {n}(1,:));
        subX2Nuevos = max(orgboxNuevos {n}(1,:));
        subY1Nuevos = min(orgboxNuevos {n}(2,:));
        subY2Nuevos = max(orgboxNuevos {n}(2,:));

        %%Inicializo matriz de solapamiento
        areaSolapamiento = [];

        for m = 1:nobjects

            subX1 = min(box{m}(1,:));
            subX2 = max(box{m}(1,:));
            subY1 = min(box{m}(2,:));
            subY2 = max(box{m}(2,:));
```

```
            a = [subX1Nuevos, subY1Nuevos, (subX2Nuevos -
subX1Nuevos), ...
                (subY2Nuevos - subY1Nuevos)];

            b = [subX1, subY1, (subX2 - subX1), (subY2 - subY1)];
            areaSolapamiento(m)= rectint(a,b);

        end

        if (~exist('areaSolapamiento', 'var') ||
(sum(areaSolapamiento(:)) == 0))

            clear ObjDescr;
            clear ObjFrames;
            %%Añado el objeto
            nobjects = nobjects + 1;

            orgbox{nobjects} = orgboxNuevos{n};
            inside{nobjects} = find(inpolygon(pointsActual(1,:),
pointsActual(2,:), orgbox{nobjects}(1,:), orgbox{nobjects}(2,:)));

            to_add_points = inside{nobjects};

            ObjFrames = pointsActual(:,to_add_points);
            ObjDescr  = descrActual(:,to_add_points);

            %Create the object feature list
            CurrObjDescr{nobjects}  = ObjDescr;
            CurrObjScore{nobjects}  = zeros(1,size(ObjDescr,2));
            CurrObjFrames{nobjects} = [ObjFrames(1:2,:);
ones(1,size(ObjFrames,2))];
            ntomatch{nobjects} =
min(obj_points_to_use,length(CurrObjScore{nobjects}));

            Models{nobjects} = AllModels;

            box{nobjects} = [orgbox{nobjects} ;
ones(1,size(orgbox{nobjects},2))];
            prevbox{nobjects} = box{nobjects};

            %Start with a cost lambda of 10% of box
            boxmaxx = 0.1*(max(orgbox{nobjects}(1,:)) -
min(orgbox{nobjects}(1,:)));
            boxmaxy = 0.1*(max(orgbox{nobjects}(2,:)) -
min(orgbox{nobjects}(2,:)));
            M = eye(2);
            t = [boxmaxx; boxmaxy];
            H = [M t;0 0 1];

            cost{nobjects} =
projective_cost_function(H,box{nobjects});

            etiqueta{nobjects} = numObjetos;%%Número del objeto
            numObjetos = numObjetos + 1;

            boxTotal{i} = box(1:nobjects);
            nobjectsTotal{i} = nobjects;
```

```
            numObjectoTotal{i} = etiqueta(1:nobjects);
            instante{nobjects} = i;

        end
    end

    descrPrev = descrActual;
    pointsPrev = pointsActual;

    frametimeDetector(i) = toc(tDetector);
end
catch errormessage

    disp('ERROR found.');
    disp(errormessage);
    for m = 1:length(errormessage.stack)
        disp(errormessage.stack(m));
    end
end
```

El fichero *detection.m* forma objetos conectando puntos de interés en movimiento.

```
function [orgboxNuevos] =
detection(MovieFrameGray,descrPrev,pointsPrev,descrActual,pointsActual
,numMinPoints,kCluster,dist_thresh,Models)

    [correspondencias]= surfmatchFix(descrPrev,descrActual);
    MatchedPointsPrev = pointsPrev(:,correspondencias(1,:));
    MatchedPointsPrev = [MatchedPointsPrev;
ones(1,size(MatchedPointsPrev,2))];
    MatchedPointsActual = pointsActual(:,correspondencias(2,:));
    MatchedPointsActual = [MatchedPointsActual;
ones(1,size(MatchedPointsActual,2))];

    subX1 = 1;
    subX2 = size(MovieFrameGray,2);
    subY1 = 1;
    subY2 = size(MovieFrameGray,1);
    cuadX1 = subX1 + subX2 * 0.05;
    cuadX2 = subX2 - subX2 * 0.05;
    cuadY1 = subY1 + subY2 * 0.05;
    cuadY2 = subY2 - subY2 * 0.05;
    box = [subX1 subX1 subX2 subX2 subX1;subY2 subY1 subY1 subY2
subY2; ones(1,5)];

    lambda = 0;
    [~, inliers] = RAMOSAC(MatchedPointsPrev, MatchedPointsActual, ...
                           box, lambda, Models, dist_thresh);
    frametime(2,1) = toc;

    outliers = setdiff(1:size(MatchedPointsActual,2),inliers);
    clear OutliersPointsActual;

    OutliersPointsActual = MatchedPointsActual(:,outliers);

    orgboxNuevos = cell(0,1);
```

```
    if (exist('OutliersPointsActual', 'var'))

        cols = OutliersPointsActual(1,:)';
        rows = OutliersPointsActual(2,:)';

        Y = pdist([rows,cols],'cityblock');

        if ~isempty(Y)

            Z = linkage(Y,'single');
            clear Y;
            grupos =
cluster(Z,'cutoff',kCluster,'criterion','distance');
            numGrupos=max(grupos);

            numFinales = 0;
            indiceTemp = [];

            for i=1:numGrupos

                indiceGrupo=find(grupos==i);

                if size(indiceGrupo,1) > numMinPoints ...
                    && max(cols(indiceGrupo))< cuadX2 &&
min(cols(indiceGrupo)) > cuadX1 ...
                    && max(rows(indiceGrupo))< cuadY2 &&
min(rows(indiceGrupo)) > cuadY1

                    numFinales = numFinales + 1; %%Aumentamos el
número de objetos
                    indiceTemp(numFinales) = i;

                end

            end

            orgboxNuevos = cell(numFinales,1);

            for i=1:numFinales

                indiceFinal=find(grupos==indiceTemp(i));
                orgboxNuevos{i} = [min(cols(indiceFinal))
min(cols(indiceFinal))...
                                   max(cols(indiceFinal))
max(cols(indiceFinal));...
                                   min(rows(indiceFinal))
max(rows(indiceFinal))...
                                   max(rows(indiceFinal))
min(rows(indiceFinal))];

            end
        end
    end
```

El fichero *tracking.m* se encarga de obtener la transformación, al *frame* actual, de objetos almacenados.

```
function [ntomatch, inliers, CurrObjScore, CurrObjFrames,
CurrObjDescr, cost,...
          matches, objinliers, objoutliers, box, prob, nobjects,
prevbox, numColor] = ...
          tracking(MovieFrame, box, nobjects, CurrObjScore,
CurrObjFrames,...
                   CurrObjDescr, obj_points_to_use, ntomatch, cost,...
                   Models, prevbox, numColor, frames, descr, i,
instante, minScore)

    dist_thresh = 3;
    matches      = cell(nobjects,1);
    objinliers   = cell(nobjects,1);
    objoutliers  = cell(nobjects,1);
    frameinliers = cell(nobjects,1);
    lambda       = cell(nobjects,1);

    H       = cell(nobjects,1);
    inliers = cell(nobjects,1);
    prob    = cell(nobjects,1);

    obj = 1;

    subX1 = 1;
    subX2 = size(MovieFrame,2);
    subY1 = 1;
    subY2 = size(MovieFrame,1);

    cuadX1 = subX1 + subX2 * 0.05;
    cuadX2 = subX2 - subX2 * 0.05;
    cuadY1 = subY1 + subY2 * 0.05;
    cuadY2 = subY2 - subY2 * 0.05;

    while obj <= nobjects

        %%Extraemos los descriptores del objeto almacenado hasta un
máximo de obj_points_to_use
        [CurrObjScore{obj}, scoreindex] =
sort(CurrObjScore{obj},'descend');
        CurrObjFrames{obj} = CurrObjFrames{obj}(:,scoreindex);
        CurrObjDescr{obj}  = CurrObjDescr{obj}(:,scoreindex);

        ntomatch{obj} =
min(obj_points_to_use,length(CurrObjScore{obj}));

        %Buscamos correspondencias entre los puntos almacenados y los
        %puntos del frame actual
        [matches{obj}]=
surfmatchFix(CurrObjDescr{obj}(:,1:ntomatch{obj}),descr);

        %Estimate lambda for previous information
        lambda{obj} = 1/mean(cost{obj});
        clear MatchedPoints;
        clear MatchedObjPoints;
```

```
        MatchedObjPoints = CurrObjFrames{obj}(1:3,matches{obj}(1,:));
        MatchedPoints    = frames(:,matches{obj}(2,:));
        MatchedPoints    = [MatchedPoints;
ones(1,size(MatchedPoints,2))];

        %%% Buscamos la transformación de movimiento del objeto
        [H{obj}, inliers{obj}, ~,s] = RAMOSAC(MatchedObjPoints,...
                                        MatchedPoints,   ...
                                        box{obj},        ...
                                        lambda{obj},     ...
                                        Models{obj},     ...
                                        dist_thresh);

        outliers{obj} =
setdiff(1:size(MatchedObjPoints,2),inliers{obj});

        objinliers{obj}   = matches{obj}(1,inliers{obj});
        objoutliers{obj}  = matches{obj}(1,outliers{obj});
        frameinliers{obj} = matches{obj}(2,inliers{obj});

        %Almacenamos la última posición del objeto
        prevbox{obj} = box{obj};
        %Transformamos el box del objeto
        box{obj} = H{obj}*box{obj};

        box{obj}(1,:) = box{obj}(1,:)./box{obj}(3,:);
        box{obj}(2,:) = box{obj}(2,:)./box{obj}(3,:);
        box{obj}(3,:) = box{obj}(3,:)./box{obj}(3,:);

        mbox = zeros(2,size(box{obj},2));
        mbox(1,:) = box{obj}(1,:) ./ box{obj}(3,:);
        mbox(2,:) = box{obj}(2,:) ./ box{obj}(3,:);
        meanbox = [ mean(mbox(1,:)); mean(mbox(2,:))];

        %Transformamos los puntos almacenados del objeto
        CurrObjFrames{obj}(1:3,:) = H{obj}*CurrObjFrames{obj}(1:3,:);

        %Calculamos el coste y probabilidad de la estimación de
transformación
        cost{obj}(i - instante{obj} + 1) =
projective_cost_function(H{obj},prevbox{obj});
        prob{obj} = exp(-lambda{obj}*cost{obj}(i - instante{obj} +
1));

        %Comprobamos si el centro del objeto está fuera del área de
detección,
        %de ser así se elimina.
        if (cuadX1 > meanbox(1) ||...
            meanbox(1) > cuadX2 ||...
            cuadY1 > meanbox(2) ||...
            meanbox(2) > cuadY2 ||...
            s < minScore)

            clear counter
            for counter=obj+1:nobjects

                ntomatch{counter-1}=ntomatch{counter};
                inliers{counter-1}=inliers{counter};
                CurrObjScore{counter-1}=CurrObjScore{counter};
```

```
                CurrObjFrames{counter-1}=CurrObjFrames{counter};
                CurrObjDescr{counter-1}=CurrObjDescr{counter};
                matches{counter-1}=matches{counter};
                objinliers{counter-1}=objinliers{counter};
                objoutliers{counter-1}=objoutliers{counter};
                frameinliers{counter-1}=frameinliers{counter};
                box{counter-1}=box{counter};
                prob{counter-1}=prob{counter};
                numColor{counter-1}=numColor{counter};

            end

            ntomatch{end}=[];
            inliers{end}=[];
            CurrObjScore{end}=[];
            CurrObjFrames{end}=[];
            CurrObjDescr{end}=[];
            matches{end}=[];
            objinliers{end}=[];
            objoutliers{end}=[];
            frameinliers{end}=[];
            box{end}=[];
            prob{end}=[];
            numColor{end}=[];
            nobjects = nobjects-1;
            obj = obj-1;

        end
        obj = obj + 1;

    end
```

El fichero *update.m*, partiendo de la implementación de Peter Strandmark [30], actualiza los objetos almacenados con nuevos puntos de interés.

```
function [CurrObjScore, CurrObjFrames, CurrObjDescr] = ...
          update(nobjects, CurrObjScore, objinliers, objoutliers, box,
frames, matches, CurrObjFrames, CurrObjDescr, ntomatch, descr)

    %%%Actualizamos los objetos almacenados
    points_to_add      = cell(nobjects,1);
    exterior           = cell(nobjects,1);

    for obj = 1:nobjects
        %Actualizamos las puntuaciones de cada objeto
        CurrObjScore{obj}(objinliers{obj})  =
CurrObjScore{obj}(objinliers{obj})  + 2;
        CurrObjScore{obj}(objoutliers{obj}) =
CurrObjScore{obj}(objoutliers{obj}) - 1;

        %Lista de puntos a actualizar
        points_to_add{obj}      = [];

        plotbox = zeros(2,size(box{obj},2));
        plotbox(1,:) = box{obj}(1,:) ./ box{obj}(3,:);
        plotbox(2,:) = box{obj}(2,:) ./ box{obj}(3,:);
```

```
        boxcenter = mean(plotbox,2);
        exteriorbox = plotbox - 0.1*(boxcenter*ones(1,size(plotbox,2))
- plotbox);

        exact{obj}    = find(inpolygon(frames(1,:), frames(2,:),
plotbox(1,:), plotbox(2,:)));
        exterior{obj} = find(inpolygon(frames(1,:), frames(2,:),
exteriorbox(1,:), exteriorbox(2,:)));
    end

    for obj = 1:nobjects

            points_to_add{obj} = exact{obj};

            for obj2 = setdiff(1:nobjects,obj)

            overlap = intersect(exact{obj}, exact{obj2});
                if ~isempty(overlap)

                    nmatch1 = length(intersect(overlap,
matches{obj}(2,:)));
                    nmatch2 = length(intersect(overlap,
matches{obj2}(2,:)));

                    if nmatch1 <= nmatch2
                        points_to_add{obj} =
setdiff(points_to_add{obj}, exterior{obj2});
                    end
                end
            end

            CurrObjFrames{obj} = [CurrObjFrames{obj}
[frames(:,points_to_add{obj}); ones(1,length(points_to_add{obj}))]];
            CurrObjDescr{obj}  = [CurrObjDescr{obj}
descr(:,points_to_add{obj})];
            initial_score =
median(CurrObjScore{obj}(1:ntomatch{obj}));
            CurrObjScore{obj}  = [CurrObjScore{obj}
initial_score*ones(1,length(points_to_add{obj}))];

    end
```

El fichero *surfmatch.m* busca correspondencias entre dos conjuntos de descriptores.

```
function [matches] = surfmatch(descr1,descr2)

        [indexPairs,matchmetric]  =
matchFeatures(descr1',descr2','MatchThreshold',10);
        matches = indexPairs';
        I = matchmetric';
        [b1, m1, n1] = unique(matches(2,:),'first');
        [b2, m2, n2] = unique(matches(2,:),'last');
        nunique = length(b1);
        matchesBuenos = zeros(2,nunique);
        ix = find(m1 == m2);
        matchesBuenos(:,ix) = matches(:,m1(ix));
        ix = find(m1 ~= m2);
```

```
        nix = length(ix);

        for i = 1:nix

            irepe = b1(ix(i));
            jx = find(matches(2,:) == irepe);
            [~, maxjx] = min(I(jx));
            matchesBuenos(:,ix(i)) = matches(:,jx(maxjx));
        end

        matches = matchesBuenos;
```

El fichero *asignaObj.m* empareja objetos situados en el mismo lugar de la escena, vistos desde distintas cámaras.

```
function [numObj, numObj2] = asignaObj(orgbox, orgbox2, Hom, nobjects,
nobjects2, disObj)

numObj    = zeros(1,nobjects);
numObj2   = zeros(1,nobjects2);
orgboxpru = cell(nobjects,1);
dis       = 0;
matches   = 0;
maxDis    = 1000;

for i = 1 : nobjects

    orgboxpru{i} = Hom * [mean(orgbox{i},2)];
    orgboxpru{i}(1,:) = orgboxpru{i}(1,:) ./ orgboxpru{i}(3,:);
    orgboxpru{i}(2,:) = orgboxpru{i}(2,:) ./ orgboxpru{i}(3,:);
    orgboxpru{i} = [orgboxpru{i}(1:2,:); 1];

end

for j = 1 : nobjects
    for i = 1 : nobjects2

        dist = orgboxpru{j} - mean(orgbox2{i},2);
        dis(j,i) = sqrt(dist(1)^2 + dist(2)^2);

    end
end

while matches < min(nobjects, nobjects2)

    [Y,Ij] = min(dis,[],1);
    for i = 1:min(nobjects, nobjects2)

        X(i,:) = dis(i,:) - Y(1,:);

    end

    if min(Y) < disObj

        if nobjects < nobjects2 %%Cuando el número de objetos no es
concordante
```

```
            [Y,Ii] =  min(Y,[],2);
        else
            [Y,Ii] = max (sum (X));
        end

        numObj(1,Ij(Ii)) = Ij(Ii);
        numObj2(1,Ii) = Ij(Ii);
        dis (Ij(Ii), :) = [maxDis];
        dis (:, Ii) = [maxDis];

    end
    matches = matches+1;

end
```

El fichero *getPoints.m* selecciona puntos fijos del escenario, desde cada cámara.

```
function [points1 points2] = getPoints(camara1,camara2)

figure(1)
imshow(camara1);
hold on;
orgbox = zeros(2,0);
while 1
    [x y button] = ginput(1);
    if button ~= 1
        if size(orgbox,2) < 4
            warning('Necesitas al menos 4 puntos...');
        else
            break
        end

    else
        orgbox(:,end+1) = [x;y;];
    end

    plot(orgbox(1,:),orgbox(2,:),'.r','LineWidth',2);
    for obj = 1:size(orgbox,2)

text(orgbox(1,obj),orgbox(2,obj),['\fontsize{14}\bf\leftarrow',num2str
(obj)]);
    end
end

points1 = orgbox;

figure(2)
imshow(camara2);
hold on;
orgbox = zeros(2,0);
while 1
    [x y button] = ginput(1);
    if button ~= 1
        if size(orgbox,2) < 4
            warning('Necesitas al menos 4 puntos...');
        else
            break
        end
```

```
    elseif isempty(x)

        if size(orgbox,2) < 4
            warning('Necesitas al menos 4 puntos...');
        end
        orgbox = zeros(2,0);
    else
        orgbox(:,end+1) = [x;y;];
    end

    plot(orgbox(1,:),orgbox(2,:),'.r','LineWidth',2);
    for obj = 1:size(orgbox,2)

text(orgbox(1,obj),orgbox(2,obj),['\fontsize{14}\bf\leftarrow',num2str
(obj)]);
    end
end

points2 = orgbox;
points1(3,:) = 1;
points2(3,:) = 1;
```

El fichero *plottingTotal.m* representa los objetos detectados y seguidos por **MODET**.

```
function pointCentroide = plottingTotal(MovieFrame, nobjects, box,
color, numObjecto)

    subX1 = 1;
    subX2 = size(MovieFrame,2);
    subY1 = 1;
    subY2 = size(MovieFrame,1);

    cuadX1 = subX1 + subX2 * 0.05;
    cuadY1 = subY1 + subY2 * 0.05;
    pointCentroide = [];

    boxcolors = {[1 1 0],[1 0 1],[0 1 1],[1 0 0],[0 1 0],[0 0 1],...

[1,.4,.6],[1,.6,.4],[.6,1,.4],[.6,.4,1],[.4,1,.6],[.4,.6,1],...

[.4,.6,0],[.4,0,.6],[0,.4,.6],[1,.4,0],[1,0,.6],[0,1,.6]};

    for obj = 1:nobjects

        numColor = mod(color{numObjecto{obj}}, 19) + 1;
        boxcolor = boxcolors{numColor};

        plotbox = zeros(2,size(box{obj},2));
        plotbox(1,:) = box{obj}(1,:) ./ box{obj}(3,:);
        plotbox(2,:) = box{obj}(2,:) ./ box{obj}(3,:);
        plot(plotbox(1,[1:end 1]) - cuadX1,plotbox(2,[1:end 1]) -
cuadY1,'-.','Color',boxcolor,'LineWidth',2);

    end
```

PRESUPUESTO

1) Ejecución Material	
▪ Compra de ordenador personal (Software incluido)	1.200 €
▪ Compra de dos cámaras IP	680 €
▪ Material de oficina	50 €
▪ Total de ejecución material	1.930 €
2) Gastos generales	
▪ 16 % sobre Ejecución Material	308,8 €
3) Beneficio Industrial	
▪ 6 % sobre Ejecución Material	115,8 €
4) Honorarios Proyecto	
▪ 1000 horas a 15 € / hora	15.000 €
5) Material fungible	
▪ Gastos de impresión	150 €
▪ Encuadernación	50 €
6) Subtotal del presupuesto	
▪ Subtotal Presupuesto	17.554,6 €
7) I.V.A. aplicable	
▪ 21 % Subtotal Presupuesto	3.686,4 €
8) Total Presupuesto	
▪ Total Presupuesto	21.241 €

Madrid, Mayo de 2013

El Ingeniero Jefe de Proyecto

Fdo.: Carlos Legua Cruz

Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un *"Seguimiento automático de objetos en sistemas con múltiples cámaras"*. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales.

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.
2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.
3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.
4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.
5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.
6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.
7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.
9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.
10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.
11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.
12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.
13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.
14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.
15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.
16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.
17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.
18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.
19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.
21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.
22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.
23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares.

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.
2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.
3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.
4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.
5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.
7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.
8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.
9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.
10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.
11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.
12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.