ESTUDIO DE VIABILIDAD DE UN ALGORITMO DE SEGUIMIENTO BASADO EN EL APRENDIZAJE PREVIO DE TRAYECTORIAS "HABITUALES"

PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIONES

ESPECIALIDAD DE ELECTRÓNICA

Autor: Miguel Alberto Sanz Pardo

Director: José Elias Herrero Jaraba

Zaragoza - 15 de junio de 2011

Universidad de Zaragoza - Centro Politécnico Superior

Departamento de Electrónica y Comunicaciones

Universidad Zaragoza

ESTUDIO DE VIABILIDAD DE UN ALGORITMO DE SEGUIMIENTO BASADO EN EL APRENDIZAJE PREVIO DE TRAYECTORIAS "HABITUALES"

Proyecto Fin de Carrera

Titulación: Ingeniería de Telecomunicaciones

Autor: Miguel Alberto Sanz Pardo

Director: José Elias Herrero Jaraba

Zaragoza - 15 de junio de 2011

RESUMEN

En este proyecto de fin de carrera se pretende estudiar la viabilidad de un algoritmo de seguimiento basado en el aprendizaje previo de trayectorias “habituales”. Para ello nos apoyaremos en un trabajo anterior formado por distintos métodos de detección y seguimiento de objetos.

Las trayectorias “habituales” o “comunes” son aquellas trayectorias asociadas a los objetos en movimiento (en nuestro caso personas, pero se podría aplicar a otros objetos usando los modelos apropiados) que aparecen con más frecuencia durante un cierto intervalo temporal en la zona espacial en la que pretendemos usar nuestro algoritmo de seguimiento. Para la obtención de trayectorias nos apoyaremos, como ya hemos mencionado antes, en un sistema de detección y seguimiento que ya ha sido desarrollado préviamente. Posteriormente a esto nos encargaremos de obtener las trayectorias medias a partir del conjunto de trayectorias obtenidas inicialmente. A estas trayectorias medias es a lo que llamaremos trayectorias “habituales” o “comunes”.

Una vez calculadas, trataremos de realimentar a los algoritmos de seguimiento con dichas trayectorias con el fin de que mejore el funcionamiento de dichos algoritmos. Se pretenderá además que la introducción de esta nueva información afecte lo menos posible al sistema en cuanto a inestabilidad y practicidad, de forma que el sistema pueda ejecutarse en tiempo real o en su defecto cercano a ello.

En este trabajo en concreto se va a trabajar con las posiciones y las trayectorias de los jugadores de dos equipos que se están enfrentando en un partido en un campo de futbol. En nuestro caso disponemos de 3 cámaras, las cuáles se han usado para grabar el partido (una en la zona izquierda, otra en la zona central y otra en la zona derecha del campo). De esta manera estaremos trabajando en un entorno real; el cuál a pesar de parecer estar controlado, es más caótico de lo que se pueda pensar a priori, puesto que cada jugador tiende a moverse por diferentes zonas del campo. De esta manera, si somos capaces de mejorar el sistema de seguimiento en este entorno, en teoría podremos ponerlo en práctica también en otros ámbitos fuera de lo deportivo.

Palabras clave:

Visión por computador, detección, seguimiento, trayectorias, kalman, UKF, MHT, homografía, tracker, mutiobjeto.

AGRADECIMIENTOS

En primer lugar, muchas gracias a todos/as los que me habeis dado vuestro apoyo a lo largo de la carrera, en especial a: Fran, Jesús, Antonio e Isabel, ya que sin su ayuda posiblemente aún no habría empezado el P.F.C. También agradecer los buenos momentos pasados a lo largo de la carrera a: Diego, Jorge, Emilio, Diana, Julia, Manolo, Andrés, Rillo, David, Alberto, Alex, Jesús, Javi, Ángel, Loreto, Aitor, Héctor o Murfi entre otros tantos compañeros de la carrera.

También agradecer el apoyo de aquellas personas que han estado en mi vida y le han dado un cierto sentido, desde amigos de la infancia (Domingo, Dani, Asier y Victor, entre otros) hasta amigos más recientes (Carlos, Tirso, Musta, Fernando, Alfredo y Rafa, entre otros).

Por supuesto agradecer y dedicar este trabajo a toda mi familia: abuelos, tíos, primos, padre y en especial a mi madre que a veces parece sufrir más que yo con mi carrera y cuando la finalice estoy seguro de que se sentirá muy contenta.

Y ante todo gracias a mi director Elías Herrero, por permitirme hacer este interesante proyecto y aportar buenas ideas para su desarrollo; a los profesores David Buldain y Carlos Orrite por aportar ideas interesantes para éste; y a todos mis compañeros de laboratorio por hacer más amenos los días: Miguel, Mario, Jorge y Pedro; en especial a Jorge, ya que sin su ayuda este P.F.C. igual aún no estaría finalizado.

Y si por descuido se me ha olvidado mencionar a alguien en los agradecimientos, también gracias a todas esas personas por estar ahí.

Índice general

CAPÍTULO 1

INTRODUCCIÓN

La Visión artificial, también conocida como *Visión por Computador*, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen. Los objetivos más comunes de la visión artificial suelen ser:

- Detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).
- Registro de diferentes imágenes de una misma escena u objeto.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de dicha escena.
- Estimación de las posturas tridimensionales de humanos.

En la actualidad los sistemas de detección y seguimiento de objetos (ya sean personas, vehículos u otro tipo de figuras) mediante *Visión por Computador* son usados para distintos tipos de aplicaciones como pueden ser:

- Control de procesos industriales.
- Control de calidad industrial.
- Otras aplicaciones no industriales: control de tráfico, sistemas de videovigilancia, procesamiento de imágenes en el ámbito de la medicina, estudio de tácticas, estadísticas y/o eventos deportivos.

1.1. Descripción del proyecto

Este proyecto de fin de carrera ha sido llevado a cabo en el departamento de Ingeniería Electrónica y Comunicaciones – en concreto, en el campo de la Visión por Computador perteneciente al Área de Tecnología Electrónica dicho departamento – del Centro Politécnico Superior (C.P.S.) de la Universidad de Zaragoza. Dicho trabajo ha sido desarrollado bajo la dirección del Dr.Ing. D.José Elías Herrero Jaraba y ha sido apoyado en especial en un gran trabajo previo (Tesis doctoral [1]) desarrollado anteriormente por el Dr.Ing. D.Jorge Raúl Gomez.

En nuestro caso el proyecto va a estar desarrollado en el ámbito deportivo y va a partir de un sistema de detección y seguimiento; el cuál ha sido implementado con anterioridad. La ventaja de los métodos basados en *Visión por Computador* respecto a otros métodos en los que es necesario el uso de dispositivos electrónicos en los propios jugadores estriba en que es un método no intrusivo. Puesto que no requiere ningún tipo de autorización por parte los organismos controladores de las diferentes disciplinas deportivas, ni puede poner en peligro a los deportistas, se trata de un medio bastante acertado para ser usado en entornos deportivos; y lo que es más, puede ser extendido a otros escenarios.

Los algoritmos de detección y segmentación que han sido desarrollados con anterioridad no van a sufrir ningún tipo de modificación. Nos centraremos en el intento de aportar mejoras respecto a dos algoritmos de seguimiento los cuáles han sido desarrollados préviamente(MSUKF y MHT):

- MSUKF (Multiple Sensor Unscented Kalman Filter)
  Es una variación del filtro de Kalman, en la cuál en vez de seguir un modelo lineal se sigue un modelo no lineal determinado. Además se trabaja con más de un sensor a la vez (en nuestro caso usamos tres cámaras).
- MHT (Multiple Hypothesis Tracking)
  Es otra variación del filtro de Kalman, en la cuál se tienen en cuenta un número de hipótesis, indeterminado a priori, para cada objetivo a seguir (aunque limitado a un número máximo de hipótesis por objetivo) en cada instante temporal. En este caso también se trabaja con más de un sensor a la vez (al igual que con el algoritmo anterior también se trabaja con tres cámaras).

Para ello vamos a apoyarnos en el aprendizaje previo de trayectorias “habituales” o “comunes”. Como hemos mencionado en el resumen, las trayectorias “habituales” o “comunes” son las trayectorias asociadas a los objetos en movimiento(en nuestro caso personas, pero se podría aplicar a otros objetos usando los modelos adecuados), las cuáles aparecen con mayor frecuencia durante un determinado intervalo temporal en la zona espacial en la que queremos dar uso de nuestro algoritmo de seguimiento.

En primer lugar, mediante un sistema de detección y seguimiento como el que está implementado bajo el

entorno gráfico CVLAB (el cuál funciona bajo Visual C++), nos encargaremos de etiquetar a las diferentes personas de interés que aparecen en la zona de estudio durante el intervalo temporal que deseemos. Deberemos de ir supervisando el etiquetado frame por frame, puesto que este sistema de detección y seguimiento implementado no siempre funciona correctamente y debido a cruces entre objetos y a oclusiones puede haber instantes en que queden mal etiquetados. En nuestro problema vamos a trabajar en un entorno deportivo, en concreto trabajaremos con las posiciones y trayectorias de los jugadores de dos equipos que se están enfrentando en un partido en un campo de futbol. Tan solo nos interesarán los jugadores de campo, ni porteros, ni árbitros, ni el resto del personal que hay en los alrededores del terreno de juego(gente calentando en la banda, entrenador, etc . . .). De tal forma dispondremos de 20 *"trackers"* (estructura de datos que contiene la posición y el estado de cada objeto en movimiento) para cada frame.

Una vez dispongamos de los jugadores etiquetados para el intervalo temporal deseado, deberemos de procesar y obtener las trayectorias individuales de cada jugador. Para ello usaremos el software matemático Matlab. Dividiremos el campo en una serie de cuadrantes uniformes e iremos almacenando las trayectorias de cada jugador para cada zona de dicho campo.

Tras esto, a partir de las trayectorias individuales, nos encargaremos de agrupar las trayectorias más similares y de obtener las trayectorias medias que aparecen en cada cuadrante, las cuáles serán una aproximación de lo que hemos llamado trayectorias "habituales" o "comunes".

Finálmente, usaremos dichas trayectorias para realimentar al sistema de seguimiento mencionado anteriormente y así podremos ver si hay mejoras en cuanto al seguimiento. Para ello calcularemos cuál es el error del sistema de seguimiento con y sin el uso de trayectorias medias y compararemos los resultados.

1.2. Objetivos del proyecto

El alcance de este proyecto de fin de carrera es tan ambicioso que el objetivo principal establecido es el de estudiar la viabilidad de los algoritmos de seguimiento apoyados en el aprendizaje de trayectorias "habituales" o "comunes".

Los sistemas de detección y seguimiento se dividen en dos partes, en nuestro proyecto intentaremos aportar mejoras a la parte correspondiente al seguimiento, mientras que la correspondiente a la detección no la modificaremos. Se pretende obtener mejoras respecto a los algoritmos de seguimiento que han sido implementados préviamente; los cuáles no usan información de las trayectorias "habituales", no obstante hasta que no sean realizadas las pruebas necesarias no se puede asegurar que vayan a mejorar de una manera sustancial. De este modo intentaremos que se puedan obtener las posiciones y las trayectorias de los objetivos que están presentes en la escena en cada instante, de forma que estemos trabajando de

la manera más cercana posible al tiempo real y que el sistema funcione de una forma lo más automática posible. Para ello alimentaremos a nuestro sistema de seguimiento con las medidas obtenidas por las distintas cámaras usadas en el escenario durante la detección y con las trayectorias que hayan sido procesadas préviamente. Podemos divir nuestro objetivo principal en varios ojetivos intermedios:

1. Obtener las trayectorias individuales de un conjunto de objetivos en movimiento.
2. Calcular las trayectorias medias a partir de las trayectorias individuales obtenidas con anterioridad.
3. Adaptar los algoritmos de seguimiento a la información de entrada correspondiente a las trayectorias medias.
4. Minimizar la tasa de error correspondiente al seguimiento de los objetos, teniendo en cuenta situaciones complejas como los cruces entre jugadores y las oclusiones.
5. Conseguir que el coste computacional no aumente demasiado respecto al no uso de trayectorias medias, de manera que los sistemas de seguimiento funcionen de la forma más cercana al tiempo real en un ordenador personal.

Cuando se producen cruces entre jugadores, o lo que es peor, cuando se produce una agrupación muy concentrada de jugadores en la misma zona (como suele pasar cuando hay faltas cerca del área o saques de esquina), en muchas ocasiones el sistema de detección y seguimiento llega a confundir la identificación de algunos jugadores. Mediante la realimentación del sistema a través de las trayectorias "habituales" se pretende que el sistema de seguimiento sea capaz de que, una vez los objetivos se hayan vuelto a separar, volver a identificar y seguir a cada jugador de forma correcta.

Puesto que un campo de futbol es un escenario bastante grande, hemos decidido utilizar 3 cámaras para obtener nuestra información de partida de una manera aceptable. En nuestro caso estaban colocadas de forma que cada una enfocaba a una zona distinta del campo: izquierda, centro y derecha. No obstante el tener más puntos de vista siempre ayudará mejor a resolver los problemas que puedan aparecer. Independientemente del número de cámaras usado, una vez obtengamos las medidas en cada cámara, dichas medidas serán pasadas a un plano común. Como veremos posteriormente, se obtiene mejor resultado en el caso de usar la información de las 3 cámaras de manera conjunta que tratándola de manera independiente.

En los campos de futbol, en realidad, el fondo no presenta un color tan uniforme como podría parecer a primera vista. Por otra parte, los colores de los equipamientos de los jugadores de ambos equipos, que será el criterio que usaremos para distinguirlos, sufren variaciones en un escenario tan grande en el que se producen cambios de iluminación importantes. Además, la presencia de sombras durante los partidos nocturnos también puede actuar como un ruido adicional. De tal manera podemos concluir con que un

campo de futbol se acerca más a un escenario real que a un entorno controlado. Por lo tanto, los algoritmos de detección y seguimiento que han sido desarrollados podrían ser usados en otros escenarios sin demasiadas dificultades si disponemos de los modelos (de los objetos a detectar y seguir) adecuados.

1.3. Estructura del proyecto

El proyecto estará dividido en 4 capítulos y 3 anexos, los cuáles estarán organizados de la siguiente manera:

- Capítulo 1: Introducción.
  Capítulo inicial en el que se describe el proyecto en términos generales, la motivación existente para su realización y los objetivos finales de dicho proyecto.

- Capítulo 2: Trayectorias "habituales" o "comunes".
  En el tercer capítulo se describe el método usado para la obtención de trayectorias iniciales y el cálculo de trayectorias medias. Además se explica el método usado para la inclusión de las trayectorias medias obtenidas previamente, en los algoritmos de seguimiento.

- Capítulo 3: Análisis de resultados.
  En el penúltimo capítulo nos encargaremos de analizar los resultados obtenidos en las distintas pruebas realizadas a partir de nuestros algoritmos de seguimiento.

- Capítulo 4: Conclusiones y sugerencias para futuras investigaciones.
  En el cápítulo final se valorará el grado de cumplimiento de los objetivos definidos anteriormente y cuáles son los aspectos que podrían mejorarse para un futuro.

- Anexo A: Estado del arte.
  En el primer anexo se habla de diferentes algoritmos de detección y de seguimiento los cuáles han sido desarrollados préviamente.

- Anexo B: Sistema de Detección.
  En el segundo anexo se describe el sistema de detección utilizado para obtener las medidas(supuestos jugadores que han sido detectados) que servirán como datos de entrada para nuestro sistema de seguimiento.

- Anexo C: Sistema de Seguimiento.
  En el último anexo se describen los sistemas de seguimiento utilizados en nuestro proyecto, los cuáles en un principio tan solo disponen como datos de entrada de las medidas obtenidas en el sistema de detección. Una vez sean obtenidas las trayectorias medias serán realimentados además

con la información de dichas trayectorias. Mediante dichos algoritmos de seguimiento podremos estudiar si existe mejora o no al realimentar nuestro sistema con las trayectorias medias.

CAPÍTULO 2

TRAYECTORIAS “HABITUALES” O “COMUNES”

2.1. Introducción

Llamaremos *trayectorias “habituales” o “comunes”* a aquellas trayectorias asociadas a objetos en movimiento(en nuestro caso jugadores de futbol, pero se podría aplicar a otros objetos usando los modelos adecuados), las cuáles aparecen con mayor frecuencia durante un determinado intervalo temporal en la zona espacial en la que vamos a dar uso de nuestro sistema de seguimiento.

En primer lugar obtendremos las trayectorias individuales de cada jugador para un intervalo temporal determinado. A continuación, agruparemos las más similares y obtendremos las *trayectorias “habituales”* como la media de cada grupo de trayectorias. Finalmente realimentaremos a nuestro sistema de seguimiento con dichas trayectorias, de manera que sirvan como un parámetro de entrada junto con las medidas obtenidas en la detección. Para ver si hay mejoras en cuanto al seguimiento, calcularemos cuál es el error del sistema de seguimiento con y sin el uso de trayectorias medias y compararemos los resultados.

2.2. Adquisición y filtrado previo de trayectorias individuales

Para hallar las trayectorias individuales de cada jugador, en primer lugar deberemos de obtener mediante nuestro *sistema de detección y seguimiento* (de forma supervisada), las posiciones de los diferentes jugadores a lo largo del intervalo temporal que deseemos. Es decir, deberemos de realizar un etiquetado previo de las posiciones reales de los jugadores de campo de ambos equipos(ni porteros, ni árbitros, ni el resto del personal que hay en los alrededores del terreno de juego). De tal forma dispondremos de 20 *“trackers”* (estructura de datos que contiene la posición y el estado de cada objeto en movimiento) en cada frame. Dicho etiquetado será realizado a través de un entorno gráfico como CVLAB(que funciona bajo Visual C++), el cuál se basará en nuestro *sistema de detección y seguimiento*, de manera que si alguna de las estimaciones de algún jugador es incorrecta, dicho error deberá ser corregido mediante la

Figura 2.1: Interfaz gráfica de CVLAB.

supervisión de un usuario. Podemos ver un ejemplo del la interfaz gráfica de CVLAB en la Figura 2.1. En este trabajo, guardaremos las posiciones de los diferentes jugadores de campo para 5 intervalos temporales diferentes:

- Del frame 1 al frame 15000.
- Del frame 1 al frame 3333.
- Del frame 3334 al frame 6666.
- Del frame 6667 al frame 9999.
- Del frame 10000 al frame 13333.

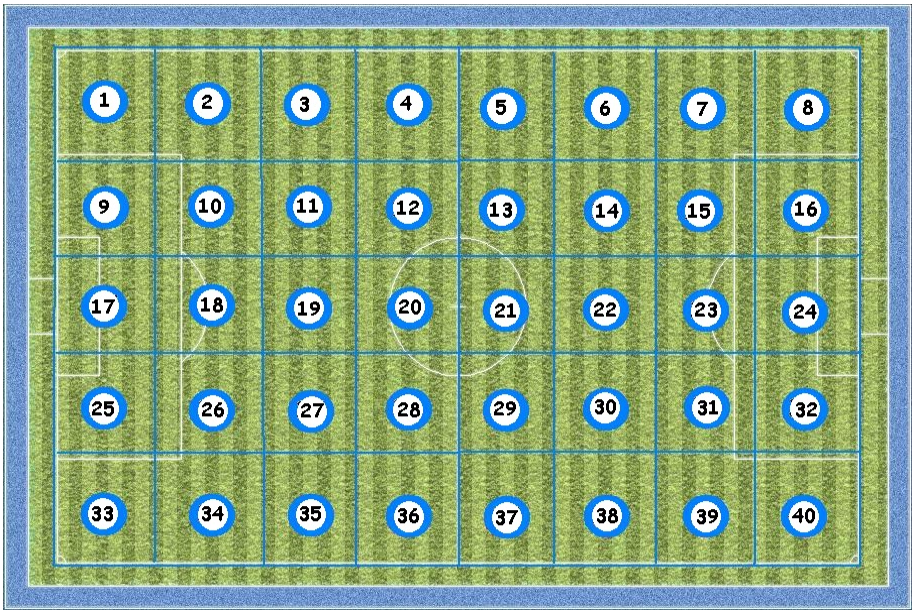

Figura 2.2: Representación gráfica de la división en cuadrantes realizada sobre el campo de futbol.

En nuestro caso, hay que tener en cuenta que el intervalo temporal correspondiente a 45 minutos será igual a 15000 fotogramas o frames.

Una vez realizado este paso previo, nos encargaremos de obtener las trayectorias individuales de cada jugador. Para ello, mediante el software matemático “Matlab”, dividiremos el campo de fubol en 40 cuadrantes uniformemente distribuidos(tal y como podemos ver en la Figura 2.2); y a continuación almacenaremos las trayectorias individuales por cuadrante en una estructura de datos determinada.

Definiremos a una trayectoria como un vector de puntos que definen la evolución temporal de un objeto en movimiento. Cada punto contendrá el mismo número de características, las cuáles pueden ser: posición, velocidad, forma, relación de aspecto, color o tamaño entre otras. En nuestro caso procesaremos las trayectorias por cuadrantes, de manera que definiremos a una trayectoria individual por cuadrante como aquella trayectoria asociada a un jugador que entra por un borde del cuadrante y sale ,después de un número de frames determinado, por otro borde de dicho cuadrante. Podemos ver un ejemplo de trayecoria individual en la Figura 2.3.

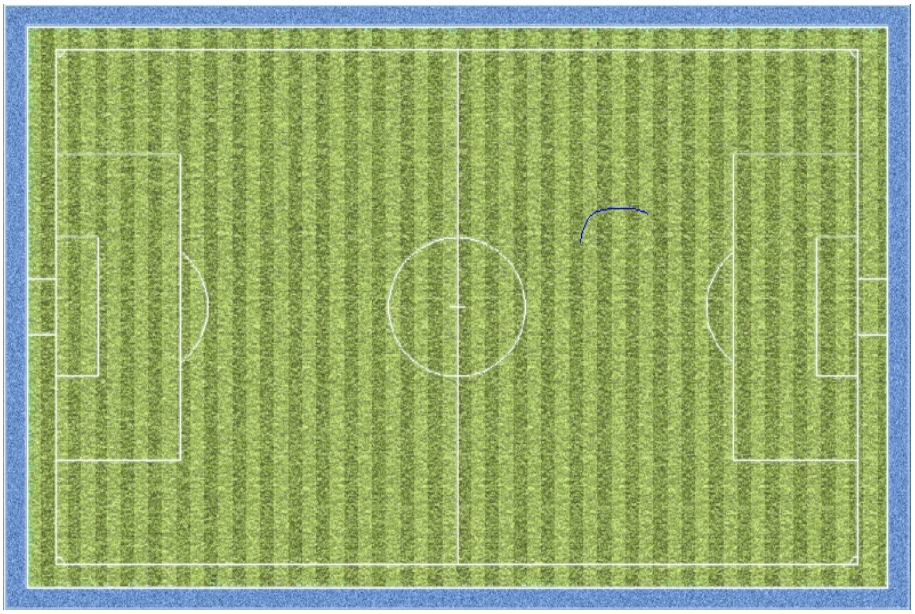

Figura 2.3: Ejemplo de trayectoria individual en el cuadrante 14.

Una vez que disponemos de las trayectorias individuales procederemos a filtrarlas, de manera que no serán consideradas como válidas aquellas trayectorias en las que se cumpla alguna de estas dos restricciones:

- Que estén formadas por un número puntos inferior a un umbral establecido (Dichas trayectorias no las consideraremos significativas).

- Que la distancia entre alguno de sus pares de puntos sucesivos sea mayor que un umbral determinado (Dichas trayectorias serán consideradas como erróneas).

Tras este filtrado, realizaremos un suavizado de las trayectorias, previo a la obtención de las trayectorias“habituales”. A la hora de obtener las trayectorias por cuadrante podremos plantear nuestro problema de dos formas diferentes:

- Obtener las trayectorias por cuadrante de *forma individual* para cada jugador.

- Obtener las trayectorias por cuadrante de *forma global* para los 20 jugadores.

En ambos casos estudiaremos si hay mejora o no respecto al caso de no usar la información de las trayectorias en nuestros sistemas de seguimiento.

2.3. Obtención de trayectorias medias

Una vez disponemos de las trayectorias por cuadrante, agruparemos las más similares para cada cuadrante y obtendremos las trayectorias “habituales” como la media de cada grupo de trayectorias. Para realizar dicho agrupamiento usaremos un algoritmo jerárquico([14]), el cuál se divide en 4 etapas:

1. Cálculo de una matriz de distancias, por cuadrante, que contenga la distancia dos a dos entre cada par de trayectorias de dicho cuadrante.
2. Búsqueda de la distancia más pequeña que esté fuera de la diagonal de dicha matriz. En caso de que dicha distancia supere el umbral de distancia que hayamos definido, el algoritmo se dará por finalizado.
3. Fusión de las dos trayectorias asociadas a la distancia mínima obtenida en la etapa anterior, mediante el cálculo de la media de dichas trayectorias.
4. Actualización de la matriz de distancias, teniendo en cuenta la eliminación de las trayectorias a partir de las cuales se ha realizado la fusión y la inclusión de la nueva trayectoria obtenida. Tras esto se vuelve a la fase 2.

A continuación describiremos los métodos asociados al cálculo de la *distancia entre dos trayectorias* y los asociados al cálculo de la *media de n trayectorias*.

2.3.1. Distancia entre dos trayectorias

Introducción

En la actualidad, la distancia más conocida entre dos trayectorias es la distancia de Hausdorff([13]). Dicha distancia solo es commutativa si se calcula el máximo de aplicarlo dos veces para cada par de trayectorias:

$$H(A,B) = \max\{h(A,B), h(B,A)\} \tag{2.1}$$

$$h(A,B) = \max_{a\in A}\{\min_{b\in B}\{d(a,b)\}\} \tag{2.2}$$

Esta distancia tan solo tiene en cuenta un punto de cada trayectoria a la hora de calcular dicha distancia, de manera que el ruido en los puntos iniciales y finales puede alterar el resultado final. Existen modificaciones de la Distancia Hausdorff como la Distancia Parcial de Hausdorff o la Distancia de Proximidad de Hausdorff, pero en ambas modificaciones a pesar de obtener mejores resultados que en la original,

existen otros problemas que complican el uso de dichas Distancias. De tal manera, se pretenderá diseñar un método que resuelva los problemas que sucedían con la Distancia de Hausdorff y sus modificaciones. Buscaremos que la nueva Distancia sea commutativa y que sea capaz de obtener resultados coherentes con trayectorias ruidosas.

Método usado

Comenzaremos clasificando cada trayectoria respecto al resto de trayectorias en dos grupos, dividiéndolas entre las que podemos medir la distancia entre ellas de una manera coherente y las que no. Al primer grupo lo llamaremos *trayectorias "cercanas"* mientras que al segundo grupo lo llamaremos *trayectorias "lejanas"*. En el caso de que dos trayecorias sean cercanas se calculará la distancia entre ellas.

Consideraremos que dos trayectorias son cercanas entre sí en el caso de que cumplan estas dos condiciones a la vez:

$$C_1 = \frac{d(a_s, b_s) + d(a_f, b_f)}{2} < th_1 \tag{2.3}$$

$$C_2 = \text{mín} \left\{ \frac{\sum_{i=s}^{f-1} d(a_i, a_{i+1})}{\sum_{i=1}^{n_A - 1} d(a_i, a_{i+1})}, \frac{\sum_{i=s}^{f-1} d(b_i, b_{i+1})}{\sum_{i=1}^{n_B - 1} d(b_i, b_{i+1})} \right\} > th_2 \tag{2.4}$$

donde:

n_A, n_B : Número de puntos que definen las trayectorias A y B respectivamente.

$d(x, y)$: Distancia entre los puntos x e y.

c_p^i : Punto más cercano (de la otra trayectoria) al punto p de la trayectoria i .

a_s, b_s : Puntos que definen la primera distancia mínima.

a_f, b_f : Puntos que definen la última distancia mínima.

$$a_s = \begin{cases} a_1 & si \;\; d(a_1, c_1^A) \leq d(b_1, c_1^B) \\ c_1^B & si \;\; d(a_1, c_1^A) > d(b_1, c_1^B) \end{cases} \tag{2.5}$$

$$a_f = \begin{cases} a_{n_A} & si \;\; d(a_{n_A}, c_{n_A}^A) \leq d(b_{n_B}, c_{n_B}^B) \\ c_{n_B}^B & si \;\; d(a_{n_A}, c_{n_A}^A) > d(b_{n_B}, c_{n_B}^B) \end{cases} \tag{2.6}$$

$$b_s = \begin{cases} b_1 & si \;\; d(b_1, c_1^B) \leq d(a_1, c_1^A) \\ c_1^A & si \;\; d(b_1, c_1^B) > d(a_1, c_1^A) \end{cases} \tag{2.7}$$

$$b_f = \begin{cases} b_{n_B} & si \;\; d(b_{n_B}, c_{n_B}^B) \leq d(a_{n_A}, c_{n_A}^A) \\ c_{n_A}^A & si \;\; d(b_{n_B}, c_{n_B}^B) > d(a_{n_A}, c_{n_A}^A) \end{cases} \tag{2.8}$$

Podemos ver un ejemplo de puntos que definen la primera y la última distancia mínima entre dos trayectorias, en la Figura 2.4. Las trayectorias que no cumplan las dos condiciones a la vez se considerará que

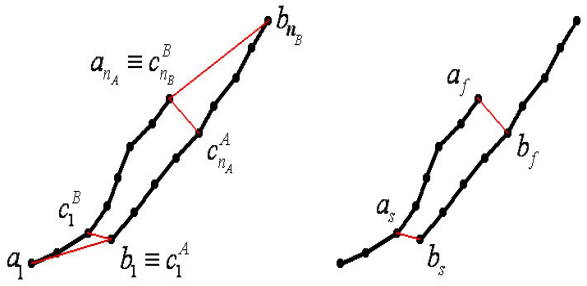

Figura 2.4: Ejemplo en el que se muestran los puntos que son necesarios definir para comprobar si dos trayectorias están lo suficientemente cerca como para medir su distancia con exactitud o no(para considerarlas como “cercanas” o “lejanas”).

están demasiado alejadas como para poder calcular su media de forma que podamos obtener un resultado coherente. Para calcular la distancia entre dos trayectorias “cercanas” dividiremos a ambas trayectorias en segmentos, situando el mismo número de puntos n de forma equidistante a lo largo de ambas trayectorias. A dichos puntos los llamaremos *“puntos clave”*. Dada una trayectoria A, sus “puntos clave” serán:

$$a'_i = a_{[s+\frac{i.(f-s)}{n-1}]} \qquad i = 0, 1, 2, ..., n-1 \tag{2.9}$$

De la misma forma, los “puntos clave” de una trayectoria B serán:

$$b'_i = b_{[s+\frac{i.(f-s)}{n-1}]} \qquad i = 0, 1, 2, ..., n-1 \tag{2.10}$$

donde [x] representará al entero más próximo a x. Una vez se calculen todos los “puntos clave” para ambas trayectorias, se calculará el punto más cercano a cada “punto clave” respecto de la otra trayectoria. De tal manera dispondremos de una distancia parcial para cada “punto clave” :

$$d_{a'_i} = \min_{b \in B} d(a'_i, b) \qquad d_{b'_i} = \min_{a \in A} d(b'_i, a) \qquad i = 0, 1, 2...n \tag{2.11}$$

Finálmente, calcularemos la distancia total entre dos trayectorias “cercanas” como:

$$D = \frac{\sum_{i=1}^{n} \min\{d_{a'_i}, d_{b'_i}\}}{n} \tag{2.12}$$

Al calcular la distancia entre dos trayectorias, no tenemos por qué referirnos tan solo a la distancia posicional, sino que podemos tener en cuenta otras características como puede ser la velocidad. De tal manera, la distancia parcial entre dos puntos podrá ser calculada como una media ponderada de las distancias de cada una de estas características. Para ello habrá que definir un peso para cada característica. Si disponemos de n características, representadas como c_1, c_2,...,c_n, con sus correpondientes pesos w_1,

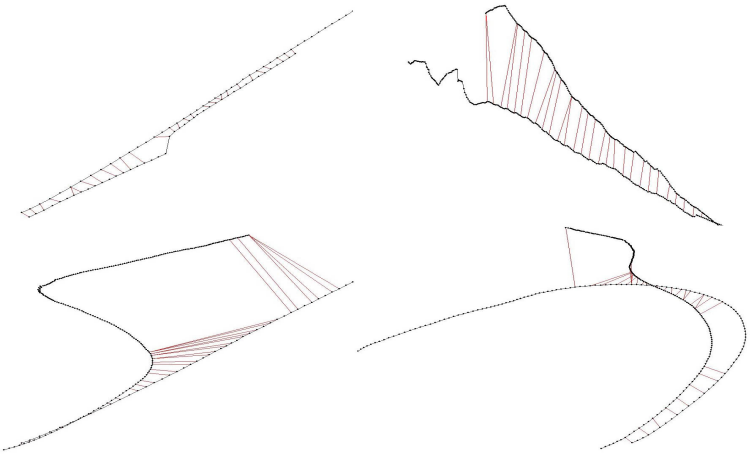

Figura 2.5: Ejemplos de distribuciones de distancias parciales entre trayectorias.

w_2,...,w_n, definiremos la distancia parcial entre dos puntos como:

$$d = \sum_{i=1}^{n} w_i.d_{f_i} \tag{2.13}$$

Mediante este método calculamos una serie de distancias parciales entre los puntos de cada trayectoria y después realizamos su media. De la manera que está planteado el cálculo de la distancia global, el algoritmo tiende a evitar calcular una distancia parcial entre los puntos en los que las trayectorias están más separadas momentáneamente. En caso de que las trayectorias permanezcan separadas durante un período largo de tiempo, el algoritmo acabará por calcular la distancia parcial entre puntos en los que las trayectorias estén más alejadas. Podemos ver algunas distribuciones de distancias parciales en la Figura 2.5.

Uno de los parámetros más importantes a tener en cuenta será el número de puntos usados para realizar el cálculo de la distancia total. En caso de trayectorias bastante cercanas, el número de puntos usados no suele afectar al resultado obtenido (distancia total obtenida). En caso de trayectorias con una parte común(cercana) y otra parte divergente(lejana), el efecto de dicho extremo divergente será mayor cuantos menos “puntos clave” se usen, de forma que la distancia total se reducirá si aumentamos el número de “puntos clave”. Podemos ver un ejemplo del efecto que tiene el número de puntos usado en los dos casos comentados anteriormente en la Figura 2.6.

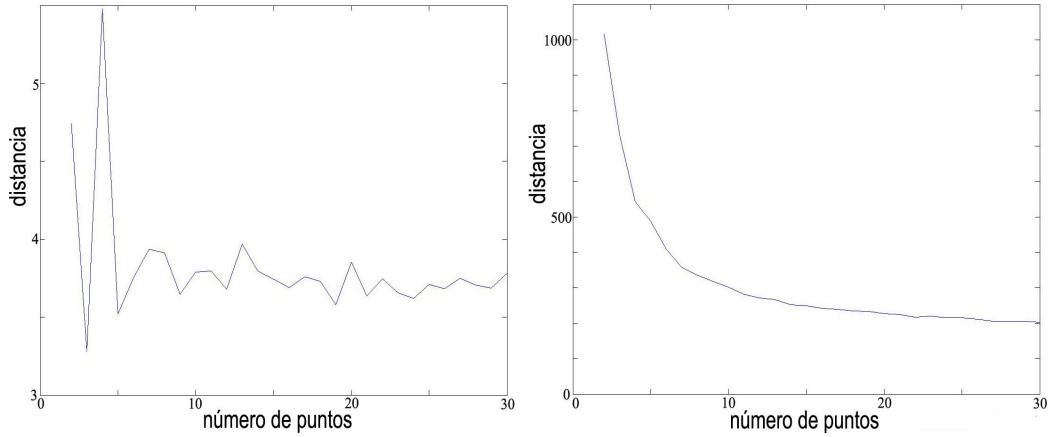

Figura 2.6: Efecto del número de puntos usados para obtener la distancia total entre dos trayectorias. En la imagen de la izquierda podremos ver el efecto causado en el caso de dos trayectorias cercanas y en la imagen de la derecha podremos ver el efecto causado en el caso de dos trayectorias con una parte común y otra divergente.

2.3.2. Media de dos trayectorias

Introducción

Una vez disponemos de la matriz de distancias dos a dos entre cada par de trayectorias (en el caso de trayectorias lejanas dicha distancia será igual a un número fijo lo suficientemente alto como para indicar que no son “cercanas”) , si la distancia más pequeña que esté fuera de la diagonal de dicha matriz está por debajo del umbral de distancia, calcularemos la media de las trayectorias asociadas a dicha distancia. En un principio vamos a ver como realizar la media para dos trayectorias y después veremos como podemos extender el método para calcular la media de n trayectorias.

La media de dos trayectorias deberá cumplir una serie de propiedades:

- El número de puntos de la trayectoria media deberá ser aproximadamente la media de los puntos de las trayectorias usadas para calcularla.
- La distribución de la trayectoria media deberá ser lo más parecida a una mezcla entre las distribuciones de las dos trayectorias usadas para calcularla.
- Deberá ser lo más commutativa posible.
- Deberá de abarcar una zona que sea la unión de las zonas abarcadas por las trayectorias usadas.

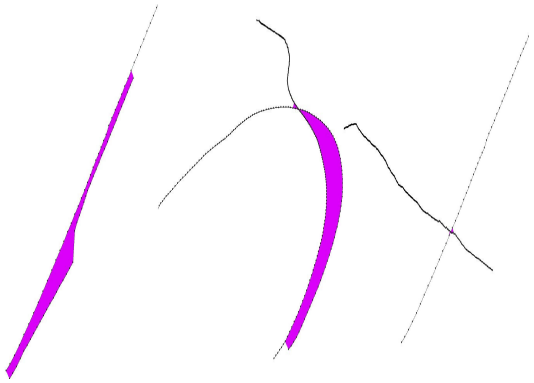

Figura 2.7: Ejemplo de partes comunes para distintos pares de trayectorias.

Método usado

Lo primero que haremos es dividir las trayectorias en dos partes antes de calcular su media: parte común y parte no común (o divergente). Previamente a esto, definiremos el concepto de *"emparejamiento de puntos"*. Cada emparejamiento estará formado por un punto de cada trayectoria, de manera que sean los puntos más cercanos entre ambas trayectorias de forma recíproca. De tal manera definiremos a la parte común de ambas trayectorias como la región de cada una de ellas que se encuentre entre el primer y el último emparejamiento de puntos, tal y como podemos ver en los ejemplos de la Figura 2.7.

Las trayectorias podrán disponer hasta de tres partes: parte divergente inicial, parte común y parte divergente final. En ocasiones dispondrán de dos parte e incluso solo de una.

En un principio tenderíamos a pensar que la trayectoria media va a pasar por los puntos medios de los segmentos que unen los "emparejamientos de puntos". Pero esto no es tan fácil como parece, ya que normalmente, no todos los puntos (de la parte común) de las trayectorias a unir tienen emparejamiento con otros puntos de la otra trayectoria. Por lo tanto deberemos de hacer una interpolación si queremos hallar el resto de los puntos. Aun así, suele suceder que el número de puntos que hay en cada trayectoria entre cada par de "emparejamientos", es diferente; y por tanto complicará dicha interpolación. Para ello, una posible solución es la de realizar muestreos locales entre cada par de "emparejamientos", de forma que el número de puntos de ambas trayectorias sea el mismo. De este modo llamaremos *"correspondencia"*, en lugar de "emparejamiento", a la relación de cada uno de los puntos remuestreados de una trayectoria con respecto a los puntos remuestreados de la otra trayectoria.

Nuestro objetivo, por lo tanto será disponer del mismo número de puntos entre cada par de "emparejamientos". De tal manera, consideraremos que la sección entre dos emparejamientos en la trayectoria A tiene n_a puntos y que la sección entre dos emparejamientos en la trayectoria B tiene n_b puntos. El número de puntos que deberemos tomar en la trayectoria media para esa sección, deberá ser de $(n_A + n_B)/2$

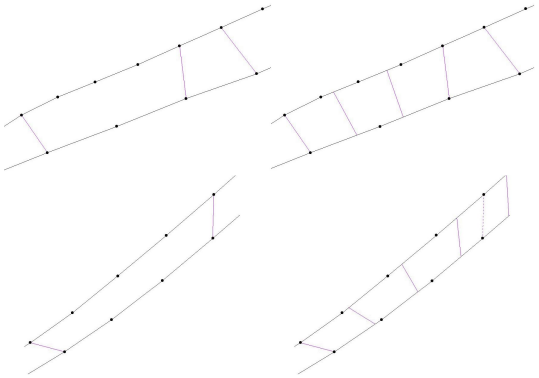

Figura 2.8: Representación gráfica del remuestreo de un número de puntos entero(imágenes de la izquierda) y de un número de puntos no entero (imágenes de la derecha) en una sección entre dos emparejamientos para dos pares de trayectorias diferentes.

puntos, ya que es una de las propiedades que queremos que cumpla la media de dos trayectorias.

El número medio de puntos entre dos emparejamientos puede no ser entero. En tal caso, si se redondeara a un número de puntos entero, podría ocurrir que la trayectoria obtenida tuviera un número de puntos diferente o que su distribución variara. Para evitar esto crearemos un método que siga estos pasos:

1. En primer lugar, asignaremos un índice a cada uno de los puntos de ambas trayectorias, el cuál represente el número de puntos de la trayectoria media que se generaría a partir de ese punto de una de las trayectorias. Este índice puede no ser entero.
2. A continuación se buscarán los índices enteros. Si es necesario, se interpolará.
3. Por último, se trazarán los segmentos entre índices enteros.

Podemos ver una muestra de este método en la Figura 2.8. Este método puede ser explicado matemáticamente de la manera que expondremos a continuación. Si llamamos a_i a los índices de puntos en la trayectoria A y b_i a los índices de puntos en la trayectoria B - siendo n_i^a y n_i^b el número de puntos entre emparejamientos en cada trayectoria- los índices intermedios serán:

$$a_{i+k} = a_i + \frac{(n_i^a + n_i^b) \cdot k}{2 \cdot n_i^a} + r_i \quad k = 1, 2, ..., n_i^a - 1 \tag{2.14}$$

$$b_{i+k} = b_i + \frac{(n_i^a + n_i^b) \cdot k}{2 \cdot n_i^b} + r_i \quad k = 1, 2, ..., n_i^b - 1 \tag{2.15}$$

$$r_{i+1} = dec\left\{ \ r_i + \tfrac{n_i^a + n_i^b}{2} \ \right\} \tag{2.16}$$

donde:

r_i : resto obtenido al remuestrear.

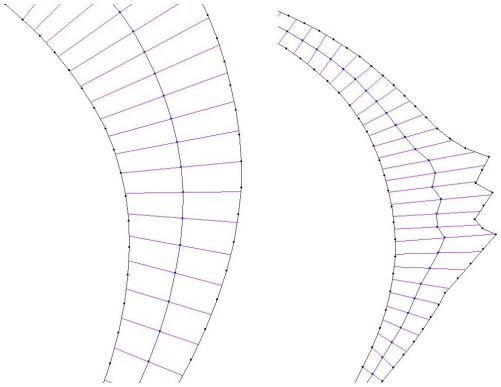

Figura 2.9: Ejemplos de dos pares de trayectorias con sus respectivas correspondencias y su trayectoria media, en la parte común de dichas trayectorias.

$dec\{x\}$: parte decimal de x.

Si se desea mantener la simetría a lo largo de toda la trayectoria, será necesario calcular de forma previa el número medio de puntos de la parte común de las trayectorias de las que se está calculando la media. En caso de no ser un número entero, deberá de ser añadido un valor a todos los índices de la parte común, de forma que la parte decimal se reparta entre los dos extremos. Sea R el resto global obtenido en la parte común de las trayectorias, inicializaremos el primer resto de esta manera:

$$r_o = R/2 \qquad donde \qquad R = dec\left\{ \frac{L_a^c + L_b^c}{2} \right\} \tag{2.17}$$

siendo:

L_a^c,L_b^c : Longitud de la parte común de las trayectorias A y B respectivamente.

Una vez dispongamos de los segmentos asociados a todos los emparejamientos y correspondencias entre trayectorias, calcularemos el punto medio para cada uno de ellos, de forma que se obtendrán los puntos asociados a la trayectoria media, tal y como podemos ver en la Figura 2.9. Fuera de la parte común de las trayectorias a unir, tanto en la parte divergente inicial como en la parte divergente final (donde no existen correspondencias entre ellas), se realizara una operación de "morphing". Consideraremos que la trayectoria que disponga de mayor longitud tendrá un mayor peso. Por tanto, si llamamos (x_i^a, y_i^a) y (x_i^b, y_i^b) a la primera o a la última correspondencia y L_a y L_b a las longitudes de las partes divergentes de las trayectorias, los puntos seleccionados para obtener la trayectoria media (x_i^s, y_i^s) fuera de la parte común serán:

$$(x_i^s, y_i^s) = \frac{L_a}{L_a + L_b} \cdot (x_i^a, y_i^a) + \frac{L_b}{L_a + L_b} \cdot (x_i^b, y_i^b) \tag{2.18}$$

En la Figura 2.10 podremos ver un par de ejemplos relativos al cálculo de trayectorias medias en la parte divergente.

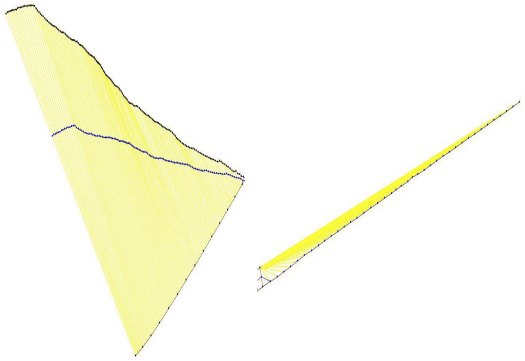

Figura 2.10: Ejemplos de dos pares de trayectorias con sus respectivas correspondencias y su trayectoria media, en la parte divergente de dichas trayectorias.

2.3.3. Media de n trayectorias

La media de n trayectorias deberá de cumplir, además de las propiedades mencionadas anteriormente para dos trayectorias, la propiedad de la asociatividad. En primer lugar calcularemos la media de dos de las trayectorias y a continuación iremos añadiendo el resto de una en una; calculando una media ponderada, en la que dicha ponderación será igual al número de trayectorias originales que han sido usadas para calcular dicha trayectoria media.

Este método no garantiza el cumplimiento de la propiedad asociativa, pero se aproxima bastante. Fuera de la parte común no se garantiza el cumplimiento de la propiedad asociativa. Dentro de la parte común se garantiza aunque con pequeños errores de debidos a redondeos, ya que las posiciones de las correspondencias no resultan ser exactamente las mismas. Si el número de puntos tomado fuese infinito se garantizaría al $100\,\%$ en la zona común.

Conforme se van añadiendo más trayectorias, estos errores tienden a reducirse, de manera que el peor caso para que se cumpla la propiedad asociativa será aquel en el que se realice la media de tres trayectorias. En la Figura 2.11 podemos ver un ejemplo en el que se ha realizado la media para tres trayectorias de las tres formas posibles. Por otra parte, este método es bastante resistente al ruido. En la Figura 2.12 podemos ver como la trayectoria media presenta menos ruido que las tres trayectorias iniciales a partir de las que se ha obtenido dicha trayectoria media. Esto se debe a que al calcular la media, el ruido de media nula tiende a compensarse. Si posteriormente realizamos un filtrado de dicha trayectoria media, aún eliminaremos una mayor cantidad de ruido.

Podemos ver el proceso seguido para la obtención de trayectorias medias, después de algunas iteraciones, en el ejemplo de la Figura 2.13.

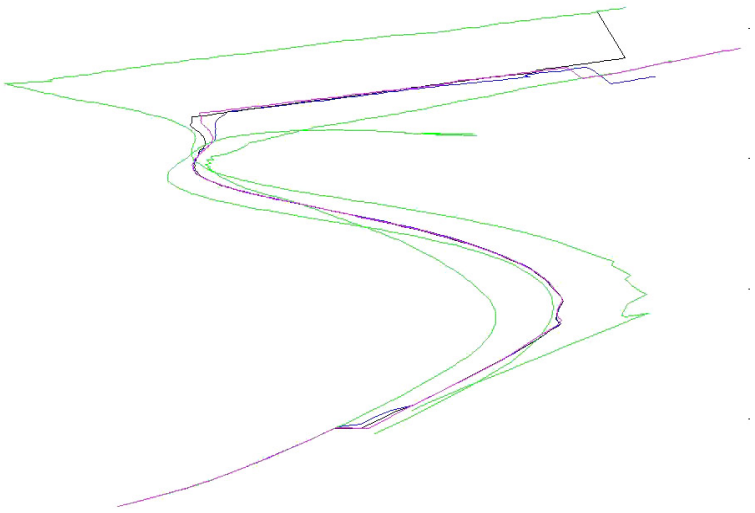

Figura 2.11: Cálculo de la trayectoria media para un grupo de tres trayectorias. En color verde, las trayectorias originales. En colores negro, azul y morado, las trayectorias medias obtenidas de las tres formas posibles.

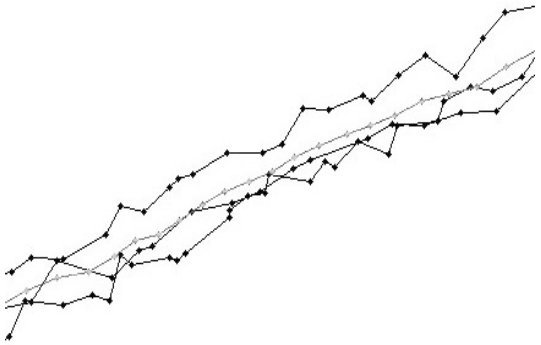

Figura 2.12: Cálculo de la trayectoria media para un grupo de tres trayectorias. Podemos ver como la trayectoria media dispone de menos ruido que las trayecrorias iniciales a partir de las cuáles se ha obtenido dicha trayectoria.

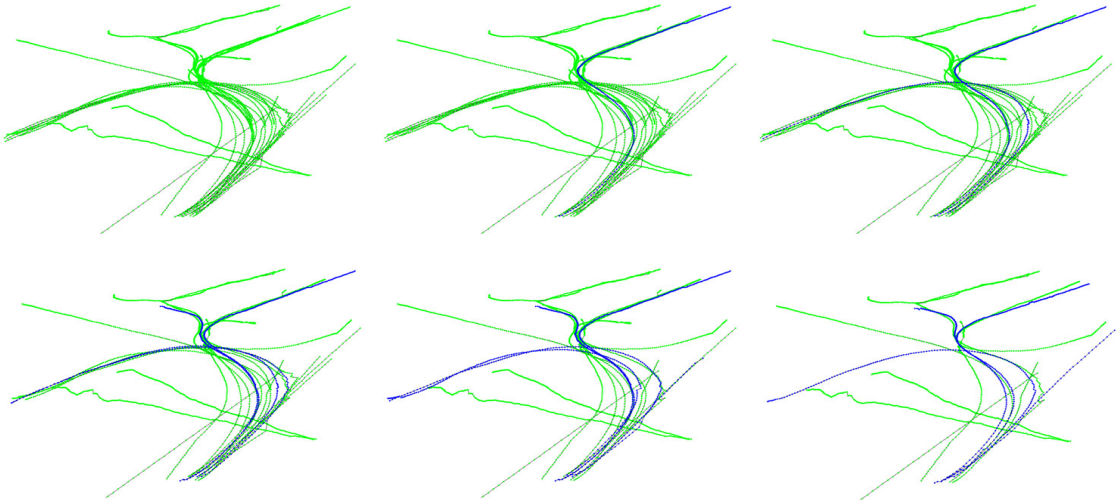

Figura 2.13: Ejemplo del proceso de obtención de trayectorias medias. Las trayectorias iniciales se muestran en color verde y los agrupamientos de éstas - los cuáles habrá un momento en que sean considerados como trayectorias medias - en color azul.

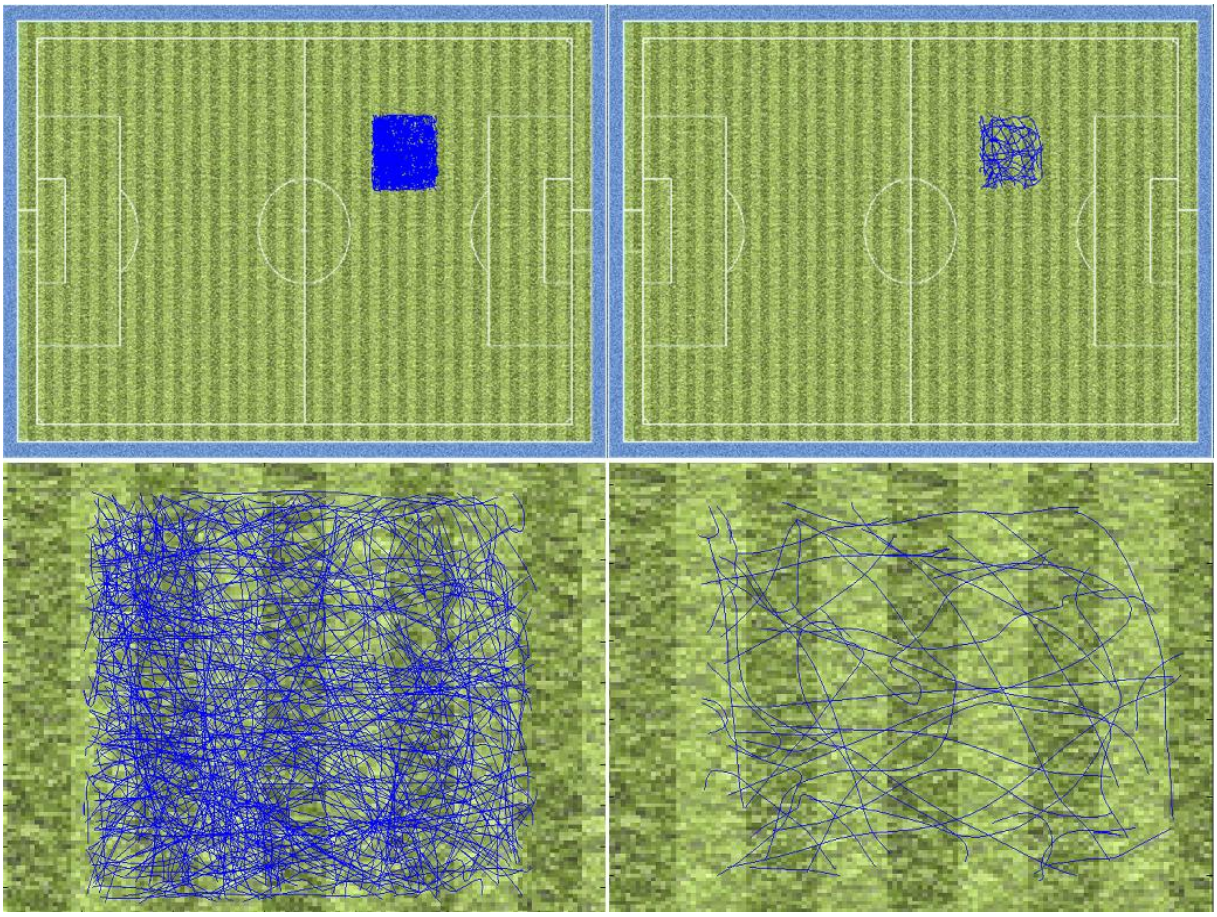

Figura 2.14: En la figura de la izquierda(superior) podemos ver las trayectorias individuales que se han obtenido inicialmente para el cuadrante 14, mientras que en la figura de la derecha(superior) podemos ver las trayectorias medias que se han obtenido a partir de dichas trayectorias individuales. En las figuras inferiores podemos ver lo mismo que en las superiores pero con un zoom.

Por otra parte podemos ver un ejemplo de obtención de trayectorias medias a partir de trayectorias individuales en uno de los cuadrantes en los que se divide el campo de futbol en la Figura 2.14.

2.4. Inclusión de las trayectorias "habituales" en los algoritmos de seguimiento

2.4.1. Introducción

Una vez disponemos de las trayectorias "habituales" en cada cuadrante, deberemos realimentar a nuestro sistema de seguimiento con dichas trayectorias, de manera que sirvan como parámetro de entrada

junto con las medidas obtenidas en la detección. NOTA: Se recomienda leer los anexos B y C de forma previa a la lectura de los próximos apartados del capítulo 2.

2.4.2. Inclusión de trayectorias “habituales” en el algoritmo de seguimiento MSUKF

Tal y como se comenta en el Anexo C (*Sistema de Seguimiento*), Apartado 3, una vez hemos realizado las predicciones de estado, el matching de las predicciones y las medidas, y las predicciones de medida, pasaremos a calcular la estimación de estado para cada jugador.

En primer lugar, una vez disponemos de las trayectorias “habituales” y de las medidas asociadas a cada cámara para cada jugador(medidas, las cuáles han sido pasadas a las coordenadas del plano del suelo), para cada medida que haya sido asociada a cada cámara(para los distintos jugadores), se calculará la ganancia de Kalman, de la cuál nos interesarán los valores asociados a las coordenadas X e Y $(K_X(cam), K_Y(cam) \quad cam \in (1,3))$. Cuanto más cercanos a 1 sean los valores de las dos componentes de K, mejor consideraremos que son las medidas tomadas por las cámaras. De tal manera para nuestras trayectorias calcularemos un valor KTRAY tal que:

$$\begin{array}{ll} KTRAY_X = 1 - max(K_X(cam)) \; y \; KTRAY_Y = 1 - max(K_Y(cam)) & si \;\; nMed \neq 0 \\ KTRAY_X = 1 \;\; y \;\; KTRAY_Y = 1 & si \;\; nMed = 0 \end{array} \tag{2.19}$$

$$KTRAY = \frac{\sqrt{((KTRAY_X)^2 + (KTRAY_Y)^2)}}{\sqrt{2}} \tag{2.20}$$

Es decir, si el número de medidas asociadas a cada tracker(hasta un máximo de tres medidas, una por cámara) es distinto de cero entonces $KTRAY_X = 1 - max(K_X(cam))$ y $KTRAY_Y = 1 - max(K_Y(cam))$. En caso contrario $KTRAY = KTRAY_X = KTRAY_Y = 1$

Usaremos dos métodos diferentes para trabajar con las trayectorias “habituales”:

- En el primer método, de entre todas las trayectorias que aparecen en el cuadrante en el que se encuentra el jugador, usaremos la información de aquellas que se encuentren por debajo de un umbral(área de gating); el cuál estará asociado a la distancia entre el estado anterior del jugador cuyo estado queremos estimar y el punto más cercano de cada trayectoria respecto a este estado anterior.

- En el segundo método calcularemos la suma de dos distancias:

  - Distancia entre la posición asociada al Estado anterior de un jugador y el punto más cercano de la trayectoria respecto a dicho Estado anterior.
  - Distancia entre la posición asociada al Estado predecido de un jugador y el punto siguiente al punto más cercano de la trayectoria respecto al estado anterior.

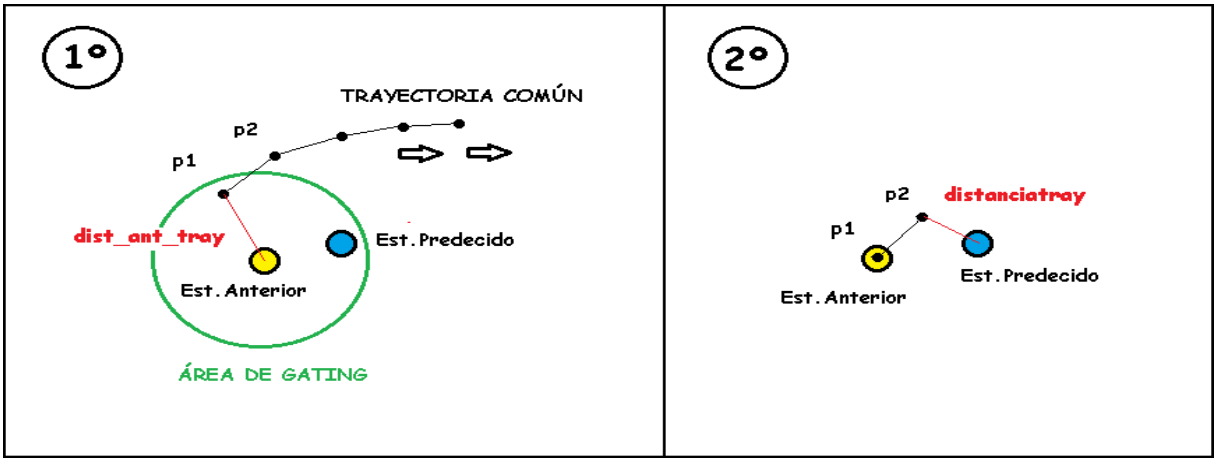

Figura 2.15: Ejemplo de como calcular el peso en el caso de una trayectoria. El peso asociado a una trayectoria será proporcional al inverso de la distancia entre el estado predecido de un jugador y el vector formado por los dos puntos más cercanos de la trayectoria respecto al estado anterior, trasladado al estado anterior. En primer lugar se creará un área de gating alrededor del estado anterior, de forma que filtremos aquellas trayectorias cuyos puntos más cercanos están dentro del área de gating. Una vez han sido seleccionadas las trayectorias, trasladaremos el vector formado por los dos puntos más cercanos de cada trayectoria al estado anterior. Calcularemos el peso asociado a cada trayectoria como el inverso de la distancia entre el estado predicho y el vector (formado por los dos puntos más cercanos) trasladado al estado anterior.

De tal manera tan solo usaremos la información de la trayectoria cuya suma de distancias sea la menor de entre todas las que haya en el cuadrante para cada jugador en cada frame.

A continuación procederemos a calcular los pesos para cada medida y para cada trayectoria.

Para cada trayectoria que intervenga "crearemos una cámara virtual" cuyo peso será igual a:

$$peso(cam) = \frac{KTRAY}{distanciatray(cam-3)} \quad cam \in (4, 4+ntray-1) \tag{2.21}$$

donde :

$distanciatray(cam)$: es la distancia de mahalanobis entre el estado predecido de un jugador y el vector formado por los dos puntos más cercanos de la trayectoria respecto al estado anterior, trasladado al estado anterior. Podemos ver un ejemplo de como calcular dicha distancia en la Figura 2.15.

Para cada medida su peso será igual a:

$$\begin{aligned} peso(cam) &= \tfrac{1}{distanciamedida(cam)} \quad cam \in (1,3) \quad si \quad medida\_asociada(cam) \neq 0 \\ peso(cam) &= 0 \qquad\qquad\qquad\quad cam \in (1,3) \quad si \quad medida\_asociada(cam) = 0 \end{aligned} \tag{2.22}$$

donde:

$distanciamedida(cam)$: es la distancia de mahalanobis que hay entre el estado predecido de un

jugador y la medida que ha sido asociada al tracker de dicho jugador para una de las tres cámaras usadas.

medida_asociada(cam): es la medida asociada a la camara cam respecto al tracker de un jugador.

Una vez hayamos calculado todos los pesos, éstos se deberán normalizar de forma que la suma de todos ellos sea igual a 1.

Finalmente calcularemos el estado estimado para un jugador j como:

$$\begin{aligned} est\_est(j)_X &= est\_pred(j)_X + \sum_{cam=1}^{3} inno\_med(cam)_X + \sum_{cam=4}^{4+ntray} inno\_tray(cam)_X \\ est\_est(j)_Y &= est\_pred(j)_Y + \sum_{cam=1}^{3} inno\_med(cam)_Y + \sum_{cam=4}^{4+ntray} inno\_tray(cam)_Y \end{aligned} \tag{2.23}$$

$$\begin{aligned} inno\_med(cam)_X &= peso(cam) \cdot K_X(cam) \cdot (Match(j,cam)_X - med\_pred(j,cam)_X) \\ inno\_med(cam)_Y &= peso(cam) \cdot K_Y(cam) \cdot (Match(j,cam)_Y - med\_pred(j,cam)_Y) \end{aligned} \tag{2.24}$$

$$\begin{aligned} inno\_tray(cam)_X &= peso(cam) \cdot KTRAY_X \cdot (MatchVir(cam-3)_X - est\_pred(j)_X) \\ inno\_tray(cam)_Y &= peso(cam) \cdot KTRAY_Y \cdot (MatchVir(cam-3)_Y - est\_pred(j)_Y) \end{aligned} \tag{2.25}$$

donde:

$inno\_med(cam)_X$: Innovación asociada a la medida obtenida en la cámara cam (coordenada X).

$inno\_med(cam)_Y$: Innovación asociada a la medida obtenida en la cámara cam (coordenada Y).

$inno\_tray(cam)_X$: Innovación asociada a la trayectoria obtenida en la cámara virtual cam(coord.X).

$inno\_tray(cam)_Y$: Innovación asociada a la trayectoria obtenida en la cámara virtual cam(coord.Y).

$est\_pred(j)_X$: Estado predecido del jugador j (coordenada X).

$est\_pred(j)_Y$: Estado predecido del jugador j (coordenada Y).

$med\_pred(j,cam)_X$: Medida predecida de la cámara cam para el jugador j (coordenada X).

$med\_pred(j,cam)_Y$: Medida predecida de la cámara cam para el jugador j (coordenada Y).

$Match(j,cam)_X$: Medida obtenida para la cámara cam respecto al jugador j (coordenada X).

$Match(j,cam)_Y$: Medida obtenida para la cámara cam respecto al jugador j (coordenada Y).

$MatchVir(cam-3)_X$: Vector formado por los dos puntos más cercanos de la trayectoria cam ($con\ \ cam \in (4, 4+ntray)$) respecto al estado anterior, trasladado al estado anterior (coordenada X).

$MatchVir(cam-3)_Y$: Vector formado por los dos puntos más cercanos de la trayectoria cam ($con\ \ cam \in (4, 4+ntray)$) respecto al estado anterior, trasladado al estado anterior (coordenada Y).

Una vez hemos realizado distintas simulaciones, hemos podido comprobar que usando la información de las trayectorias "habituales" en todos los fotogramas(siempre y cuando estuviesen dentro del área de gating del estado anterior), el resultado no siempre era satisfactorio, en cierta medida porque el peso asociado a las medidas era distorsionado por el peso de las trayectorias. Por lo tanto hemos creado un umbral para el factor KTRAY, de modo que si nuestro KTRAY no supera dicho umbral en dicho fotograma, no se usará la información de las trayectorias en ese fotograma.

Simulando para distintos umbrales, hemos podido observar que la ganancia de kalman K(la ganancia máxima entre las 3 cámaras en un frame determinado) de las cámaras como muy baja es de 0.5. Por tanto, si tomamos un umbral para nuestro KTRAY mayor que 0.5 significará que solo vamos a usar la información de las trayectorias en caso de que no haya medidas en ninguna de las tres cámaras en un frame determinado. Como se verá más adelante en el apartado de resultados, en el caso de usar la información de las trayectorias tan solo en los fotogramas en los que no haya medidas(cuando ninguna de las 3 cámaras obtengan medidas al mismo tiempo para un jugador), obtendremos mejores resultados que en el caso de no usar trayectorias.

2.4.3. Inclusión de trayectorias "habituales" en el algoritmo de seguimiento MHT

Una vez se realiza la predicción de estado y de medida para cada hipótesis, pasaremos a la etapa de asociación de datos, tal y como se explica detalladamente en el Anexo C: *Sistema de Seguimiento*, Apartado 4. La etapa de asociación de datos se divide en 5 fases:

1. Selección de medidas
2. Obtención de la matriz de combinaciones
3. Cálculo de la probabilidad a priori de cada combinación
4. Generación de hipótesis
5. Renormalización conjunta de hipótesis

En la fase de selección de medidas, una vez disponemos de las trayectorias "habituales" por cuadrante, de entre todas las trayectorias que aparecen en el cuadrante en el que se encuentra cada jugador, seleccionaremos para cada hipótesis de cada "tracker" aquellas que se encuentren por debajo del umbral umbral_tray(área de gating); el cuál estará asociado a la distancia entre el estado anterior del jugador cuyo estado queremos estimar y el punto más cercano de cada trayectoria respecto a este estado anterior.

Tras esto para cada hipótesis de cada tracker seleccionaremos aquellas trayectorias(las cuáles han sido

filtradas por el umbral anterior) en las las cuáles el vector formado por los dos puntos más cercanos de dicha trayectoria respecto al estado anterior, trasladado al estado anterior, se encuentren dentro del área de gating de la predicción de estado. Para ello calcularemos la distancia de Mahalanobis entre cada trayectoria(par de puntos de la trayectoria más cercanos al estado anterior, los cuáles han sido trasladados al estado anterior) y la predicción de estado de cada hipótesis.

A su vez seleccionaremos para cada hipótesis de cada tracker aquellas medidas las cuáles se encuentren dentro del área de gating de la predicción de estado. Para ello calcularemos la distancia de Mahalanobis entre cada medida y la predicción de estado de cada hipótesis y si dicha distancia está por debajo del umbral umbral_medida entonces la medida asociada a dicha distancia será seleccionada.

Por tanto disponemos de un umbral para seleccionar medidas y de dos umbrales para seleccionar trayectorias. De tal manera, la información obtenida de las medidas y de las trayectorias le servirá como entrada a nuestro sistema de seguimiento, de modo que tras estudiar las posibles combinaciones de medidas y trayectorias para cada tracker y tras eliminar el exceso de trackers y normalizar las probabilidades podremos pasar a estimar los estados de cada jugador.

En el algoritmo de seguimiento MHT no existe un parámetro en sí el cuál indique como son de buenas las medidas de las cámaras, como la ganancia de Kalman usada en el MSUKF(en la teoría existe, pero en el algoritmo desarrollado en la práctica no existe como tal). Si realizamos simulaciones con y sin trayectorias podemos ver que no hay grandes mejoras, incluso hay empeoramiento. En caso de usar las trayectorias solo cuando las cámaras no disponen de ninguna medida podemos encontrar que hay mejoría respecto al caso de usarlas siempre (siempre que no superen ninguno de los dos umbrales).

Podemos ver que si limitamos el número de hipótesis por jugador los resultados obtenidos son mejores que respecto a usar un número elevado de hipótesis. Esto se debe, posiblemente, a que al disminuir el número de hipótesis tenemos en cuenta las hipótesis más probables y eliminamos aquellas peores, de manera que el algoritmo de seguimiento no tienda a seguir las medidas y las trayectorias asociadas a las hipótesis más malas y que posiblemente sean equívocas.

De igual manera, si los umbrales usados para filtrar las trayectorias son más bajos, el error tiende a disminuir. Esto se debe a que al disminuir los umbrales, permitimos tan solo la entrada de información asociada a las trayectorias más cercanas a nuestros “trackers”, de manera que evitamos introducir información de trayectorias que estén demasiado alejadas, las cuáles pueden estar introduciendo ruido y haciendo que en vez de mejorar, empeore nuestro sistema se seguimiento.

Los resultados obtenidos serán comentados de una forma más detallada en el siguiente capítulo.

CAPÍTULO 3

ANÁLISIS DE RESULTADOS

3.1. Introducción

En este capítulo vamos a mostrar, analizar y comparar los resultados obtenidos con el sistema inicial de detección y seguimiento respecto al mismo sistema realimentado con las trayectorias "medias" de los distintos jugadores. Para ello compararemos el error obtenido antes y después de introducir la información de las trayectorias medias. Dicho error será calculado como la distancia euclídea entre la que sería la posición real de cada jugador en cada frame y la posición obtenida con el sistema de detección y seguimiento. Para obtener la posición real de cada jugador, etiquetaremos previamente a los distintos jugadores durante 15000 frames(que en nuestro caso equivalen a los 45 minutos de la primera parte de un partido de futbol). A continuación lanzaremos a simular un número de fotogramas determinado de dicho partido de futbol y obtendremos las posiciones estimadas de los jugadores para cada fotograma. De tal manera podremos calcular el error mencionado anteriormente.

Obviamente, si el sistema que es realimentado con las trayectorias dispone de un menor error que el sistema incial, habremos obtenido una mejora; no obstante deberemos de observar los tiempos de cómputo usados por nuestro computador para realizar los cálculos con y sin trayectorias, ya que queremos obtener una mejoría en términos de error de manera que nuestro sistema funcione a una velocidad lo más cercana posible a la del sistema inicial (el cuál pretende funcionar de la forma más cercana posible al tiempo real). Por tanto, además de obtener errores deberemos de obtener tiempos de cómputo o simulación.

3.2. Resultados obtenidos con el sistema de seguimiento MSUKF

En primer lugar, hemos obtenido el término de error para el sistema incial. Para ello hemos realizado varias simulaciones, modificando en cada una de ellas el umbral que está relacionado con la distancia de Mahalanobis entre las predicciones de estado y la medidas asociadas a éstas. De esta manera, aquellas medidas que estén por debajo de dicho umbral intenvendrán a la hora de realizar la asociación entre medidas y la predicción de estado, y a la hora de realizar la estimación de estado. En caso contrario esas medidas serán desechadas. Llamaremos a dicho umbral: umbral_medida.

	umbral_medida			
	4	5.99	9	12
Nºframes				
200	3.6158E+05	2.6894E+05	2.2431E+05	1.9091E+05
1000	1.2339E+06	1.0141E+06	1.0194E+06	8.8234E+05

Figura 3.1: Tabla que muestra el error obtenido en el sistema inicial para distintos valores de umbral_medida, para 200 y para 1000 fotogramas.

En la Figura 3.1 podemos ver los errores obtenidos para distintos umbrales. Hemos calculado el error para los 200 fotogramas iniciales y para los 1000 fotogramas iniciales. Vistos los resultados obtenidos usaremos para el resto de las simulaciones con MSUKF un valor de umbral_medida igual a 12.

En segundo lugar, para el sistema el cuál ya está realimentado con las trayectorias "habituales", calcularemos el error para distintos valores de umbral_KTRAY y distintos valores de umbral_tray.

Recordemos que cada medida asociada a cada cámara(una medida por cámara como mucho) dispone de una ganancia K, la cuál indica lo buena o mala que es la medida; y que hemos creado una variable $KTRAY = 1 - max(K(cam)) \quad cam \in (1,3)$. De tal manera, llamaremos umbral_KTRAY a aquel umbral que permitirá el uso de la información de las trayectorias en caso de que KTRAY sea mayor que el valor de dicho umbral.

Por otra parte llamaremos umbral_tray al umbral asociado a la distancia eculídea que hay entre el estado anterior de un jugador y el punto más cercano de una trayectoria que se encuentre en el mismo cuadrante en que se encuentra el jugador. De tal modo, si la distancia asociada a una trayectoria es menor a ese umbral su información será usada y en caso contrario no.

Hemos podido observar que el valor de K como muy bajo es cercano a 0.5. Recordemos que el valor de K puede ir entre 0 y 1 (de peor a mejor medida). En los casos en los que no hay medidas KTRAY=1. En las Figuras 3.2 y 3.3 podemos ver que los valores de error, para distintos valores de umbral_tray, oscilan para un $KTRAY \in (0,0{,}5)$, de manera que a veces el error es mayor y otras menor que en el caso de no usar trayectorias. Por otra parte podemos observar que para valores de KTRAY mayores de 0.5 el valor del error se estabiliza (puesto que el valor de la K de las cámaras es como muy bajo igual a 0.5, sino pasa directamente a 0; es decir, cuando no hay medidas en ninguna cámara), de manera que dicho error resulta ser menor que en el caso de no usar trayectorias.

Por tanto fijaremos umbral_KTRAY a un valor superior a 0.5, por ejemplo de 0.9, de manera que solo usaremos la información de las trayectorias cuando ninguna de las 3 cámaras disponga de medidas.

Finalmente, una vez hemos fijado los valores de umbral_medida(12) y de umbral_KTRAY(0.9), comenzaremos a realizar las simulaciones que realmente nos van a interesar. Realizaremos simulaciones

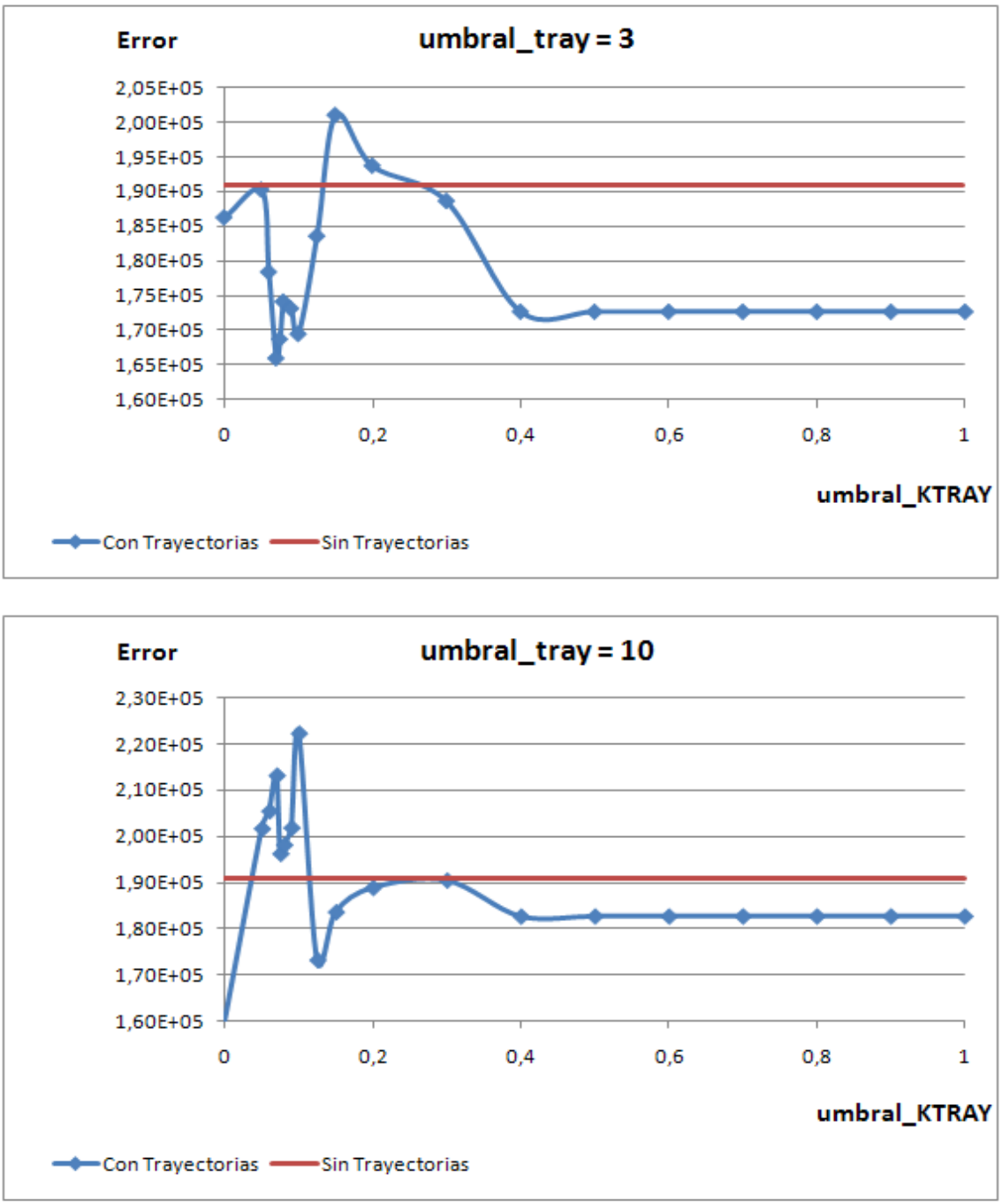

Figura 3.2: Gráficas que muestran el error obtenido en el sistema modificado, para distintos valores de de umbral_KTRAY para umbral_tray: 3 y 10 (Gráficas superior e inferior respectivamente). Estos errores han sido obtenidos realizando una simulación de 200 fotogramas o frames.

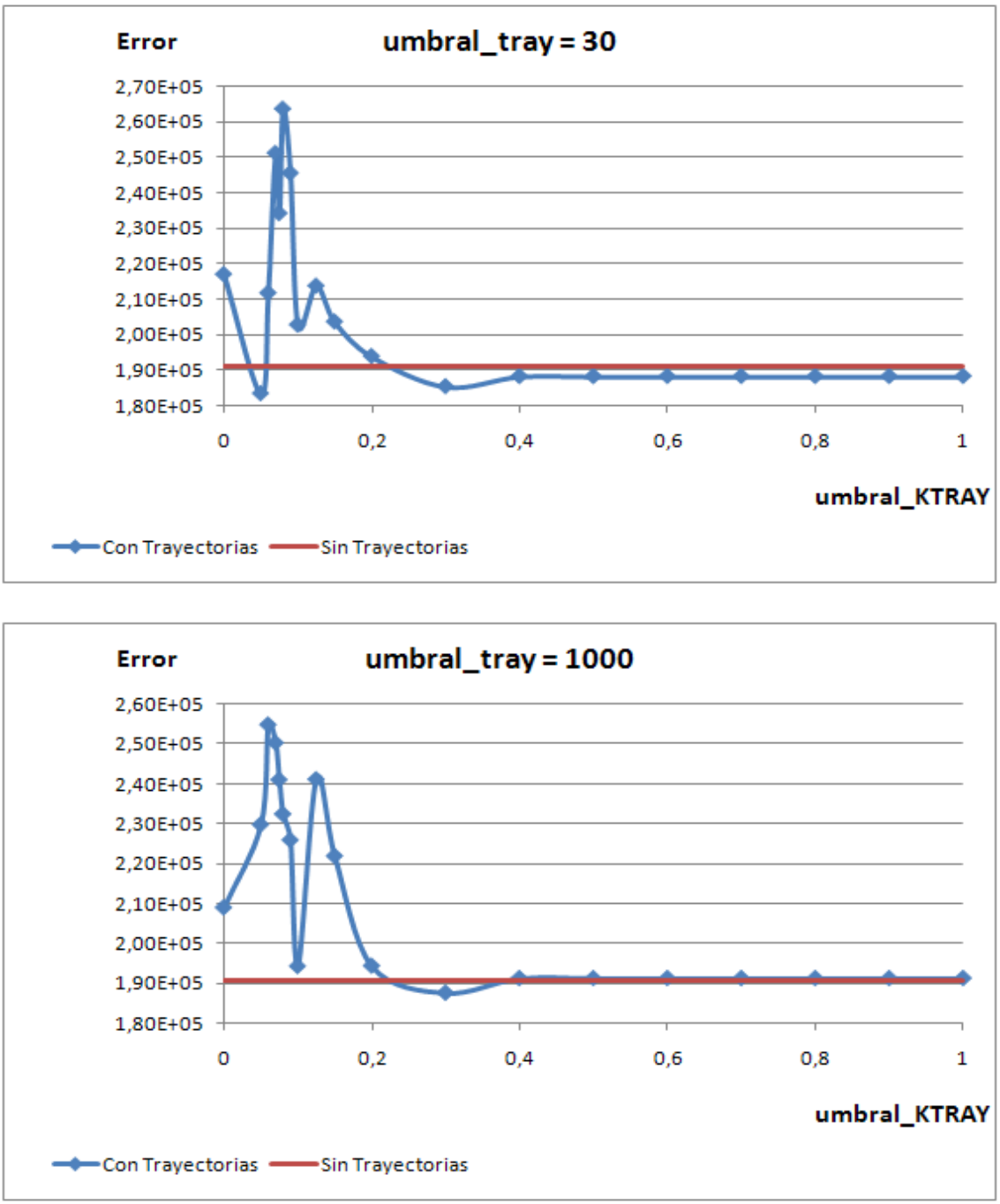

Figura 3.3: Gráficas que muestran el error obtenido en el sistema modificado para distintos valores de de umbral_KTRAY para umbral_tray: 30 y 1000 (Gráficas superior e inferior respectivamente). Estos errores han sido obtenidos realizando una simulación de 200 fotogramas o frames.

para distintos valores de umbral_tray, en un principio para 200 frames y finalmente para 1000 frames. Hay que tener en cuenta varios factores. Tal y como hemos mencionado en el capítulo anterior vamos a utilizar dos conjuntos de trayectorias "comunes" diferentes:

- Trayectorias independientes
  En la fase del procesamiento de trayectorias por cuadrante procesaremos las trayectorias de forma individual para cada jugador.

- Trayectorias globales
  En la fase del procesamiento de trayectorias por cuadrante procesaremos las trayectorias de forma conjunta para todos los jugadores.

Recordemos que disponemos de 5 grupos de trayectorias "comunes" (ya sean tratadas de forma global o de forma independiente):

- Del frame 1 al frame 15000.
- Del frame 1 al frame 3333.
- Del frame 3334 al frame 6666.
- Del frame 6667 al frame 9999.
- Del frame 10000 al frame 13333.

Por otra parte vamos a utilizar dos métodos diferentes a la hora de tratar el umbral umbral_tray:

- En el primer método, de entre todas las trayectorias que aparecen en el cuadrante en el que se encuentra el jugador, usaremos la información de aquellas que se encuentren por debajo del umbral umbral_tray(área de gating); el cuál estará asociado a la distancia entre el Estado anterior del jugador cuyo estado queremos estimar y el punto más cercano de cada trayectoria respecto a este Estado anterior.
- En el segundo método, inicialmente calcularemos la suma de dos distancias:
  - Distancia entre la posición asociada al Estado anterior de un jugador y el punto más cercano de la trayectoria respecto a dicho Estado anterior.
  - Distancia entre la posición asociada al Estado predecido de un jugador y el punto siguiente al punto más cercano de la trayectoria respecto al Estado anterior.

De tal manera tan solo usaremos la información de la trayectoria cuya suma de distancias sea la menor de entre todas las que haya en el cuadrante para cada jugador en cada frame. En este método no tendrá sentido hablar del umbral umbral_tray.

A su vez, además de trabajar solo con trayectorias "comunes", analizaremos los resultados obtenidos al usar además de dichas trayectorias, aquellas que no han sido seleccionadas para la obtención de las trayectorias "comunes" debido a que no son lo suficientemente parecidas a éstas. A este grupo de trayectorias le llamaremos trayectorias "no usuales".

A continuación, vamos a mostrar tan solo las tablas de términos de error asociados a las simulaciones de 1000 frames, ya que las de 200 frames son menos significativas, además de que se obtienen peores resultados tal y como veremos más adelante en las Figuras 3.12 y 3.13.

TRAYECTORIAS GLOBALES(TRAYECTORIAS “HABITUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	7.9228E+05	8.0870E+05	7.6464E+05	7.8902E+05
	3334-6666	8.1558E+05	7.7679E+05	7.8100E+05	7.9764E+05
	6667-9999	8.3537E+05	8.5378E+05	8.2338E+05	7.5914E+05
	10000-13333	8.0244E+05	7.9873E+05	7.3840E+05	7.9883E+05
	Todas	8.4049E+05	8.3927E+05	8.2198E+05	8.9368E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	7.6785E+05	8.0519E+05	8.0124E+05	7.8628E+05
	3334-6666	7.6057E+05	7.4999E+05	7.3702E+05	7.9117E+05
	6667-9999	8.2807E+05	7.9708E+05	7.8136E+05	7.4017E+05
	10000-13333	8.0254E+05	7.9474E+05	7.5497E+05	7.9671E+05
	Todas	8.3821E+05	8.3669E+05	8.2451E+05	8.2801E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.1263E+05	8.3433E+05	8.0866E+05
	3334-6666	7.7899E+05	7.9780E+05	8.3150E+05
	6667-9999	7.7182E+05	8.0474E+05	8.3567E+05
	10000-13333	7.9508E+05	8.1020E+05	8.3424E+05
	Todas	7.9266E+05	8.1046E+05	8.1481E+05
	umbral_tray	**25**	**30**	**10000**

Figura 3.4: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS GLOBALES(TRAYECTORIAS “HABITUALES”): MÉTODO 2

1000 frames		Error
Sin trayectorias	-	8.8234E+05
Con trayectorias	1-3333	8.5057E+05
	3334-6666	8.1590E+05
	6667-9999	8.5102E+05
	10000-13333	8.6017E+05
	Todas	8.0754E+05

Figura 3.5: Tabla que muestra el error obtenido en el sistema modificado (para 1000 fotogramas).

TRAYECTORIAS GLOBALES(T.“HABITUALES” + T.“NO USUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.0105E+05	8.0830E+05	8.3112E+05	8.3990E+05
	3334-6666	8.2241E+05	8.0201E+05	8.8944E+05	8.3793E+05
	6667-9999	7.7963E+05	7.8730E+05	8.2458E+05	8.1583E+05
	10000-13333	7.4788E+05	7.3204E+05	7.4907E+05	7.40E+05
	Todas	8.0997E+05	8.6332E+05	8.8001E+05	8.4595E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.3864E+05	8.1373E+05	8.1724E+05	7.8920E+05
	3334-6666	8.2186E+05	8.2442E+05	7.8339E+05	8.0295E+05
	6667-9999	8.1226E+05	8.2293E+05	8.4944E+05	7.8673E+05
	10000-13333	7.9099E+05	7.9215E+05	8.2288E+05	8.1204E+05
	Todas	8.4842E+05	8.7872E+05	8.3768E+05	8.1313E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	8.8234E+05	8.8234E+05	1.9091E+05
Con trayectorias	1-3333	8.1746E+05	8.1653E+05	8.4710E+05
	3334-6666	8.0565E+05	8.4111E+05	8.3933E+05
	6667-9999	7.9729E+05	7.8634E+05	7.8338E+05
	10000-13333	7.9781E+05	8.2012E+05	8.1242E+05
	Todas	8.5628E+05	8.5155E+05	8.4005E+05
	umbral_tray	**25**	**30**	**10000**

Figura 3.6: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS GLOBALES(T.“HABITUALES” + T.“NO USUALES”): MÉTODO 2

1000 frames		Error
Sin trayectorias	-	8.8234E+05
Con trayectorias	1-3333	8.4138E+05
	3334-6666	8.4761E+05
	6667-9999	7.5785E+05
	10000-13333	8.2707E+05
	Todas	8.6211E+05

Figura 3.7: Tabla que muestra el error obtenido en el sistema modificado (para 1000 fotogramas).

TRAYECTORIAS INDEPENDIENTES(T.“HABITUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.6271E+05	8.8082E+05	8.8691E+05	9.1235E+05
	3334-6666	8.8217E+05	8.8648E+05	8.8649E+05	8.8224E+05
	6667-9999	8.8018E+05	8.8018E+05	8.8022E+05	8.8104E+05
	10000-13333	8.8044E+05	9.5913E+05	9.5920E+05	9.59E+05
	Todas	8.7696E+05	8.6658E+05	8.8618E+05	8.2809E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.9795E+05	8.1768E+05	8.0694E+05	7.9972E+05
	3334-6666	8.6352E+05	8.7683E+05	8.4631E+05	8.5771E+05
	6667-9999	8.8665E+05	8.7742E+05	8.6887E+05	9.4815E+05
	10000-13333	9.5363E+05	8.9829E+05	8.9874E+05	8.8232E+05
	Todas	8.4795E+05	9.3070E+05	8.4182E+05	8.4906E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	8.8234E+05	8.8234E+05	1.9091E+05
Con trayectorias	1-3333	8.2010E+05	8.1726E+05	8.2846E+05
	3334-6666	8.5115E+05	8.3044E+05	9.0267E+05
	6667-9999	9.0595E+05	8.1492E+05	8.1353E+05
	10000-13333	8.5963E+05	8.5828E+05	9.0130E+05
	Todas	8.1741E+05	8.1884E+05	8.3641E+05
	umbral_tray	**25**	**30**	**10000**

Figura 3.8: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS INDEPENDIENTES(T.“HABITUALES”): MÉTODO 2

1000 frames		Error
Sin trayectorias	-	8.8234E+05
Con trayectorias	1-3333	8.2826E+05
	3334-6666	8.7387E+05
	6667-9999	8.2616E+05
	10000-13333	9.0945E+05
	Todas	8.7207E+05

Figura 3.9: Tabla que muestra el error obtenido en el sistema modificado (para 1000 fotogramas).

TRAYECTORIAS INDEPENDIENTES(T.“HABITUALES” + T.“NO USUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.6869E+05	8.6446E+05	8.5088E+05	8.2533E+05
	3334-6666	8.0216E+05	8.4939E+05	8.4459E+05	8.8204E+05
	6667-9999	9.0323E+05	8.6653E+05	8.8448E+05	8.7166E+05
	10000-13333	8.5695E+05	8.2410E+05	8.2347E+05	7.86E+05
	Todas	8.3037E+05	8.3313E+05	8.1155E+05	8.0688E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	8.8234E+05	8.8234E+05	8.8234E+05	8.8234E+05
Con trayectorias	1-3333	8.8689E+05	8.3398E+05	7.6908E+05	8.7418E+05
	3334-6666	8.1694E+05	8.8243E+05	8.3494E+05	8.1049E+05
	6667-9999	8.3434E+05	8.0673E+05	8.1712E+05	8.1371E+05
	10000-13333	8.2105E+05	8.3860E+05	7.7334E+05	7.9020E+05
	Todas	8.3067E+05	8.8475E+05	7.8206E+05	8.0036E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	8.8234E+05	8.8234E+05	1.9091E+05
Con trayectorias	1-3333	8.5373E+05	8.3464E+05	7.8176E+05
	3334-6666	8.0215E+05	7.9603E+05	8.1528E+05
	6667-9999	8.0387E+05	8.0565E+05	8.1058E+05
	10000-13333	8.0600E+05	8.0826E+05	7.7608E+05
	Todas	7.5686E+05	7.5712E+05	7.7682E+05
	umbral_tray	**25**	**30**	**10000**

Figura 3.10: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS INDEPENDIENTES(T.“HABITUALES” + T.“NO USUALES”): MÉTODO 2

1000 frames		Error
Sin trayectorias	-	8.8234E+05
Con trayectorias	1-3333	8.8863E+05
	3334-6666	9.0793E+05
	6667-9999	8.0109E+05
	10000-13333	9.1778E+05
	Todas	8.6470E+05

Figura 3.11: Tabla que muestra el error obtenido en el sistema modificado (para 1000 fotogramas).

Nota aclaratoria respecto a las gráficas:

- En los casos en los que el error está sombreado de color verde significa que dicho error, el cuál ha sido obtenido mediante el sistema modificado(sistema realimentado con la información de las trayectorias), es menor que el obtenido mediante el sistema original(sistema en el que no se usa la información de las trayectorias).

- En los casos en los que el error está sombreado de color amarillo significa que dicho error, el cuál ha sido obtenido mediante el sistema modificado, es algo mayor que el obtenido mediante el sistema original, pero por una diferencia pequeña.

- En los casos en los que el error no está sombreado significa que dicho error, el cuál ha sido obtenido mediante el sistema modificado, es bastante mayor que el obtenido mediante el sistema original.

3.2.1. Conclusiones(MSUKF)

A grandes rasgos podemos ver que para una simulación más larga obtenemos mejores resultados que para una simulación más corta, respecto a no usar la información de las trayectorias. Existe un mayor número de casos en los que el error obtenido con la información de las trayectorias es menor que el error obtenido sin la información de las trayectorias. Podemos ver esto en la Figuras 3.12 , 3.13 y 3.14.

		Nº frames = 200	
Trayectorias Globales		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	42	13
	Método 2	4	1
Tray.Hab+No Usuales	Método 1	44	11
	Método 2	4	1
Total		94	26

		Nº frames = 1000	
Trayectorias Globales		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	54	1
	Método 2	5	0
Tray.Hab+No Usuales	Método 1	54	1
	Método 2	5	0
Total		118	2

Figura 3.12: Tabla que muestra el número de casos en los que hay mejora y empeoramiento respecto a no usar la información de las trayectorias. En dicha tabla se muestra el caso de las trayectorias globales para los 200 y para los 1000 primeros frames.

		Nº frames = 200	
Trayectorias Independientes		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	15	40
	Método 2	1	4
Tray.Hab+No Usuales	Método 1	31	24
	Método 2	2	3
Total		49	71

		Nº frames = 1000	
Trayectorias Independientes		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	37	18
	Método 2	4	1
Tray.Hab+No Usuales	Método 1	50	5
	Método 2	2	3
Total		93	27

Figura 3.13: Tabla que muestra el número de casos en los que hay mejora y empeoramiento respecto a no usar la información de las trayectorias. En dicha tabla se muestra el caso de las trayectorias independientes para los 200 y para los 1000 primeros frames.

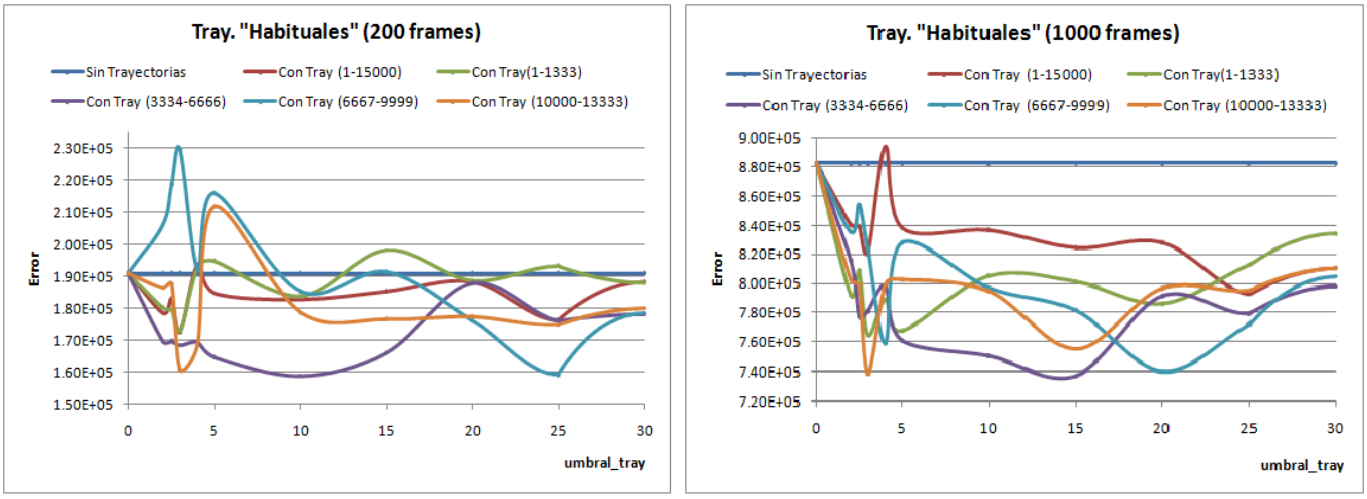

Figura 3.14: Gráfica que muestra el error para distintos valores de umbral_tray para el caso de trayectorias globales(trayectorias “habituales”) para los 200 y para los 1000 primeros frames. Se puede apreciar que para el caso de 1000 frames los errores del sistema modificado(el que usa la info. de las trayectorias) suelen estar por debajo de los errores del sistema inicial(el que no usa la info. de las trayectorias); lo cuál no ocurre en tan gran proporción para el caso de 200 frames.

Trayectorias globales					
1000 frames		Error	Nº trayectorias	Error	Nº trayectorias
Sin trayectorias	-	8.8234E+05	0	8.8234E+05	0
Con trayectorias	1-3333	8.0124E+05	59873	8.3433E+05	121853
	3334-6666	7.3702E+05	55878	7.9780E+05	109998
	6667-9999	7.8136E+05	62003	8.0474E+05	123808
	10000-13333	7.5497E+05	57815	8.1020E+05	117162
	Todas	8.2451E+05	207543	8.1046E+05	405585
	umbral_tray	15		30	

Trayectorias independientes					
1000 frames		Error	Nº trayectorias	Error	Nº trayectorias
Sin trayectorias	-	8.8234E+05	0	8.8234E+05	0
Con trayectorias	1-3333	8.0694E+05	3095	8.1726E+05	5257
	3334-6666	8.4631E+05	2417	8.3044E+05	4764
	6667-9999	8.6887E+05	1910	8.1492E+05	4159
	10000-13333	8.9874E+05	1463	8.5828E+05	3018
	Todas	8.4182E+05	21415	8.1884E+05	41652
	umbral_tray	15		30	

.

Figura 3.15: Tabla que muestra el número de trayectorias que intervienen para distintos valores de umbral_tray para los casos de trayectorias globales “habituales” y trayectorias independientes “habituales” (para 1000 frames). Se puede ver que al aumentar el valor de umbral_tray entran más trayectorias en juego, como es de esperar. Por otra parte para el mismo valor de umbral_tray se puede ver que interviene un número mayor de trayectorias en el caso de las trayectorias globales.

Por otra parte observamos que el uso de las trayectorias globales suele ser mejor frente el uso de las trayectorias independientes. Aunque antes de realizar las simulaciones podríamos pensar que iba a suceder lo contrario, si vemos los resultados obtenidos, se puede interpretar que esto es debido a que en el caso de usar trayectorias independientes, interviene un número muy pequeño de trayectorias en comparación con el uso de trayectorias globales, tal y como podemos ver para distintos valores de umbral_tray en la Figura 3.15.

Algo que puede resultar bastante interesante de ver, es que usando trayectorias “habituales” que no corresponden al intervalo inicial obtenemos mejores resultados que respecto a no usar trayectorias. De tal manera, si en un partido procesáramos durante un número determinado de fotogramas las trayectorias “habituales” y después usaramos esa información en fotogramas posteriores (en los cuáles no han intervenido esas trayectorias), deberíamos de obtener mejorías en cuanto a términos de error; que es justamente lo deseable en este sistema.

A simple vista, no existe mucha diferencia en términos de errores a la hora de usar solo trayectorias “habituales” o de usar trayectorias “habituales” + trayectorias “no usuales”. Por lo tanto será preferible usar solo trayectorias “habituales”, ya que el sistema se sobrecargará menos (intenvendrá un número menor de trayectorias y el tiempo de computo será por tanto menor). No obstante si atendemos a la Figura 3.16 podemos ver que para 1000 fotogramas el error suma(error formado por la suma de los errores en los 5 intervalos usados para obtener las trayectorias para distintos valor de umbral_tray) es menor en el caso de solo usar trayectorias “habituales”

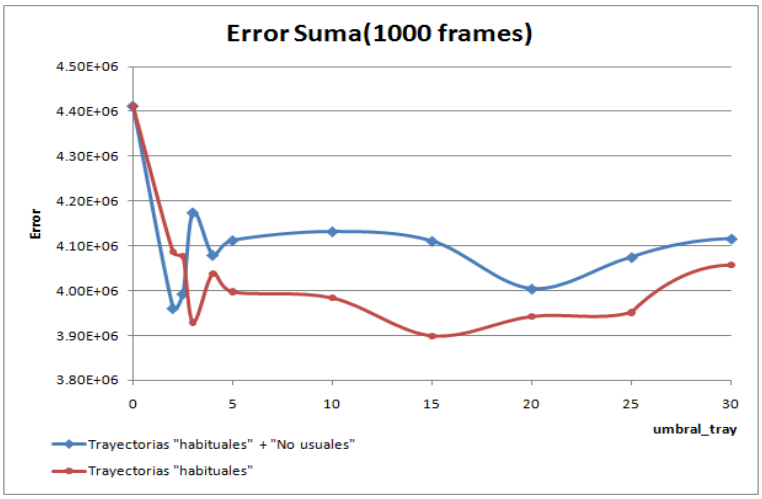

Figura 3.16: Gráfica que muestra el error suma(error formado por la suma de los errores en los 5 intervalos usados para obtener las trayectorias para distintos valor de umbral_tray) para los casos de trayectorias globales: trayectorias “habituales” y trayectorias “habituales”+“no usuales”. La simulación ha sido realizada para los primeros 1000 frames

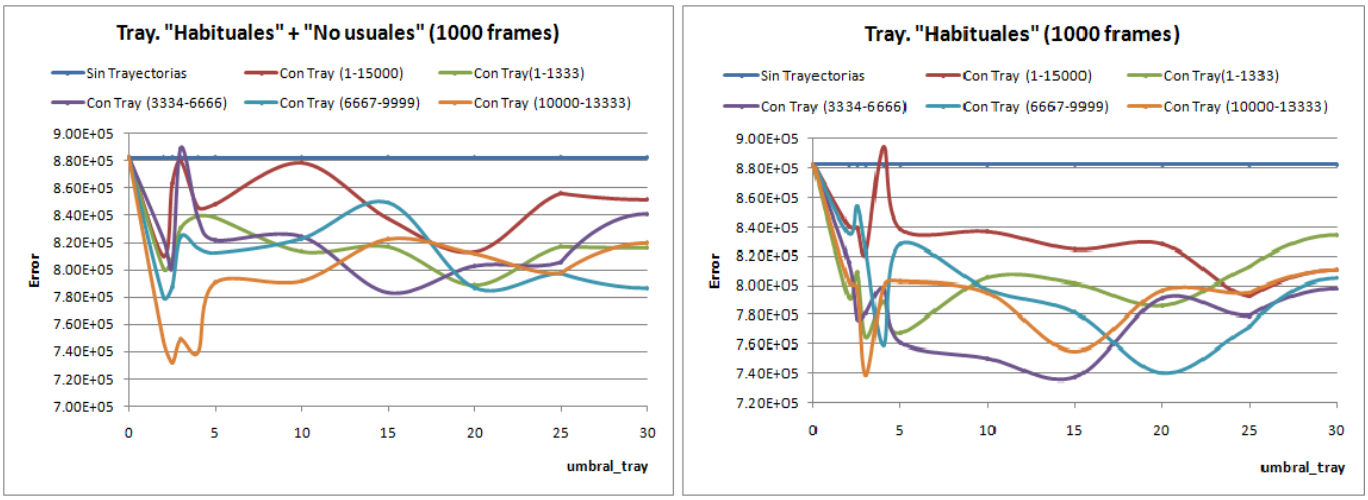

Figura 3.17: Gráfica que muestra el error para distintos valores de umbral_tray para el caso de trayectorias globales: trayectorias “habituales” y trayectorias “habituales”+“no usuales”. En ambos casos la simulación ha sido realizada para los primeros 1000 frames.

Respecto a usar el método 1 o el método 2, parece no haber tampoco mucha diferencia en cuanto a términos de errores. En caso de usar el método 1, a priori no hay un umbral_tray óptimo para el cuál se obtengan mejores resultados que para el resto, tal y como podemos ver en la Figura 3.17

3.3. Resultados obtenidos con el sistema de seguimiento MHT

En primer lugar analizaremos el error obtenido para el sistema inicial, el cuál trabaja solo con la información de las medidas y no con la de las trayectorias. Para ello hemos realizado varias simulaciones para los 200 y para los 1000 primeros fotogramas, modificando en cada una de ellas varios parámetros:

- Por una parte disponemos del umbral que está relacionado con la distancia de Mahalanobis entre las predicciones de estado y la medidas asociadas a éstas; al cuál llamaremos umbral_medida. Si las medidas disponen de una distancia inferior a dicho umbral, en tal caso su información será tenida en cuenta.
- Por otra parte disponemos del nº hipótesis máximo por jugador a tener en cuenta en cada fotograma.
- Además disponemos de otro umbral, el cuál está relacionado con la distancia euclídea entre la posición del estado anterior de un jugador y la posición del estado estimado del mismo jugador para la misma hipótesis; al cuál llamaremos umbral_antest. En caso de que esa distancia sea menor que el umbral, la posición del estado estimado del tracker se actualizará, en caso contrario la posición del estado estimado será igual a la posición estado anterior en dicho fotograma.

Como primera aproximación realizaremos simulaciones cortas(de 200 fotogramas), pero en general solo vamos a mostrar las simulaciones largas(de 1000 fotogramas). No obstante en el apartado de resultados de MHT, si es necesario e interesante comentaremos algún aspecto a resaltar en las simulaciones cortas. Podemos ver el error obtenido para distintos valores de los 3 parámetros anteriores en la Figura 3.18.

Se puede observar que al aumentar el nº de hipótesis máximo por jugador, en general, el error tiende a aumentar. Esto se debe, posiblemente, a que al disminuir el número de hipótesis tenemos en cuenta las hipótesis más probables y eliminamos aquellas peores, de modo que el algoritmo de seguimiento no tienda a seguir las medidas asociadas a las hipótesis más malas y que posiblemente sean equívocas.

Por otra parte el error tiende a ser menor en las simulaciones realizadas con el umbral_antest máximo y el umbral_medida de 5.99. De tal manera, fijaremos para el resto de las simulaciones los parámetros:

- Nº Hipótesis máxima: 2.
- umbral_antest: max.
- umbral_medida: 5.99.

1000 frames		umbral_medida			
NºHipótesis	umbral_antest	4	5.99	12	
max por jug	max	1.2272E+06	1.2246E+06	1.3909E+06	
5 por jug		1.0414E+06	1.1643E+06	1.2262E+06	1.3627E+07
3 por jug		1.0810E+06	1.0633E+06	1.1580E+06	
2 por jug		9.5912E+05	9.6546E+05	1.1251E+06	
max por jug	100	1.0958E+06	1.1607E+06	1.4460E+06	
5 por jug		1.1381E+06	9.8647E+05	1.2416E+06	1.3981E+07
3 por jug		1.1337E+06	9.6370E+05	1.3547E+06	
2 por jug		1.0996E+06	9.3536E+05	1.4252E+06	
max por jug	50	1.2199E+06	1.2908E+06	1.3498E+06	
5 por jug		1.2039E+06	1.2474E+06	1.3582E+06	1.4919E+07
3 por jug		1.3272E+06	1.1689E+06	1.2624E+06	
2 por jug		1.1204E+06	1.0609E+06	1.3089E+06	
max por jug	30	1.2888E+06	1.3103E+06	1.3687E+06	
5 por jug		1.3606E+06	1.1440E+06	1.1968E+06	1.4681E+07
3 por jug		1.1584E+06	1.0590E+06	1.3665E+06	
2 por jug		1.0673E+06	1.0058E+06	1.3547E+06	
		1.8522E+07	1.7751E+07	2.0934E+07	

Figura 3.18: Tabla que muestra el error obtenido en el sistema inicial para distintos valores de umbral_medida, nºhipótesis máxima por jugador y umbral_antest, para los 1000 primeros fotogramas. Los valores que vemos a la derecha de la tabla están asoaciados a la suma total del error obtenido para los distintos valores de umbral_antest. Los valores que vemos debajo de la tabla están asoaciados a la suma total del error obtenido para los distintos valores de umbral_medida.

Una vez hemos fijado los parámetros antes mencionados, pasaremos a realizar simulaciones de 1000 fotogramas para el sistema el cuál ya está realimentado con la información de las "trayectorias comunes". Dichas simulaciones serán realizadas para distintos valores de umbral_tray (2.5, 5, 10 20). El umbral umbral_tray estará asociado a la distancia euclídea entre la posición del estado anterior del jugador cuyo estado queremos estimar y el punto más cercano de cada trayectoria (que se encuentre en el mismo cuadrante que el jugador) respecto al anterior. En caso de que esa distancia sea menor a umbral_tray, la información de la trayectoria pasará por un segundo filtrado.

Dicho filtrado consistirá en otro umbral, el cuál está relacionado con la distancia de Mahalanobis entre el vector formado por los dos puntos más cercanos de cada trayectoria respecto al estado anterior (trasladado al estado anterior) y la predicción de estado. En caso de que dicha distancia sea menor al umbral mencionado, la información de dicha trayectoria será tenida en cuenta. En el caso de este umbral usaremos el mismo valor que hemos usado para umbral_medida. Podemos ver los resultados obtenidos en la Figura 3.19.

En principio podemos ver que los resultados empeoran respecto a no usar la información de las trayectorias. Esto es debido a que en cada fotograma, para cada hipótesis de cada tracker, las trayectorias suelen ganarle el terreno a las medidas, debido a que normalmente para cada tracker, el número de trayectorias es mucho mayor que el número de medidas, además de que las trayectorias suelen disponer de una distancia más pequeña que en el caso de las medidas, respecto al estado predecido.

1000 frames	Con trayectorias		Sin trayectorias
umbral_tray	Siempre	Nº Medidas = 0	
2.5	1.1935E+06	9.5583E+05	9.6546E+05
5	1.2234E+06	8.7328E+05	9.6546E+05
10	1.3165E+06	8.2754E+05	9.6546E+05
20	1.1979E+06	8.4932E+05	9.6546E+05

Figura 3.19: Tabla que muestra el error obtenido en el sistema modificado(alimentado con la info. de las trayectorias) para distintos valores de umbral_tray, para el caso en el que siempre se usa la información de las trayectorias (siempre que no se superen umbral_tray y umbral_medida) y para el caso en que solo se usa dicha información cuando no hay medidas en ninguna de las tres cámaras (para los 1000 primeros fotogramas).

Por lo tanto, igual que en el caso del algoritmo de seguimiento MSUKF, usaremos tan solo la información de las trayectorias cuando no haya medidas en ninguna de las 3 cámaras. De tal manera podemos ver como el error disminuye respecto al caso de usar trayectorias en todo momento (siempre que no superen los 2 umbrales comentados anteriormente). Además en ocasiones el error disminuye respecto al caso de no usar trayectorias.

Una vez hemos fijado todos los umbrales en el sistema de seguimiento MHT, al igual que en el caso del MSUKF realizaremos simulaciones de 1000 fotogramas para distintos valores de umbral_tray. Utilizaremos dos conjuntos de trayectorias “habituales” diferentes:

- Trayectorias independientes
  En la fase del procesamiento de trayectorias por cuadrante procesaremos las trayectorias de forma individual para cada jugador.
- Trayectorias globales
  En la fase del procesamiento de trayectorias por cuadrante procesaremos las trayectorias de forma conjunta para todos los jugadores.

Por otra parte, además de trabajar solo con trayectorias “habituales”, también analizaremos los resultados obtenidos al usar trayectorias “no usuales” junto con las trayectorias “habituales”, como hacíamos en el algoritmo MSUKF. En el caso del MHT, de los dos métodos usados en el MSUKF, solo trabajaremos con el primer método, puesto que no aporta muchas mejoras respecto al segundo método, además de que es más dificil de implementar mediante el sistema MHT.

TRAYECTORIAS GLOBALES(TRAYECTORIAS “HABITUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	1.0194E+06	9.9503E+05	9.8092E+05	9.8321E+05
	3334-6666	1.0551E+06	1.1046E+06	1.0281E+06	9.4519E+05
	6667-9999	9.9711E+05	1.0002E+06	9.8332E+05	9.3957E+05
	10000-13333	1.0579E+06	1.0756E+06	1.0894E+06	1.0525E+06
	Todas	9.6300E+05	9.5583E+05	8.8875E+05	9.3567E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	9.9250E+05	9.2829E+05	8.7945E+05	8.9484E+05
	3334-6666	9.8705E+05	9.4516E+05	9.8720E+05	9.8100E+05
	6667-9999	1.0599E+06	1.0426E+06	9.1387E+05	9.3269E+05
	10000-13333	1.0767E+06	1.0144E+06	9.3195E+05	9.0362E+05
	Todas	8.7328E+05	8.2754E+05	8.8551E+05	8.4932E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	9.6229E+05	1.0134E+06	9.1102E+05
	3334-6666	9.8332E+05	9.4652E+05	8.8385E+05
	6667-9999	9.8097E+05	9.7447E+05	8.7415E+05
	10000-13333	9.2528E+05	9.3914E+05	8.8380E+05
	Todas	9.3811E+05	1.0209E+06	1.0916E+06
	umbral_tray	**25**	**30**	**10000**

Figura 3.20: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS GLOBALES(T.“HABITUALES” + T.“NO USUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	9.5914E+05	1.0101E+06	9.8791E+05	8.7752E+05
	3334-6666	1.0318E+06	9.5905E+05	9.5775E+05	9.7016E+05
	6667-9999	1.0352E+06	1.0468E+06	9.8146E+05	9.3932E+05
	10000-13333	9.7442E+05	9.8102E+05	8.7861E+05	8.6839E+05
	Todas	9.1800E+05	8.9916E+05	9.3105E+05	9.4314E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	9.6979E+05	9.1727E+05	9.8602E+05	9.9474E+05
	3334-6666	9.4594E+05	9.4882E+05	9.1543E+05	9.5838E+05
	6667-9999	9.8144E+05	8.6296E+05	9.6904E+05	8.8347E+05
	10000-13333	9.4742E+05	9.8674E+05	9.3240E+05	9.0115E+05
	Todas	9.0171E+05	9.9309E+05	9.6218E+05	1.0302E+06
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	9.2252E+05	9.3778E+05	9.4517E+05
	3334-6666	9.5035E+05	8.6389E+05	9.7819E+05
	6667-9999	9.8162E+05	9.6272E+05	9.3404E+05
	10000-13333	8.9763E+05	9.8241E+05	1.0380E+06
	Todas	1.0206E+06	9.5786E+05	1.0802E+06
	umbral_tray	**25**	**30**	**10000**

Figura 3.21: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS INDEPENDIENTES(T.“HABITUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	1.0685E+06	1.0723E+06	1.0522E+06	1.0487E+06
	3334-6666	1.0722E+06	1.0777E+06	1.0777E+06	1.0777E+06
	6667-9999	1.0670E+06	1.0670E+06	1.0670E+06	1.0792E+06
	10000-13333	1.0666E+06	1.0667E+06	1.0667E+06	1.0636E+06
	Todas	1.0880E+06	1.0769E+06	1.1118E+06	1.1104E+06
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	1.0553E+06	1.0529E+06	1.0505E+06	1.0859E+06
	3334-6666	1.0785E+06	1.0618E+06	1.0436E+06	1.0472E+06
	6667-9999	1.1380E+06	1.0661E+06	1.0740E+06	1.0614E+06
	10000-13333	1.0634E+06	1.0615E+06	1.0634E+06	1.0651E+06
	Todas	1.1354E+06	1.0086E+06	9.4594E+05	9.3762E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	1.0593E+06	1.0445E+06	1.0336E+06
	3334-6666	1.0445E+06	1.0491E+06	9.8160E+05
	6667-9999	1.0651E+06	1.0782E+06	1.0641E+06
	10000-13333	1.0691E+06	1.0776E+06	1.0787E+06
	Todas	9.2558E+05	9.1291E+05	9.9364E+05
	umbral_tray	**25**	**30**	**10000**

Figura 3.22: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

TRAYECTORIAS INDEPENDIENTES(T.“HABITUALES” + T.“NO USUALES”): MÉTODO 1

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	1.1036E+06	1.1078E+06	1.0476E+06	1.0367E+06
	3334-6666	1.0058E+06	1.0280E+06	1.0222E+06	1.0232E+06
	6667-9999	1.1203E+06	1.1303E+06	1.1141E+06	1.0858E+06
	10000-13333	1.1594E+06	1.1268E+06	1.1223E+06	1.1773E+06
	Todas	9.8396E+05	1.0008E+06	9.8494E+05	9.6732E+05
	umbral_tray	**2**	**2.5**	**3**	**4**

1000 frames		Error			
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	1.0524E+06	9.8139E+05	9.7820E+05	9.7997E+05
	3334-6666	1.0312E+06	9.8665E+05	1.0649E+06	9.5908E+05
	6667-9999	1.0627E+06	9.9460E+05	9.8403E+05	9.9080E+05
	10000-13333	1.1123E+06	1.0951E+06	1.0462E+06	1.0668E+06
	Todas	9.5992E+05	1.0057E+06	1.0329E+06	9.4580E+05
	umbral_tray	**5**	**10**	**15**	**20**

1000 frames		Error		
Sin trayectorias	-	9.6546E+05	9.6546E+05	9.6546E+05
Con trayectorias	1-3333	8.9624E+05	9.2744E+05	8.6965E+05
	3334-6666	1.0341E+06	9.6220E+05	9.6826E+05
	6667-9999	9.6483E+05	9.6326E+05	1.0040E+06
	10000-13333	1.0047E+06	9.9749E+05	9.8606E+05
	Todas	1.0209E+06	9.5392E+05	9.1256E+05
	umbral_tray	**25**	**30**	**10000**

Figura 3.23: Tabla que muestra el error obtenido en el sistema modificado para los valores de umbral_tray: 2, 2.5, 3, 4, 5, 10, 15, 20, 25, 30 y 10000 (para 1000 fotogramas).

Nota aclaratoria respecto a las gráficas:

- En los casos en los que el error está sombreado de color verde significa que dicho error, el cuál ha sido obtenido mediante el sistema modificado(sistema realimentado con la información de las trayectorias), es menor que el obtenido mediante el sistema original(sistema en el que no se usa la información de las trayectorias).
- En los casos en los que el error está sombreado de color amarillo significa que dicho error, el cuál ha sido obtenido mediante el sistema modificado, es algo mayor que el obtenido mediante el sistema original, pero por una diferencia pequeña.
- En los casos en los que el error no está sombreado significa que dicho error, el cuál ha sido obtenido mediante el sistema modificado, es bastante mayor que el obtenido mediante el sistema original.

3.3.1. Conclusiones(MHT)

En general obtenemos mejores resultados usando trayectorias globales que usando trayectorias independientes. Además, al revés que ocurría en el caso del MSUKF, para una simulación más larga obtenemos peores resultados que para una simulación más corta. Podemos ver esto en la Figura 3.24.

En el caso de usar trayectorias globales solo “habituales” podemos ver que la mejoría respecto a no usar trayectorias se producirá sobre todo cuando usamos la información de las trayectorias asociadas a la primera parte entera del partido(15000 frames), tal y como podemos ver en la Figura 3.25. Usando la información de intervalos de 3333 frames a veces se producen mejorías respecto a no usar la información de las trayectorias pero no se producen casi siempre como cuando usamos la información de toda la primera parte.

Usando trayectorias globales “habituales” más “no usuales” podemos ver que la mejoría no se produce para las trayectorias asociadas a un intervalo de fotogramas determinado como pasaba con las globales(ver Figura 3.26).

En general no hay mucha diferencia en cuanto a términos de error entre usar la información asociada a las trayectorias globales “habituales” y usar la información asociada a las trayectorias globales “habituales” más “no usuales”. A partir del ErrorSuma(error formado por la suma de los errores obtenidos cuando se usa la info. de los distintos grupos de trayectorias para cada umbral_tray) obtenido para ambos conjuntos de trayectorias, podemos ver que para un umbral_tray entre 15 y 20 parece ser mejor usar solo trayectorias “habituales”, además que es cuando menor ErrorSuma existe. Podemos ver una muestra de esto en la Figura 3.27.

		Nº frames = 200	
Trayectorias Globales		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	40	15
Tray.Hab+No Usuales		40	15
Total		80	30

		Nº frames = 1000	
Trayectorias Globales		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	27	28
Tray.Hab+No Usuales		32	23
Total		59	51

		Nº frames = 200	
Trayectorias Independientes		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	13	42
Tray.Hab+No Usuales		45	10
Total		58	52

		Nº frames = 1000	
Trayectorias Independientes		Mejora(casos)	Empeoramiento(casos)
Tray.Habituales	Método 1	4	51
Tray.Hab+No Usuales		11	44
Total		15	95

Figura 3.24: Tabla que muestra el número de casos en los que hay mejora y empeoramiento respecto a no usar la información de las trayectorias. En dicha tabla se muestra el caso de las trayectorias globales y de las independientes para los 200 y para los 1000 primeros frames.

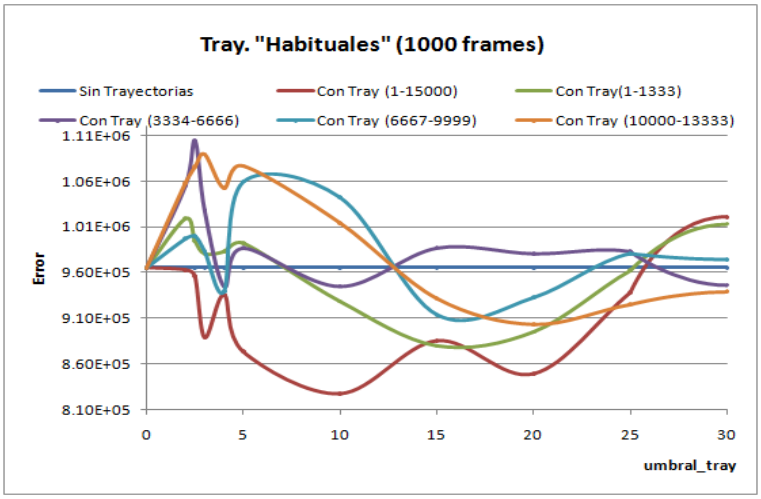

Figura 3.25: Gráfica que muestra el error obtenido en el caso de usar trayectorias globales solo “habituales” usando la información de los diferentes intervalos de trayectorias.

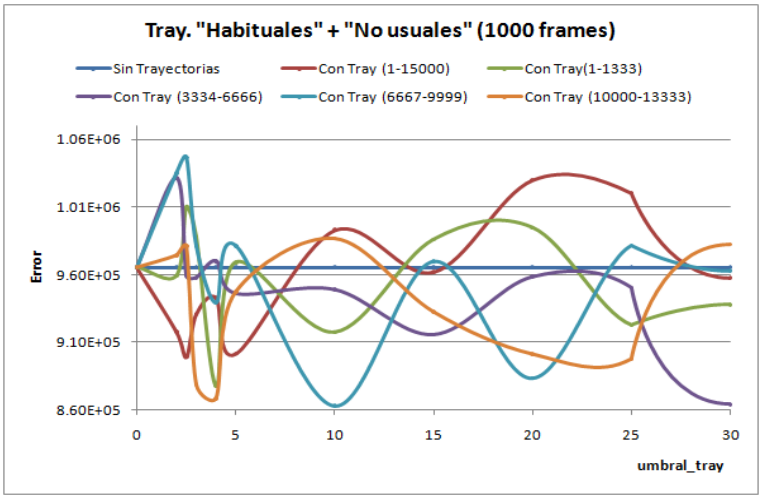

Figura 3.26: Gráfica que muestra el error obtenido en el caso de usar trayectorias globales “habituales” más “no usuales” usando la información de los diferentes intervalos de trayectorias.

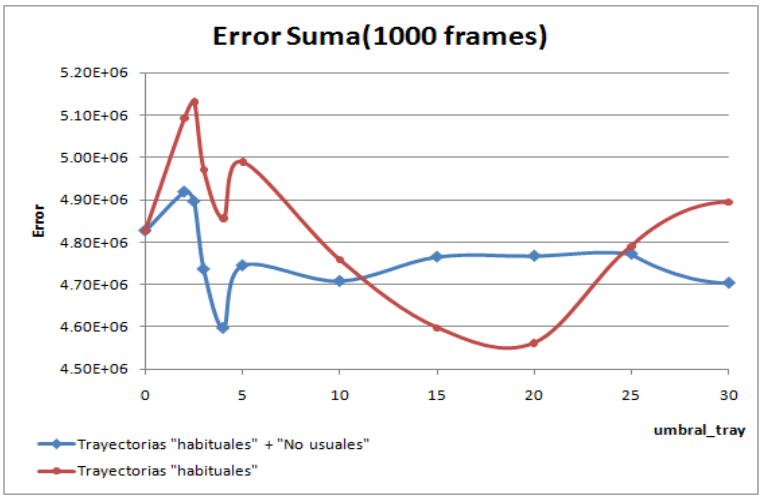

Figura 3.27: Gráfica que muestra el ErrorSuma obtenido en el caso de usar trayectorias globales “habituales” y en el caso de usar trayectorias globales “habituales” más “no usuales”. Se puede ver que para un umbral_tray perteneciente entre 15 y 20 parece haber mejoría en el caso de solo usar trayectorias “habituales” respecto al segundo caso.

Trayectorias globales					
1000 frames		Error	Nº trayectorias	Error	Nº trayectorias
Sin trayectorias	-	9.6546E+05	0	9.6546E+05	0
Con trayectorias	1-3333	8.7945E+05	58450	1.0134E+06	115981
	3334-6666	9.8720E+05	53407	9.4652E+05	105749
	6667-9999	9.1387E+05	59953	9.7447E+05	118114
	10000-13333	9.3195E+05	55593	9.3914E+05	112088
	Todas	8.8551E+05	199044	1.0209E+06	376939
	umbral_tray	15		30	

Trayectorias independientes					
1000 frames		Error	Nº trayectorias	Error	Nº trayectorias
Sin trayectorias	-	9.6546E+05	0	9.6546E+05	0
Con trayectorias	1-3333	1.0505E+06	29456	1.0445E+06	50355
	3334-6666	1.0436E+06	23508	1.0491E+06	44278
	6667-9999	1.0740E+06	21006	1.0782E+06	39419
	10000-13333	1.0634E+06	20783	1.0776E+06	38056
	Todas	9.4594E+05	72243	9.1291E+05	136786
	umbral_tray	15		30	

Figura 3.28: Tabla que muestra el número de trayectorias que intervienen para distintos valores de umbral_tray para los casos de trayectorias globales “habituales” y trayectorias independientes “habituales” (para 1000 frames). Se puede ver que al aumentar el valor de umbral_tray entran más trayectorias en juego, como es de esperar. Por otra parte para el mismo valor de umbral_tray se puede ver que interviene un número mayor de trayectorias en el caso de las trayectorias globales.

En el caso de las trayectorias independientes se puede observar que para valores de umbral_tray bajos los errores obtenidos son mayores que en el caso de no usar trayectorias. Sin embargo para valores de umbral_tray más altos, en ocasiones hay mejora respecto a el caso de no usar trayectorias. Esto se debe, posiblemente, a que en los casos de umbral_tray bajo, disponemos de muy pocas trayectorias para cada jugador por cuadrante y en vez de obtener mejoría llegamos incluso a obtener empeoramiento. Podemos ver una muestra del número de trayectorias que entran en juego para los casos de trayectorias globales “habituales” y de trayectorias independientes “habituales” para distintos valores de umbral_tray en la Figura 3.28. Igual que pasaba con las trayectorias globales, en caso de usar las trayectorias independientes “habituales” podemos ver que la mejoría respecto a no usar trayectorias se producirá sobre todo cuando usamos la información de las trayectorias asociadas a la primera parte entera del partido (15000 frames).

Con el sistema MHT no se han obtenido unos resultados tan satisfactorios como en el caso del sistema MSUKF. Esto puede ser debido a varios aspectos. Por un lado es posible que no hayamos conseguido fijar de la mejor forma posible los parámetros iniciales, puesto que el sistema MHT dispone de un amplio abanico de estos, a diferencia del MSUKF; el cuál dispone de un número menor de parámetros. Por otro lado, cabe pensar que el sistema MHT a pesar de que puede disponer de varias hipótesis por tracker, dichas hipótesis se basan cada una en un filtro de Kalman lineal; y sin embargo, el sistema MSUKF a pesar de disponer de una sola hipótesis por tracker, se basa en un filtro de Kalman no lineal(UKF).

CAPÍTULO 4

CONCLUSIONES Y SUGERENCIAS PARA FUTURAS INVESTIGACIONES

4.1. Conclusiones

Tal y como hemos podido ver en las diferentes simulaciones realizadas hemos conseguido que nuestro sistema de seguimiento consiga mejorar a base de realimentarlo con la información de las trayectorias "habituales". En nuestras simulaciones se trabajaba con los primeros fotogramas de un partido -1000 fotogramas, teniendo en cuenta que una parte entera de un partido eran 15000- usando la información de las trayectorias "habituales" obtenida para distintos intervalos (Recordemos que habíamos obtenido las trayectorias "habituales" para 5 intervalos: 1-1333 , 3334-6666, 6667-9999, 10000-13333, 1-15000).

Se tendería a pensar que si había mejoría en nuestro sistema se daría usando las trayectorias "habituales" del intervalo 1-1333 (ya que realizamos nuestras simulaciones para el intervalo 1-1000) o del intervalo 1-15000.

En el caso del algoritmo de seguimiento MHT, si usamos trayectorias globales la mejora se suele producir sobre todo cuando usamos la información de las trayectorias "habituales" para el intervalo 1-15000.

Sin embargo, en el sistema MSUKF, cuando se trabaja con trayectorias globales se obtiene una mejoría en casi todos los casos (hasta de un 16 por cierto en términos de error), de manera que incluso con trayectorias que no pertenecen al intervalo en el que se está realizando el seguimiento, se consiguen mejorar los resultados respecto a no usar la información de las trayectorias. Esto es bastante esperanzador, ya que podíamos aplicarlo en un partido para poder ir mejorando nuestro sistema en tiempo real.

A pesar de que pudiesemos esperar lo contrario, hemos obtenido mejores resultados con el sistema MSUKF que con el sistema MHT. Esto puede ser debido a varios aspectos. Por un lado es posible que no hayamos conseguido fijar de la mejor forma posible los parámetros iniciales, puesto que el sistema MHT dispone de un amplio abanico de estos, a diferencia del MSUKF; el cuál dispone de un número

MSUKF			Nº frames (info. Trayectorias)	Tiempo simulación (seg.)
Sin Trayectorias			-	297.79
Con Trayectorias	Tray. Globales	Tray. Habituales	3333	439.94
			15000	1002.87
		Tray. Habituales + No usuales	3333	765.05
			15000	1396.26
	Tray. Ind.	Tray. Habituales	3333	297.79
			15000	348.20
		Tray. Habituales + No usuales	3333	349.83
			15000	501.67

MHT			Nº frames (info. Trayectorias)	Tiempo simulación (seg.)
Sin Trayectorias			-	303.80
Con Trayectorias	Tray. Globales	Tray. Habituales	3333	512.76
			15000	1040.40
		Tray. Habituales + No usuales	3333	773.40
			15000	1525.25
	Tray. Ind.	Tray. Habituales	3333	337.27
			15000	384.43
		Tray .Habituales + No usuales	3333	419.78
			15000	556.23

Figura 4.1: Tiempos de cómputo obtenidos para diferentes simulaciones para distintos conjuntos de trayectorias "habituales". Dichas simulaciones han sido realizadas, mediante el software matemático Matlab, con un Pentium Dual Core T4400 @2.20GHz 2.20GHz con 4 gigas de memoria RAM. Si pasaramos el código a lenguage c++ se obtendrían tiempos de cómputo bastante más pequeños pero las simulaciones iniciales se realizan con Matlab.

menor de parámetros. Por otro lado, cabe pensar que el sistema MHT a pesar de que puede disponer de varias hipótesis por tracker, dichas hipótesis se basan cada una en un filtro de Kalman lineal; y sin embargo, el sistema MSUKF a pesar de disponer de una sola hipótesis por tracker, se basa en un filtro de Kalman no lineal(UKF).

Por otra parte, debemos atender a los tiempos de cómputo. Tal y como podemos ver en la Figura 4.1, atendiendo al sistema de seguimiento para los 1000 fotogramas de estudio, el tiempo de cómputo suele ser inferior a los 1000 segundos, eso sí, no estamos teniendo en cuenta el tiempo de cómputo del sistema de detección, el cuál desconocemos. Se puede ver que el sistema MHT es más lento que el MSUKF, en este caso no mucho, debido a que en nuestras simulaciones se han tomado como mucho dos hipótesis por jugador. A mayor número de hipótesis más lento irá el sistema. Si comparamos los tiempos de cómputo al usar trayectorias con respecto a no usarlas, se puede ver que dependiendo del intervalo de trayectorias usado los tiempos podrán llegar incluso a triplicarse (en caso de usar trayectorias globales para el intervalo 1-15000). No obstante para los intervalos de 3333 frames (que en nuestro caso son los que realmente interesarían para poder realizar mejoras en tiempo real), los tiempos se multiplican aprox. por un factor 1.5; lo cuál no es muy grave.

4.2. Sugerencias para futuras investigaciones

Los algoritmos de seguimiento planteaban dos grandes problemas en un principio:

- Los cruces y oclusiones que se producen entre jugadores dentro del campo de futbol. En especial en jugadas en las que hay una gran acumulación de personas en una zona del campo (como los saques de esquina o los saques de falta).
- Los momentos en los que no hay medidas en las cámaras y el algoritmo de seguimiento debe de guiarse por sí mismo (por sus predicciones).

Mediante el uso de la información de las trayectorias “habituales” parece que el primer problema es capaz de solucionarse en ocasiones sencillas. Pero en ocasiones complejas como pueden ser un cruce entre jugadores del mismo equipo o una aglomeración de jugadores(en especial este último caso) la cosa se complica.

Por otra parte, el segundo problema parece solucionarse de manera correcta, ya que cuando no hay medidas usamos la información de las trayectorias, de manera que obtenemos un menor error que si dejamos al propio algoritmo de seguimiento que se guie por sus predicciones.

Como sugerencias futuras para paliar de una forma mayor ambos problemas se podrían plantear varios aspectos:

- Disponer de cámaras de una mayor resolución que entre otras cosas fuesen capaces de capturar la silueta de los números de las camisetas de los jugadores dentro de lo posible.
- Disponer de cámaras cenitales o aéreas junto con las cámaras laterales.
- Disponer de un mayor número de cámaras, de manera que tengamos una mayor número de puntos de vista(En el problema actual tan solo se trabajaba con 3 cámaras).
- Obtener las regiones de probabilidad cada jugador (probabilidad de que se encuentre en una zona del campo u otra); y usar dicha información para realimentar nuestro sistema de seguimiento junto con la información de las medidas y de las trayectorias “habituales”.

ANEXO A

ESTADO DEL ARTE

A.1. Introducción

En la actualidad existe un gran interés en aplicar innovaciones tecnológicas en el ámbito deportivo, con varios objetivos, como son el que pueda aumentar el rendimiento de los deportistas(como realizar mejor un movimiento para obtener mejores resultados), el estudio de tácticas por parte de los equipos o el estudio de estadísticas y/o eventos por parte de los entes televisivos(de forma que puedan aportar información extra a los telespectadores). Para ello podemos usar técnicas de *Visión por Computador*. En nuestro caso vamos a aplicarlas en el deporte del futbol, aunque sabemos que el uso de dichas técnicas está presente en otros deportes como: Baloncesto, Hockey hielo, béisbol, tennis, golf, billar, patinaje, etc.

Actualmente se trabaja con sistemas completos de detección y seguimiento de jugadores de futbol que son capaces de detectar, clasificar y seguir a los jugadores; de manera que al final puedan obtener la trayectoria individual de cada uno. También existen sistemas completos que nos permiten analizar partidos enteros de manera que se puedan obtener estadísticas y detectar eventos de la forma más automática posible.

A.2. Detección

A.2.1. Detección de movimiento

Una de las opciones posibles para poder detectar personas a lo largo de una sucesión de imágenes es la detección previa de movimiento en dicha imagen. Cuando se trabaja con cámaras fijas, uno de los métodos más utilizados es el de la sustracción de la imagen respecto al fondo. Sea I_t la imagen en el instante actual t y B_{t-1} el fondo calculado en el instante anterior, serán detectados como movimiento aquellos puntos en los que su diferencia supere un umbral T:

$$I_t(x,y) - B_{t-1}(x,y) > T \tag{A.1}$$

Siendo x,y: coordenadas de la imagen.

Existen varias formas de modelar el fondo:

- A partir de la media.
- A partir de la mediana.
- A partir del histograma de una serie de fotogramas de entrenamiento.

Posteriormente se pueden aplicar distintos tipos de filtrados a dicha detección, de manera que se pueda incluir la información de la vecindad de los píxeles y reducir la influencia del ruido. De esta forma se puede mejorar el resultado final.

La sustracción de la imagen respecto del fondo es un método con el que podemos obtener las regiones de movimiento no sólo en entornos controlados, sino también en entornos complejos como escenarios con niebla o nieve [2]. Algunos autores intentan detectar las distintas partes de las personas detectadas a partir de un análisis del contorno de la detección [3]. Otros autores proponen como mejora la creación de un doble fondo, de manera que uno actúe a corto plazo y el otro actúe a largo plazo [4]. De este modo, el fondo a corto plazo detectará los objetos/figuras que se están moviendo en ese momento, mientras que el fondo a largo plazo detectará aquellos objetos/figuras que han efectuado un movimiento previo pero que en el momento actual se encuentran estáticos.

Independientemente de la forma en que hayamos calculado el fondo, los métodos de detección de movimiento suelen seguir una estructura de este tipo:

1. Modelar el fondo.
2. Inicializar el fondo (Calcular el fondo usando los fotogramas de entrenamiento que deseemos).
3. Mantener el fondo (Adaptarlo a los cambios que haya podido sufrir. Para ello se suele recalcular cada cierto tiempo).
4. Detectar figuras
5. Extraer características de las figuras

A.2.2. Segmentación de la imagen

Una vez se realiza la detección de movimiento puede ser conveniente realizar una segmentación de la figura que se ha detectado como movimiento. La segmentación de color de una imagen es la división de dicha imagen en regiones de colores homogéneos. Los métodos de segmentación más utilizados pueden dividirse en los siguientes grupos:

- Umbralización de histogramas.
- Agrupamiento.
- Crecimiento de regiones.
- Detectores de bordes.
- Modelos físicos.
- Redes neuronales.
- Mezcla de distribuciones.
- Métodos borrosos.

A.2.3. Clasificación de figuras

Por último se etiquetan manualmente algunos ejemplos de figuras, los cuáles serán usados como plantillas. Las nuevas figuras que detectemos serán comparadas con dichas plantillas, de manera que podamos detectar su identidad(en este caso el equipo al que pertenece dicha figura). Algunos autores usan múltiples observaciones en las que se integran varias características para aumentar la fiabilidad de este método [5]. Otros autores lo que hacen es crear un patrón a partir de varias observaciones [6].

A.3. Seguimiento

Actualmente existe un gran número de algoritmos de seguimiento. En nuestro caso nos centraremos en los dos grupos más importantes: los de tipo Kalman y los de tipo Monte Carlo.

A.3.1. Algoritmos de seguimiento tipo Kalman

El Filtro de Kalman [9] es un algoritmo de seguimiento de tipo recursivo que se divide en dos pasos:

1. Predicción de la posición de los objetos que están en movimento.
2. Comparación de la predicción con las medidas obtenidas en la detección(si las hay) y obtención de la estimación de estado para cada tracker. En caso de no existir medidas que puedan provenir de un tracker se tomará como estimación a la posición de su predicción.

El filtro de Kalman original presenta el inconveniente de que solo es capaz de estimar de forma correcta las posiciones en sistemas lineales y gaussianos. Debido a esto, se han desarrollado modificaciones de dicho filtro. Las modificaciones más conocidas son:

- Extended Kalman Filter (EKF): [10]
  En dicho algoritmo se aproxima la distribución de estado por una variable aleatoria gaussiana, la cuál se propaga a través un sistema no lineal de primer orden. Los errores que introduce pueden hacer que el filtro diverja en caso de que el sistema no sea lo suficientemente lineal.
- Unscented Kalman Filter (UKF): [11]
  En dicho algoritmo también se aproxima la distribución de estado por una v.a. gaussiana, solo que en este caso se representa por un conjunto de puntos determinados(los cuáles serán calculados préviamente), llamados puntos sigma. Estos puntos disponen de la media y la covarianza de la distribución gaussiana; y cuando se propagan a través de un sistema no lineal, capturan la media y la covarianza hasta el tercer orden.

Si deseamos realizar el seguimiento de varios objetos a la vez(seguimiento multiobjeto), la complejidad de estos algoritmos residirá en la asociación que se debe de realizar entre las medidas obtenidas para dichos objetos y su posición estimada.

A.3.2. Algoritmos de seguimiento tipo Monte Carlo

El algoritmo de seguimiento tipo Montecarlo, también conocido como Filtro de Partículas(Particle Filter,PC) [12], se basa en un muestreo de las distribuciones. Tiene la ventaja de poder realizar el seguimiento con modelos no lineales y/o no gaussianos, representando la densidad de probabilidad mediante un conjunto de puntos(en el que se introduce un cierto nivel de aleatoriedad), el cuál evoluciona según lo hace la imagen de probabilidad. En lugar de usar medidas discretas como datos de entrada, usaremos imágenes de probabilidad.

La complejidad de estos algoritmos reside en cómo obtener el peso de cada una de las partículas y en cómo realizar la evolución de las muestras usadas en un instante de tiempo para dar lugar a las muestras que van a ser usadas en el instante posterior. Uno de los métodos más utilizados es el Sample Importance Resampling (SIR); el cuál da lugar al algoritmo de Condensación y consiste en tres pasos:

1. Generación de partículas a partir de una función de importancia.
2. Cálculo de pesos.
3. Remuestreo eliminando las partículas con pesos bajos.

Este algoritmo dispone de un inconveniente como es el tiempo de cómputo, ya que es mayor que con otros algoritmos de seguimiento. Por lo tanto será conveniente trabajar con arquitecturas en paralelo para poder trabajar de la forma más cercana al tiempo real.

Por otra parte, si deseamos realizar un seguimiento multiobjeto mediante este tipo de algoritmos, existirán bastantes complejidades, sobre todo si las imágenes de probabilidad son ruidosas, de manera que puedan hacer que los trackers abandonen su trayectoria correcta. No obstante han sido creados diversos sistemas basados en el Filtro de Partículas, los cuáles son capaces de realizar esta tarea de una manera lo más eficiente posible.

A.3.3. Algoritmos multihipótesis

Además de los dos grupos de algoritmos mencionados anteriormente, los cuáles pueden ser usados para realizar un seguimiento multiobjeto, caben destacar los algoritmos de seguimiento multihipótesis(Múltiple Hypothesis Tracking, MHT). En este tipo de algoritmos, en caso de duda a la hora de asociar las medidas con los objetos a seguir, se pospone la decisión, creando de tal manera un árbol de hipótesis en el cuál se van calculando las probabilidades a posteriori. Cuando dichas probabilidades son lo suficientemente altas se toma la decisión. Cada rama del árbol se trata como si fuera un tracker distinto. De tal manera se deberían poder alcanzar buenos resultados en un entorno con medidas falsas.

El mayor inconveniente de este algoritmo es que el número de hipótesis con los que se trabaja suele crecer rápidamente, de manera que es necesario usar técnicas de podado de hipótesis(prunning) que vayan eliminando las menos probables. Aun así, es bastante complejo realizar una buena poda de hipótesis, de manera que el problema pueda ser tratable computacionalmente y que a la vez no se descarten hipótesis válidas.

A su vez resulta compleja la asociación entre medidas e hipótesis cuando el número de hipótesis es elevado. Algunos algoritmos multihipótesis utilizan el algoritmo de Viterbi para obtener las asociaciones de medidas. En dicho algoritmo, a partir de los diagramas de trellis, se puede calcular el camino de mayor probabilidad. Otros algoritmos multihipótesis se basan en un modelo probabilístico, como el algoritmo PMHT [19].

A.3.4. Algoritmos multisensor

Para poder cubrir todo el escenario con poca distorsión y mejorar la resolución de las oclusiones y los cruces se puede trabajar con diversos sensores a la vez. En este trabajo usaremos tres sensores a la

vez(en nuestro caso, serán cámaras), de manera que siempre exista una zona de solapamiento común entre las zonas de visión de dichas cámaras. Cada zona del campo de futbol al menos estará cubierta por dos cámaras. Una vez dispongamos de las medidas en la imagen de cada cámara, habrá que pasar dichas medidas a un plano común mediante una transformación proyectiva. De tal manera, podemos observar algunos trabajos en los que se logra obtener una mayor tasa de acierto en la resolución de cruces y oclusiones respecto a usar una sola cámara [7]. Algunos trabajos previos explican el método usado para trasladar las posiciones de los jugadores detectados en diferentes cámaras a un plano común [8].

Existen algoritmos que usan la información de varias cámaras los cuáles están basados en el filtro de partículas. Dichos algoritmos son capaces de obtener mejores resultados en el caso de oclusiones; pero a cambio, el dedicar un filtro de partículas a cada objeto de cada cámara resulta muy costoso.

En el caso de realizar un seguimiento multiobjeto con varios sensores el problema de asociación de datos se complica aún más, debido a que existe un mayor número de interacciones entre los diferentes objetivos a seguir. Esto será más grave cuando dichos objetivos sean muy parecidos, como ocurre en el caso de jugadores de futbol del mismo equipo.

A.3.5. Algoritmos de seguimiento sin detección previa

Existen diversos algoritmos de seguimiento los cuáles no están basados en una detección previa, sino que realizan el seguimiento de objetos a partir de estadísticas obtenidas directamente de la imagen o a partir de los valores de los píxeles de la misma.

El algoritmo mean-shift se basa en el reconocimiento de patrones en la imagen. En dicho algoritmo se pretende localizar los mínimos de diferencia de histograma entre un patrón de entrada y la imagen. Las funciones de similitud de histograma más usadas son el coeficiente de Bhattacharya o la divergencia de Kullback-Leibler.

También existen otros algoritmos de seguimiento los cuáles tienen como entrada la segmentación de la imagen, ya que la segmentación no llega a ser considerada como detección en sí. En ocasiones se diseña a la vez la segmentación y el seguimiento, bien sea la segmentación basada en color y en textura o solo en color.

ANEXO B

SISTEMA DE DETECCIÓN

B.1. Introducción

A pesar de que en este proyecto no vamos a realizar modificaciones en la parte correspondiente al sistema de detección consideramos importante el describir de forma resumida como funciona, ya que los datos obtenidos a partir de la detección(medidas), al igual que las trayectorias "habituales", sirven de entrada para el sistema de seguimiento.

El objetivo principal de este sistema es que sea capaz de identificar a los jugadores que haya en la imagen en tiempo real y de la forma más automática posible(de forma que se disminuya lo máximo posible la intervención humana). Debido a dicho objetivo y a que las imágenes que se van a usar deben ser de alta resolución, desarrollaremos un sistema el cuál dispondrá de dos modos de funcionamiento: modo "entrenamiento" y modo "funcionamiento en tiempo real". En primer lugar entrenaremos al sistema de manera que los resultados obtenidos con dicho entrenamiento nos sirvan para que se reduzca el tiempo de cómputo en el modo "funcionamiento tiempo real". Dicho sistema de detección se divide en varias etapas, tal y como se puede ver en la Figura B.1.

En primer lugar debemos definir cuál va a ser la región de interés(ROI) dentro de la imagen y calcular un fondo en el modo entrenamiento. Para ello necesitaremos una secuencia de imágenes. Tanto la ROI

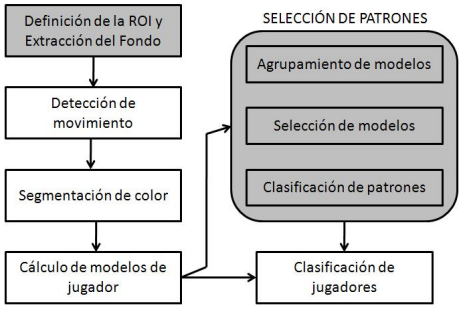

Figura B.1: Etapas del sistema de detección y clasificación de jugadores. Las partes sombreadas solo actúan en el modo entrenamiento. Por otra parte las etapas de "detección de movimiento" y de "segmentación de color" son más simples en el modo "funcionamiento en tiempo real".

como el fondo solo necesitarán ser definidos una vez; no obstante, en caso de que la iluminación cambie será necesario recalcular el fondo. A continuación podremos obtener el movimiento que haya en la imagen restando la imagen actual a la imagen de fondo, dentro de la ROI. Una vez dispongamos de las figuras que están en movimiento pasaremos a realizar la segmentación de éstas y crearemos los modelos de los jugadores. En el modo de entrenamiento, mediante estos modelos generaremos patrones; los cuáles serán usados en el modo funcionamiento en tiempo real para compararlos con los modelos obtenidos en dicho modelo y así poder detectar y clasificar a los jugadores que hayan sido detectados [15].

Debido a que las equipaciones con las que se va a trabajar son diferentes en cada partido y que aún siendo las mismas pueden verse de formas muy diferentes según las condiciones de iluminación o las sombras, se ha partido de tres supuestos básicos:

- La mayoría de los píxeles de la imagen pertenecerán al fondo.
- Los colores de los jugadores son diferentes a los del fondo.
- Los colores de las equipaciones de los jugadores del mismo equipo son iguales.

En tal caso se crearían hasta cinco modelos de equipaciones si también quisieramos seguir ,además de a los jugadores de campo de ambos equipos, a los porteros y al árbitro.

La segmentación del color de las figuras detectadas se ha realizado usando el espacio de color RGB. Se han probado otros espacios de color como HSV, YUV o CIE; pero a pesar de las ventajas que poseen estos espacios en cuanto a su mayor inmunidad a los cambios de luz, ha sido tomado el espacio de color RGB. Normalmente los colores usados en las equipaciones de los jugadores de cada equipo de futbol no se encuentran cerca entre sí en el espacio de color RGB (y sin embargo en el resto de los espacios de color sí) ya que de los espacios mencionados es el que más se acerca a la percepción humana; y por lo tanto es más sencillo diferenciar dichos colores en dicho espacio de color. Existen otros espacios de color más cercanos a la percepción humana, pero plantean otros problemas como una alta necesidad de procesamiento para su conversión.

B.2. Detección de movimiento

B.2.1. Obtención del fondo

En primer lugar obtendremos el fondo para cada cámara. Para ello usaremos un filtrado de mediana sobre la secuencia de imágenes de entrada de cada cámara. Dicho filtrado será aplicado sobre cada

Figura B.2: Fondo obtenido a partir de un filtrado de mediana. A la izquierda disponemos de una de las imágenes asociadas a la secuencia de imágenes de entrada. A la derecha disponemos del fondo obtenido a partir del filtro de mediana sobre dicha secuencia de imágenes.

píxel, ordenando los valores del píxel correspondiente en las distintas imágenes y tomando el valor intermedio. De esta manera, si los jugadores que se encuentran en el campo de juego no permanecen en el mismo lugar más de la mitad de los fotogramas que usamos para el cálculo del fondo, podremos eliminar a dichos jugadores del fondo. En la Figura B.2 podremos ver un ejemplo de como funciona dicho filtrado. El mayor problema que nos podemos encontrar son los cambios en la iluminación. Si la iluminación cambia en la escena el fondo no tendrá los mismos valores de color y puede detectarse que hay movimiento de manera errónea. Para evitar este tipo de errores habrá que recalcular el fondo cada cierto tiempo.

B.2.2. Obtención del área de interes

Por otra parte deberemos de determinar que partes de la imagen van a ser de interés; es decir, aquellas partes en las que aparece el terreno de juego y los jugadores. No nos van a interesar ni las gradas, ni los banquillos, ni las vallas publicitarias, ni el personal que se encuentra alrededor de las cuatro bandas del campo. Estas partes de la imagen pueden hacer que el tiempo de procesamiento aumente y además pueden provocar problemas en el caso de obtener medidas falsas. Por lo tanto, será necesario definir una máscara o región de interés(ROI) para cada cámara que se utilice, tal y como podemos ver en la Figura B.3. La ROI se puede definir de forma manual o de forma automática. Si la cámara es fija es posible usar la región durante todo el partido o incluso para posteriores encuentros si la cámara está instalada en el mismo sitio de forma permanente. En tal caso definiremos la ROI de forma manual.

En caso de que dispongamos de cámaras móviles será interesante obtener la ROI de forma automática. Para diferenciar el terreno de juego del resto de la imagen supondremos que el césped dispondrá del

Figura B.3: Ejemplo de la región de interés usada para una imagen. En este caso la ROI ha sido definida de forma manual.

color predominante. No obstante, suele haber diferencias de color entre las diversas zonas de cesped. Esto se debe a diferentes ángulos, iluminaciones o direcciones del corte del cesped. Para seleccionar el color predominante en la imagen, calcularemos un histograma 3D (respecto al fondo obtenido previamente) con las muestras de todos los píxeles de las imágenes de la secuencia de entrenamiento. Posteriormente realizaremos un filtrado en el que se eliminen las categorías en las que se ha obtenido un número pequeño de pixels, ya que consideraremos que los píxeles del campo se encontrarán en las categorías más pobladas. Después calcularemos la probabilidad a priori para cada una de las categorías del histograma de pertenecer o no al cesped, a partir de la distancia en el espacio RGB entre la posición de la categoría cuya probabilidad queremos estimar y las categorías que han pasado el filtrado. Una vez obtengamos la probabilidad de que un píxel de una determinada categoría pertenezca al fondo, se puede realizar la aproximación de que dicha probabilidad es la misma para todos los píxeles que entrarían en esa categoría. Finalmente podremos transformar la imagen de fondo, previamente calculada, en una imagen de "probabilidad de ser césped". Dicha imagen podrá ser filtrada y umbralizada, obteniendo una zona de césped y una zona descartada, tal y como podemos ver en la Figura B.4

B.2.3. Obtención de las zonas de movimiento

Una vez hemos obtenido la ROI y el fondo nos encargaremos de realizar la detección de movimiento. Para ello realizaremos una resta, dentro del área definida por la ROI, entre cada fotograma y el fondo, tal y como podemos ver en la Figura B.5. Para cada píxel de la imagen obtendremos la detección de movimiento como:

$$D_i = \begin{cases} 1, & \text{si } R_i.abs(I_i - B_i) \geq T_m \\ 0, & \text{en caso contrario} \end{cases} \tag{B.1}$$

Siendo:

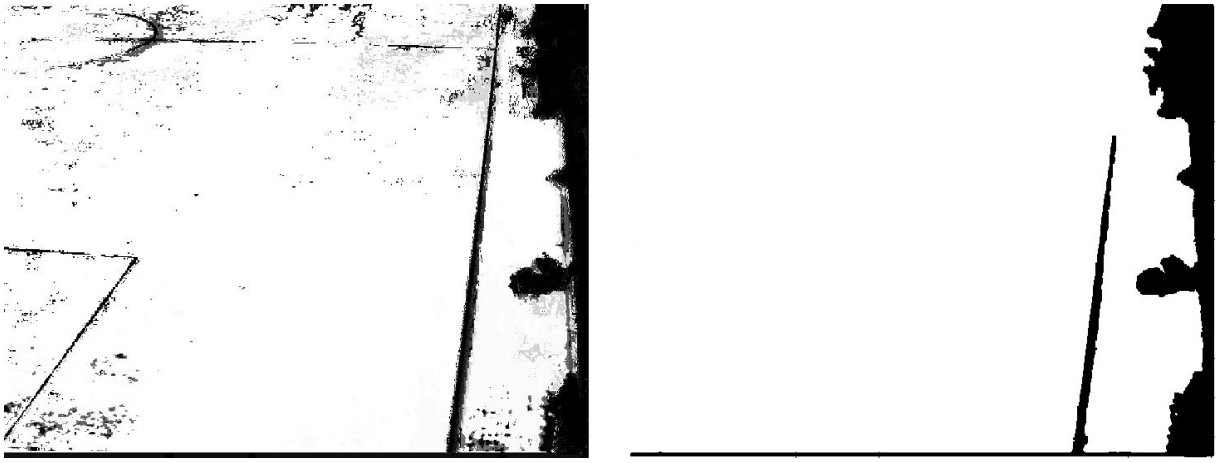

Figura B.4: Imagen de probabilidad de ser césped(izquierda) y filtrado y umbralizado de dicha imagen(derecha) .

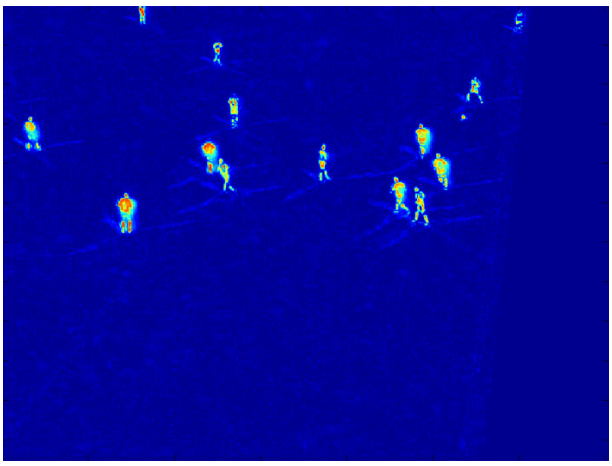

Figura B.5: Detección de movimiento obtenida como resta entre un fotograma y el fondo obtenido préviamente. En este caso la detección de movimiento se realiza sobre la imagen asociada a la figura B.4

- I_i: Píxel i de la imagen actual.
- B_i: Píxel i del fondo actual.
- R_i: Pixel i de la ROI(R_i=1 si el píxel está dentro del área de interés y R_i=0 si el píxel está fuera del área de interés).
- D_i: Detección de movimiento en el píxel i.
- T_m: Umbral de movimiento(T_m>0).

Para mejorar la calidad de la detección aplicaremos dos filtrados. El primero será aplicado antes de realizar la umbralización. Se tratará de un filtro paso bajo en 2D. El tamaño del kernel del filtro se ajustará para que sea de un tamaño similar al de los jugadores. De esa forma serán eliminados otros

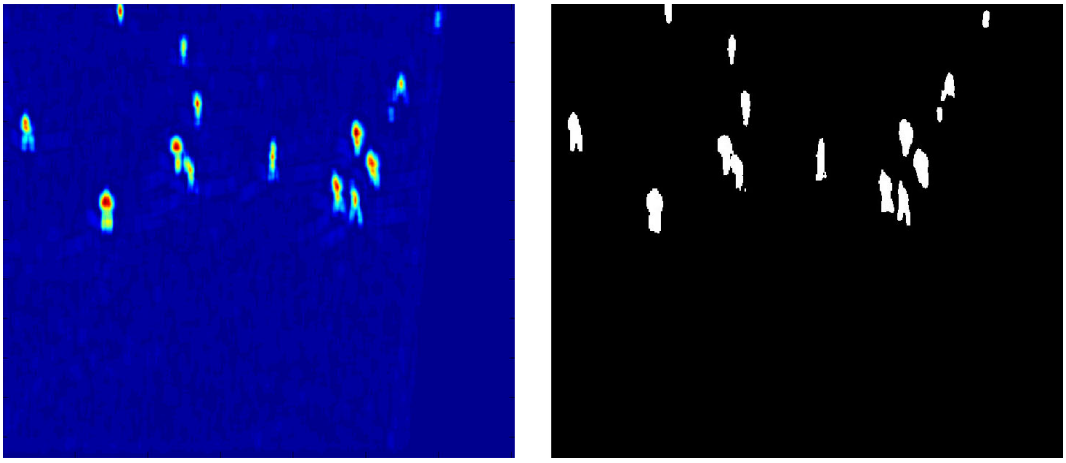

Figura B.6: Detección de movimiento tras el primer filtrado y el umbralizado aplicado a la imagen de la figura B.5.

objetos más pequeños como el balón o las posibles sombras. Tras el primer filtrado se umbralizará la imagen obtenida, usando el umbral de movimiento, de forma que se obtienen los blobs de movimiento(figuras en movimiento) y se etiquetan, tal y como podemos observar en la Figura B.6. El etiquetado consiste en asignar un mismo valor numérico a los píxeles que son vecinos y que a la vez son distintos del resto de los píxeles que no están conectados a ellos. De esta manera los píxeles de un mismo blob dispondrán de una etiqueta exclusiva.

Finalmente se aplicará un segundo filtrado, el cuál tratará de eliminar los píxeles que son demasiado parecidos al color del fondo. Este segundo filtrado definirá de una manera más detallada el contorno de los jugadores y hará que se consiga una detección más precisa(ver Figura B.7). Por otra parte deberemos elegir el umbral de movimiento de forma correcta, ya que si es demasiado bajo aparecerán en la escena zonas en las que se detecta movimiento pero que no pertenecen a los jugadores; y si es demasiado alto los jugadores se detectarán peor (en ocasiones la figura en movimiento detectada asociada a un jugador podrá dividirse y parecer que hay más de un jugador cuando solo hay uno). Podemos ver un ejemplo de esto en la Figura B.8.

B.3. Segmentación de colores de la imagen

B.3.1. Introducción

Una vez son detectadas las zonas de movimiento asociadas a los jugadores, agruparemos los colores asociados a dichas zonas, para poder obtener así los *modelos* de cada jugador. Después, a partir de estos modelos obtendremos los *patrones finales*, los cuáles serán usados en el modo “funcionamiento en

Figura B.7: Detección de movimiento tras el segundo filtrado aplicado a la imagen de la figura B.6.

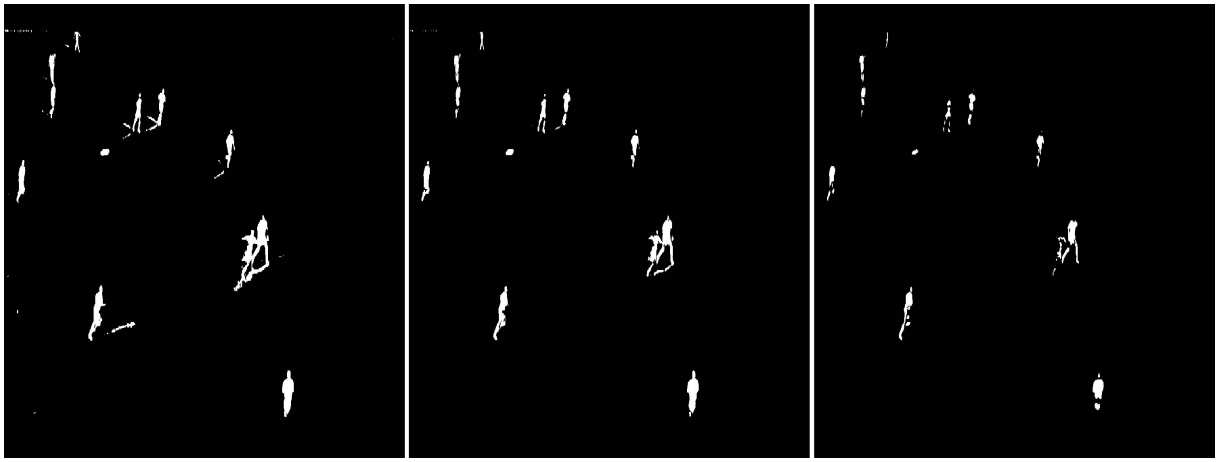

Figura B.8: Umbralización de la imagen para tres umbrales diferentes(de izd.a a dcha, de menor a mayor umbral.

tiempo real" para compararlos con las figuras en movimiento que existan en la imagen. Deberá haber al menos tres tipos de patrones: uno por jugadores de campo de cada equipo y otro para el resto de las figuras que puedan aparecer(como el árbitro o los porteros). Si queremos distinguir a los porteros y al árbitro deberemos disponer al menos de cinco patrones.

B.3.2. Agrupamiento de colores

Introducción

Realizaremos un agrupamiento de colores(a partir de un algoritmo de k-medias) para reducir la complejidad de la información asociada a los jugadores detectados, de manera que los modelos obtenidos sean más simples. Reduciendo el número de colores eliminaremos parte de la aleatoriedad y el ruido de color presente en las figuras, haciendo que los jugadores del mismo equipo sean más parecidos entre sí.

Inicialización de los centroides

En un principio deberemos de inicializar el algoritmo de k-medias, en nuestro caso situaremos los centroides iniciales en los puntos del espacio de color RGB en los que haya mayor frecuencia de color. Para ello comenzaremos calculando un histograma 3D de los píxeles de las zonas de movimiento que han sido detectadas, tal y como podemos ver en la Figura B.9 (utilizaremos los 4 ó 5 bits más significativos de cada canal de color para crear las categorías del histograma).

Antes de aplicar el algoritmo de agrupamiento se deberán de definir dos parámetros: el número máximo de centroides y la distancia mínima entre ellos. En la inicialización se comenzará asignando un centroide al punto en el que se registre el máximo del histograma; es decir, el color que se repite con más frecuencia. El siguiente centroide se asignará al punto con el siguiente valor máximo(siempre y cuando esté a una distancia mayor a la distancia mínima definida del primer centroide). Este proceso se repetirá de forma iterativa hasta que no quedan más centroides por colocar ó hasta que no haya más puntos con valor positivo del histograma fuera del área de acción de cada centroide. Podemos ver gráficamente este proceso en la Figura B.10.

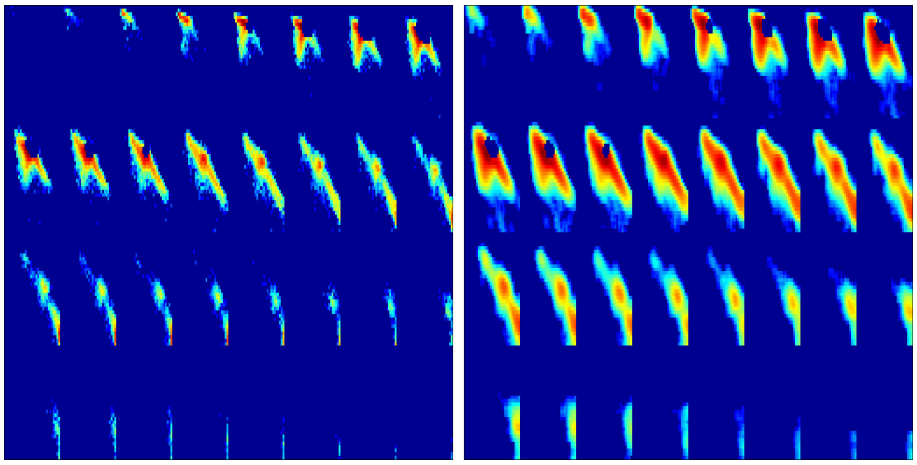

Figura B.9: Histograma 3D de los píxeles de las figuras que han sido detectadas como movimiento. Puesto que se han utilizado 5 bits para cada canal de color, se representarán los 32 histogramas 2D de las componentes R-G para cada uno de los 32 valores de B, yendo de izda. a dcha. y de arriba a abajo conforme crece su valor

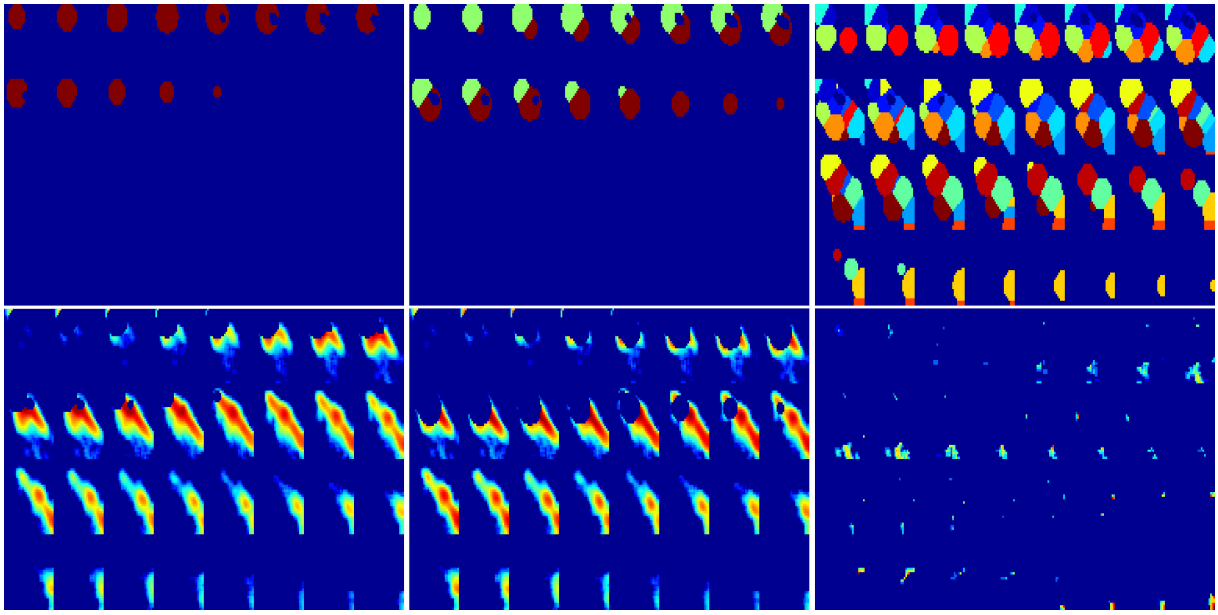

Figura B.10: Muestra de la evolución del algoritmo de inicialización de los centroides. La zona superior muestra la zona ocupada por cada centroide en el espacio de color RGB y la fila inferior muestra el histograma una vez que se han borrado las zonas asignadas. De izquierda a derecha vemos como va evolucionando la inicialización para la primera, la segunda y la última iteración

Aplicación del algoritmo de k-medias(primer agrupamiento o clusterización)

Una vez hemos incializado los centroides se aplicará el algoritmo de agrupamiento de k-medias, el cuál sigue estos pasos:

- Asignación: Se asigna cada elemento al clúster cuya media esté más cercana a dicho elemento.
- Actualización: Se calculan las nuevas medias a partir de los elementos asignados a cada clúster.

Además se han introducido dos modificaciones:

- La primera se basa en usar una distancia máxima de pertenencia al clúster, de manera que los elementos cuyo centroide más cercano esté más lejos de dicha distancia, quedarán sin asignar a ningún cluster.
- La segunda se trata de una modificación en el cálculo de la actualización, en la que se hará una media ponderada de los píxeles que haya en cada categoría del histograma.

La inicialización de las medias, según el método propuesto, obtiene por sí misma unos resultados bastante aceptables de clusterización, de manera que las medias de los grupos no suelen moverse mucho durante el proceso de clusterización de su posición inicial, tal y como podemos ver en la Figura B.11. Si han quedado zonas en el histograma con un valor positivo pero que no estén incluidas en ninguno de los clústeres, es posible incluir un cluster adicional. De tal modo podremos distinguir los colores que están fuera de los clústeres pero que han aparecido en la secuencia de imágenes de entrenamiento (colores minoritarios), respecto de los colores que son nuevos(probablemente ruido o distorsión). Para que la implementación de la segmentación sea rápida usaremos una Look-up table(LUT). Es recomendable que se use la misma LUT para el segundo filtrado de la detección de movimiento y para la segmentación.

Aplicación de un segundo agrupamiento

Mediante la clusterización por color hemos conseguido agrupar colores parecidos, de manera que si aparecen variantes de estos colores en las imágenes, seamos capaces de identificarlos. Aun así puede ocurrir que las equipaciones estén formadas por varios colores o que las variantes de un mismo color se hayan agrupado como clústeres diferentes. Por lo tanto, es interesante realizar una segunda clusterización, de manera que se unan los clústeres de colores que habitualmente aparezcan juntos; es decir, cuando exista bastante correlación entre los píxeles que hay en dos clústeres que se encuentren en el mismo blob.

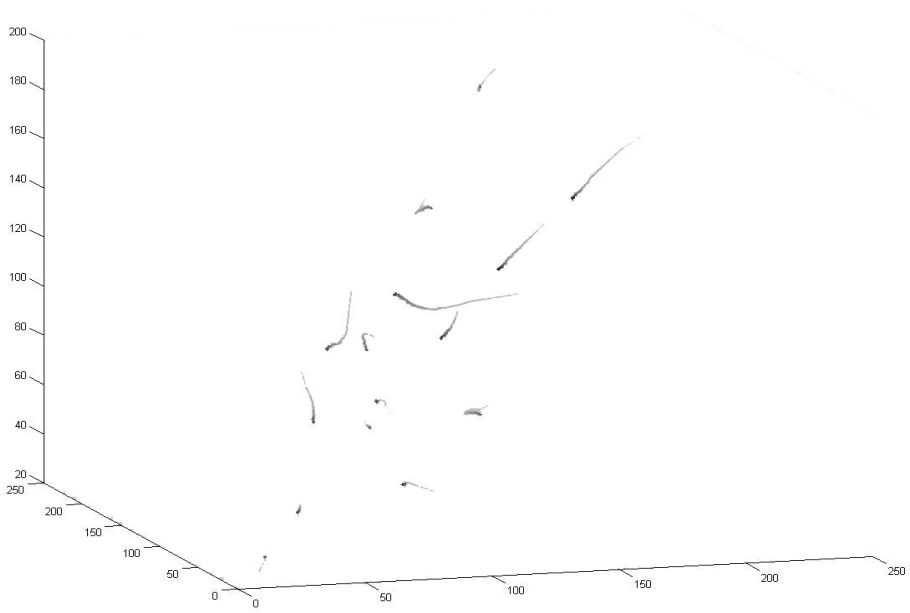

Figura B.11: Trayectoria seguida en el espacio de color RGB por las medias desde su inicialización hasta la última iteración.

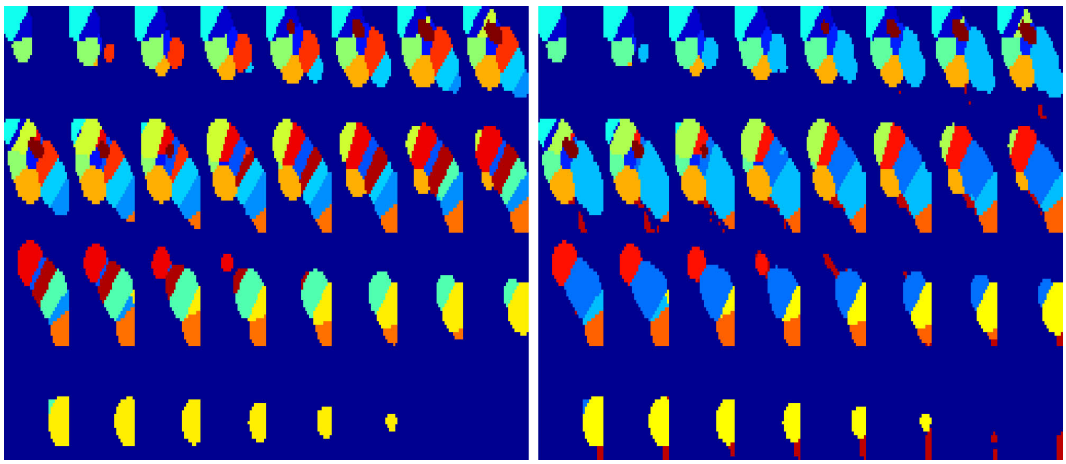

Figura B.12: Representación gráfica de los clústers antes y después de la aplicación del segundo agrupamiento.

De tal forma uniremos aquellos clústeres en los que su función de correlación supere un cierto umbral. Para obtener dicha función de correlación, cada clúster se considerará como una señal discreta cuyo valor en la posición i será el número de píxeles de ese clúster presentes en el blob i. De tal manera se conseguirán mejores resultados en la clasificación de jugadores y además disminuirá el tiempo de cómputo en el modo “funcionamiento en tiempo real”. Podemos ver un ejemplo de como ha funcionado esta segunda clusterización respecto a la inicial en la Figura B.12.

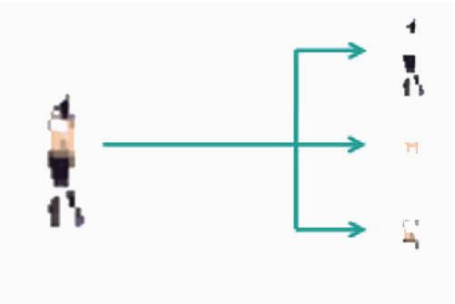

Figura B.13: Ejemplo de división de un blob en varias capas.

B.3.3. Creación del modelo

Una vez las figuras en movimiento han sido detectadas deberemos de crear un modelo para cada jugador. Buscamos que dicho modelo sea representado por la menor cantidad de datos posible de forma que se disminuya el tiempo de cómputo y se aumente la efectividad de la identificación. De todos los modelos que se han probado el que mejores resultados ha dado ha sido uno que está basado en histogramas verticales. Considerando que la información horizontal de los colores de los jugadores es poco importante, este método consigue precísamente reducir la cantidad de información de forma que disminuye el tiempo de cómputo y se aumenta la fiabilidad en la identificación.

En el modo “entrenamiento” el modelo de cada blob se normalizará a un tamaño estándar, de forma que se puedan comparar modelos de jugadores entre sí de una forma más rápida. La obtención del modelo de cada jugador basado en histogramas verticales se divide en 3 fases:

1. En primer lugar, el blob obtenido para cada jugador se dividirá en varias capas, una por cada clúster que contenga dicho blob. De tal forma cada capa podrá ser procesada por separado. Podemos ver un ejemplo en la Figura B.13.
2. En segundo lugar obtendremos los histogramas verticales, creando uno de ellos por cada capa de cada jugador.
3. Por último, se realizará una normalización de dichos histogramas(ver Figura B.14) . En primer lugar serán filtrados para reducir el ruido y la aleatoriedad. Finalmente se remuestrearán para obtener un número fijo de posiciones para cada histograma. De tal manera, podremos representar los modelos de los jugadores como una matriz, tal y como se puede ver en la Figura B.15.

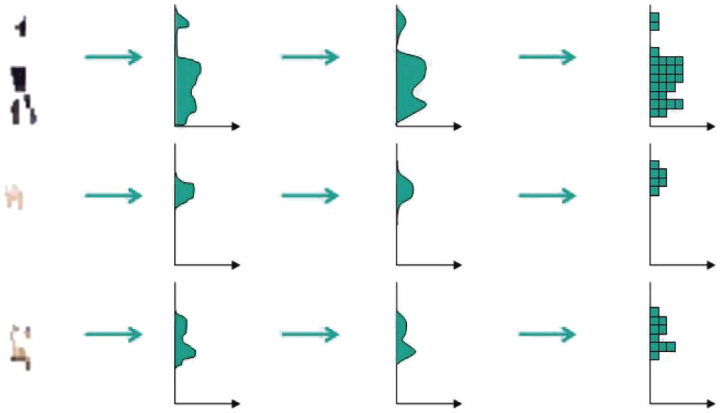

Figura B.14: Obtención del histograma vertical, filtrado y remuestreo para cada una de las capas de la imagen para el blob de la figuraB.13.

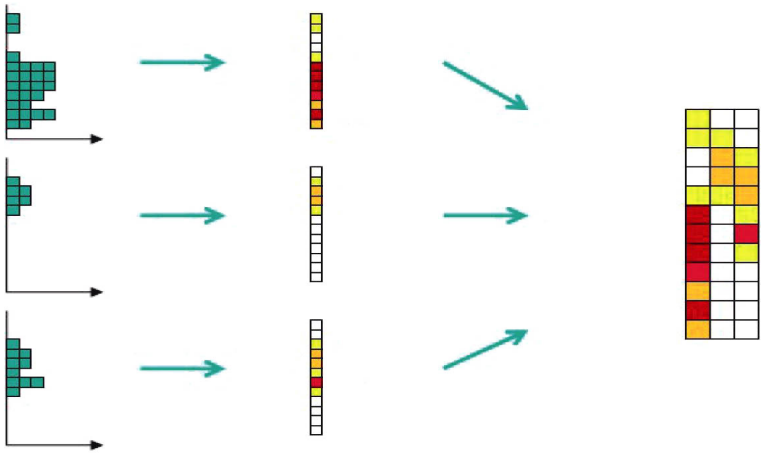

Figura B.15: Matriz que representa el modelo de un jugador asociado al histograma vertical de la figuraB.14.

B.4. Identificación de los jugadores

Una vez ha sido realizada la detección de movimiento y la segmentación y hayamos obtenido los diferentes modelos deberemos obtener los patrones que después serán usados en el modo "funcionamiento en tiempo real". Definiremos patrón como un modelo el cuál es representativo respecto a un grupo de modelos. Para obtener los patrones usaremos un algoritmo de agupamiento jerárquico; el cuál se dividirá en 3 pasos:

1. En primer lugar calcularemos la distancia dos a dos entre cada par de modelos de jugadores. Para ello calcularemos cuál es su similitud, S:

$$S = \frac{\sum_i \sum_j min(A(i,j), B(i,j))}{\sum_i \sum_j max(A(i,j), B(i,j))} \tag{B.2}$$

   donde:

   A,B: son las matrices asociadas a los modelos a comparar.

   i,j: son los elementos de las matrices.

   Cuanto más próximo a 1 sea el valor de la similitud, más parecidos serán entre sí los modelos a comparar; y cuanto más próximo a 0 sea el valor de la similitud, menos parecidos serán entre sí los modelos a comparar. Podemos ver una muestra de esta función en la Figura B.16

2. Para los dos jugadores más cercanos calcularemos su media. Si esta distancia es superior a un umbral, el algoritmo de agrupamiento habrá finalizado.
   Dados 2 modelos de jugadores, atendiendo a sus matrices A y B, la media de ambas matrices M, se definirá como la media elemento a elemento de ambas M = (A+B)/2.

3. Finalmente borraremos a ambos jugadores de la lista, incluiremos su media y volveremos al primer paso.

La mayor complejidad de este algoritmo es saber cúando ha de terminarse el agrupamiento; es decir, establecer un límite a partir del cuál no deberían seguir fusionándose más modelos de jugadores(ver cuándo los jugadores que se están fusionando comienzan a ser demasiado diferentes entre sí). Podremos realizarlo de forma manual, según el criterio del ojo humano, o de forma automática, tal y como se explicará más adelante. No resulta sencillo conocer cuál es el valor del umbral de diferencia entre dos patrones como criterio de parada porque estas diferencias son variables y pueden depender de los colores de los uniformes, de las condiciones de iluminación o de la posición de las cámaras. De tal manera, el umbral que podría ser válido en un partido no tendría por qué serlo en otro.

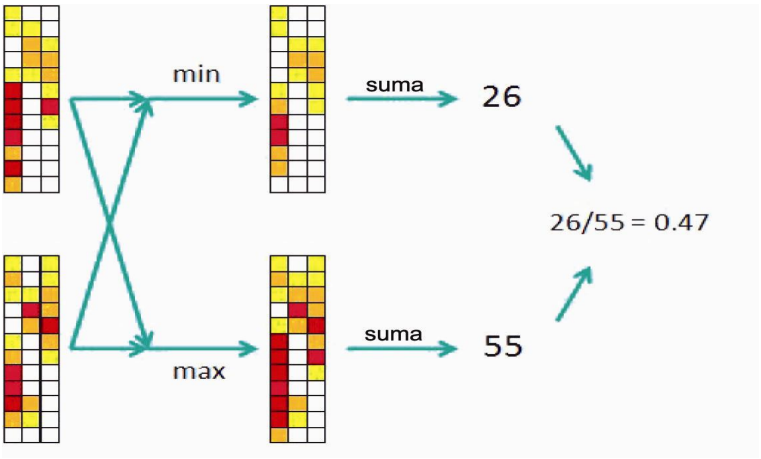

Figura B.16: Ejemplo de una comparación entre dos modelos.

Una posible solución es realizar un agrupamiento completo hasta que solo quede un clúster, habiendo guardado el número de modelos fusionados en cada clúster a lo largo de cada iteración. Con estas estadísticas es posible encontrar un punto de parada y volver a utilizar el algoritmo de agrupamiento desde el principio hasta ese punto. Tomaremos un número de patrones posibles superior al número de patrones deseado. De esta manera tendremos un cierto margen de maniobra y podremos seleccionar entre los obtenidos de forma automática, los más apropiados. La ventaja de este método es que no tenemos por qué seleccionar los grupos de modelos de jugador con más muestras, ya que los más interesantes no son siempre los que más muestras tienen. En el caso de los modelos de porteros o de árbitros éstos pueden tener una mayor importancia pero un número de muestras inferior.

Una vez que se hayan seleccionado los mejores grupos de modelos, habrá que decidir como obtener el patrón a partir de dichos grupos. En un principio existen dos posibildades:

- Usar la media de los modelos de cada grupo para obtener los patrones que se vayan a usar.
- Seleccionar uno de los modelos de cada grupo el cuál sea representativo del grupo, de forma que sea tomado como patrón de dicho grupo.

En la práctica no se ha podido determinar cuál de los dos modelos es más fiable, ya que cada uno de los modelos tiene ventajas o inconvenientes dependiendo de la situación.

Esta fase puede hacerse de forma automática;no obstante, es recomendable que sea supervisada por un operador humano. Dicha persona se encargará de ver si los patrones elegidos han sido clasificados de forma correcta y en caso contrario corregirlos. Podemos ver un ejemplo de esta clasificación a supervisar en la Figura B.17. En dicha clasificación se puede observar como uno de los patrones obtenidos para el equipo 1 es en realidad el de un entrenador. Dicho entrenador ha sido clasificado

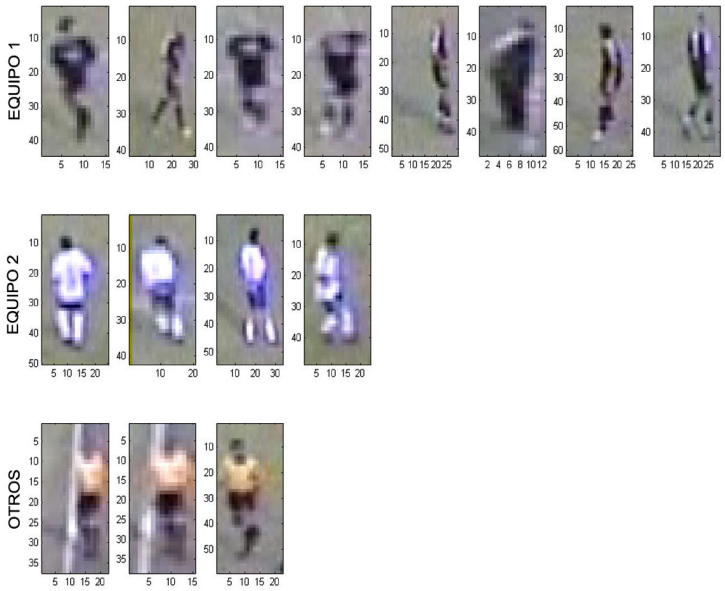

Figura B.17: Ejemplo de obtención y clasificación automática de patrones antes de realizar la supervisión.

como si perteneciese al equipo 1, debido a que viste con colores parecidos. También podemos observar que los jugadores que se han convertido en patrón parecen disponer de diferentes posiciones relativas a la cámara. Esto se debe a que es más sencillo que los jugadores que estén en las mismas posiciones tengan mayor similitud; y por lo tanto, se agrupen más fácilmente en el mismo grupo.

B.5. Modo "funcionamiento en tiempo real"

Una vez se ha obtenido la ROI, el fondo y los patrones en el modo "entrenamiento", éstos serán los pasos a seguir en el modo "funcionamiento en tiempo real":

1. Detección de movimiento basado en la resta respecto del fondo y primer filtrado.
2. Aplicación de la LUT para el segundo filtrado y la segmentación(clusterización de colores).
3. Creación de modelos para cada uno de los blobs que hayan sido detectados.
4. Comparación de los modelos con los patrones y clasificación de los mismos.

Para obtener un tiempo de ejecución menor en el modo "funcionamiento en tiempo real", un buen método (si se dispone de memoria suficiente) será el de precalcular la clusterización para cada uno de los posibles colores y anotar los resultados en una tabla a la que llamamos Look-Up table(LUT). Esto se debe a que el tiempo de acceso a memoria suele ser muy inferior al tiempo de cómputo.

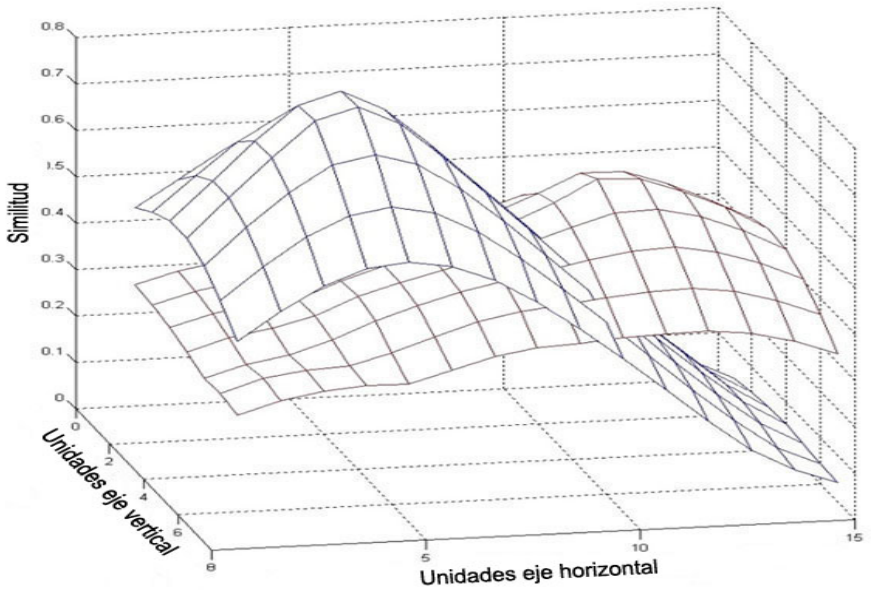

Figura B.18: Malla obtenida al aplicar la función de similitud entre un blob formado por dos jugadores, uno de cada equipo, y dos patrones, también uno de cada equipo.

Cada blob se comparará individualmente con cada patrón en cada uno de los puntos de una malla de desplazamientos. De esta forma obtendremos una serie de mallas con resultados de parecidos, una por cada patrón, y podremos detectar fácilmente a dos jugadores en el mismo blob, tal y como podemos ver en la Figura B.18. Consideraremos que hay un jugador en un punto de la malla si:

- Tiene un valor de similitud mayor que los puntos vecinos(es un máximo local).
- Tiene un valor de similitud mayor que el resto de los patrones en ese punto.
- Tiene un valor de similitud mayor que el umbral de detección.

Debemos definir la distancia mínima entre dos jugadores en la imagen para evitar duplicidad de detecciones. Para ello seguiremos un proceso, el cuál se divide en tres etapas:

1. Se buscará el punto con el valor de similitud mayor y se comprobará si se cumplen las tres condiciones mencionadas anteriormente. En tal caso se considerará que en ese punto hay un jugador y su clase será determinada por el patrón que haya obtenido el valor máximo.
2. Los valores de similitud de los puntos que hay alrededor del máximo hasta una distancia determinada serán puestos a 0, evitando de esta manera que sean elegidos como jugadores demasiado cercanos.

3. Se comprobará si hay algún máximo local que supere el umbral. Si es así, se repetirán los pasos previos. En caso contrario el proceso habrá finalizado.

Mediante esta forma de localizar e identificar a los jugadores además de saber la posición y el equipo al que pertenezca dicho jugador, obtendremos una nota de calidad de la medida obtenida.

Ante los cambios de iluminación , los patrones escogidos inicialmente pueden quedar obsoletos, de manera que haya que ir sustituyéndolos por unos nuevos. Por ello si un patrón apenás está siendo útil en la identificación de jugadores, puede ser eliminado dando lugar a otro patrón más útil. Una forma de ver si un patrón es más o menos últil es acumular el número de jugadores detectados por cada patrón y su calidad. De tal manera, aquellos patrones que hayan detectado a un menor número de jugadores al cabo de un tiempo serán descartados. Por otra parte es interesante que el operador humano comunique al sistema cuándo ha fallado en la identificación de un jugador y cuándo no ha detectado a un jugador que debería de haber detectado. El jugador que no ha sido detectado puede ser añadido como un modelo nuevo; y por otra parte, el número de fallos acumulados por cada patrón nos puede servir como criterio a la hora de tomar la decisión de qué patrones deben sustituirse por otros nuevos.

ANEXO C

SISTEMA DE SEGUIMIENTO

C.1. Introducción

El objetivo principal de este sistema es que sea capaz de realizar un seguimiento correcto de los jugadores en el campo de futbol, de manera que se puedan seguir de forma individual cada uno de los jugadores de campo de ambos equipos. Para ello, como información de entrada serán tomadas las medidas obtenidas mediante la detección y las trayectorias "habituales" obtenidas mediante el algoritmo de trayectorias medias.

En primer lugar veremos por qué y como realizar un seguimiento en el plano en vez de realizarlo directamente sobre la imagen. Para ello necesitaremos aplicar una transformación proyectiva sobre las medidas, de manera que hallemos su posición equivalente en el plano del suelo.

En segundo y último lugar analizaremos dos algoritmos de seguimiento que han sido desarrollados de tal manera que se pueden utilizar como datos de entrada las medidas obtenidas por varias cámaras a la vez:

- MSUKF (Multiple Sensor Unscented Kalman Filter)
  Es una variación del filtro de Kalman, en la cuál en vez de seguir un modelo lineal se sigue un modelo no lineal determinado. Además se trabaja con más de un sensor a la vez (en nuestro caso usamos tres cámaras).

- MHT (Multiple Hypothesis Tracking)
  Es otra variación del filtro de Kalman, en la cuál se tienen en cuenta un número de hipótesis, indeterminado a priori, para cada objetivo a seguir (aunque limitado a un número máximo de hipótesis por objetivo) en cada instante temporal. En este caso también se trabaja con más de un sensor a la vez (al igual que con el algoritmo anterior también se trabaja con tres cámaras). Este algoritmo hace posible que la decisión de las trayectorias que realmente han seguido los jugadores se retrase, de manera que pase el tiempo que sea necesario para evitar errores en la asignación de medidas a los jugadores.

C.2. Seguimiento en el plano

C.2.1. Introducción

Existen dos razones por la que es recomendable realizar un seguimiento sobre el plano del suelo en vez de sobre la imagen:

- Mejorar la precisión de seguimiento de los jugadores, ya que se reduce el efecto del error perspectivo en la imagen.
- Poder realizar un seguimiento de objetos a partir de los datos obtenidos de forma simultánea por varios sensores a la vez, uniendo a continuación las diferentes observaciones en un plano común.

En nuestro caso en primer lugar será descrita la transformación realizada para poder trasladar la información de la imagen al plano. A continuación se comentará el modelo de ruido usado a la hora de realizar el seguimiento sobre el plano; y por último, se mostrarán los resultados de la comparación entre realizar el seguimiento sobre la imagen o sobre el plano. Para ello usaremos un tracker sencillo basado en el Filtro de Kalman. Dicho algoritmo de seguimiento se usará de la misma manera tanto con medidas con coordenadas de la imagen como con medidas con coordenadas del pano. Para que la comparación sea correcta la configuración de dichos trackers deberá ser equivalente.

C.2.2. Transformación de la imagen al plano

Para trasladar la información obtenida en la imagen al plano aplicaremos una transformación proyectiva. Como primera aproximación asumiremos que no existe ninguna distorsión creada por la lente de la cámara. Un punto en el plano proyectivo representará un rayo que pasa por el origen en el espacio 3D, tal y como podemos ver en la Figura C.1. La transformación proyectiva entre dos planos puede ser representada como una transformación lineal: $p_2 = T_{21}.p_1$. Si queremos volver al sistema de referencia anterior podremos usar como matriz de transformación T_{21}^{-1}, de manera que la transformación también sea lineal. Si dicha transformación es representada en coordenadas cartesianas, los resultados no serán lineales. Sean:

$p_c = (x_i, y_i, 1)^T$: las coordenadas del punto i en el sistema de referencia de la cámara.

$p_g = (x_i^g, y_i^g, 1)$: las coordenadas homogéneas asociadas a las coordenadas del punto i en el sistema de referencia del plano del suelo(x_i^g, y_i^g).

T_{gc}: la transformación proyectiva para las coordenadas p_g y p_c.

Para obtener dicha transformación proyectiva, para cada par i de puntos se tomará está ecuación:

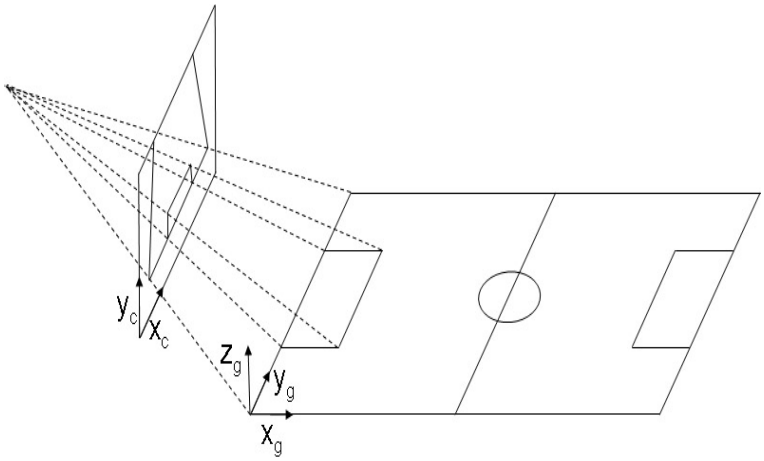

Figura C.1: Ejemplo gráfico de la transformación de coordenadas de la imagen al plano.

$$(\lambda_i x_i^g, \lambda_i y_i^g, \lambda_i)^T = (T_{gc}(x_i, y_i, 1))^T$$

Si la desarrollamos:

$$\begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i^g x_i & -x_i^g y_i & -x_i^g \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i^g x_i & -y_i^g y_i & -y_i^g \end{pmatrix} t = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{C.1}$$

donde $t = (t_{11}t_{12}t_{13}t_{21}t_{22}t_{23}t_{31}t_{32}t_{33})^T$ es un vector formado por los elementos de la matriz de homografía T_{gc}.
Si usamos cuatro pares de puntos(En nuestro caso, por ejemplo, podríamos tomar los cuatro puntos asociados a los extremos del campo que se pueden ver cada cámara - por cada cámara tendremos una homografía), de forma que tres de ellos no sean colineales, podremos construir una matriz M de 8x9 elementos, donde $Mt = 0$. De esta manera, la solución t corresponde al vector propio asociado con el menor valor propio(en este caso el valor propio nulo) de la matriz $M^T M$; el cuál puede ser calculado mediante una descomposición en valores singulares de la matriz M.
Una vez hallemos la matriz de homografía T_{gc}, podremos realizar su transformación inversa para obtener la homografía en el sistema de referencia deseado.

C.2.3. Modelo de ruido usado para realizar el seguimiento sobre el plano

Para explicar cómo vamos a implementar la transformación proyectiva de nuestro sistema de seguimiento, usaremos como ejemplo de algoritmo de seguimiento, un filtro de Kalman estándar con un modelo de velocidad constante para los objetos móviles.

Las medidas que vamos a obtener estarán formadas por un punto en la imagen y un área de incertidumbre alrededor de dicho punto. A nuestro filtro de Kalman le proporcionaremos una medida

del objeto en movimiento a seguir en la posición (x,y) de la imagen. Dichas coordenadas(tanto las medidas como su área de incertidumbre) estarán en unidades de píxeles y deberán ser transladadas al plano del suelo, en donde estarán en unidades métricas. Para transformar la matriz de covarianza de incertidumbre de la imagen al plano del suelo(como se propone en [16]), se necesitará realizar una transformación en tres pasos:

1. Realizar un cambio a coordenas homogéneas en la imagen.
2. Transformar las coordenadas de la imagen a coordenadas del plano.
3. Realizar un cambio a coordenadas no homogéneas en el plano.

Mediante estas transformaciones obtendremos un modelo de ruido que tiene en cuenta la influencia del efecto perspectivo a la hora de realizar un seguimiento en el plano del suelo a partir de la obtención de las medidas y sus modelos de ruido en la imagen. A la hora de obtener dicho modelo de ruido(en el plano), deberemos de tener en cuenta dos tipos de ruido:

- Ruido de estado
  Este ruido estará asociado a los cambios de velocidad que consideremos que pueda haber, ya que el modelo de velocidad constante que vamos a usar sólo puede ser considerado válido localmente. Considerando que el objetivo a seguir tiene una aceleración que puede ser modelada como ruido blanco de media cero y covarianza q_i, la matriz de ruido de estado Q_i para la coordenada (x_i^g, x_i^g) será:
  $$Q_i = q_i . \begin{pmatrix} \frac{dt^4}{4} & \frac{dt^3}{2} \\ \frac{dt^3}{2} & dt^2 \end{pmatrix} \tag{C.2}$$
  donde dt representa un intervalo de tiempo. Para ajustar el ruido de estado deberemos de realizar un ajuste en unidades de píxeles que dependerá del efecto perspectivo. Normalmente se toma la zona con más variaciones de ruido de estado en píxel; la cuál suele ser la más cercana a la cámara, para realizar dicho ajuste. De tal modo se asegurará el seguimiento en todas las partes de la imagen a cambio de obtener una mayor inexactitud de las posiciones elegidas.
- Ruido de medida
  Este ruido está asociado a la diferencia entre dónde está realmente la medida y donde la detecta nuestro algoritmo. La matriz de ruido de medida R se definirá en la imagen y se transformará a coordenadas del plano del suelo a partir de la matriz de homografía, a través de la transformación de tres pasos mencionada anteriormente.

C.2.4. Pruebas realizadas

Mediante las pruebas que fueron realizadas en [1](Capítulo 4.2.4) se consiguió comparar el rendimiento de un tracker en el suelo con respecto a un tracker en la imagen. Se compararon dos trackers que usaron el algoritmo de Kalman con el mismo modelo de velocidad constante, de forma que cada uno de ellos tenía un ajuste de parámetros diferente pero equivalente(ya que no podemos comparar de forma directa los parámetros en la imagen y en el plano del suelo, puesto que sus magnitudes vienen expresadas en distintas unidades).

Para ello fueron realizadas tres pruebas en las que se comprobó la precisión de las predicciones de ambos trackers para una secuencia de 550 fotogramas grabada con una cámara fija a una altura de unos 2,50 metros. El objetivo era el de seguir un coche teledirigido que se movía a lo largo de un pasillo, alejándose de la cámara, a una velocidad aproximadamente constante.

Una vez se obtuvo la posición del coche mediante el algoritmo de detección en cada fotograma, se aplicó un filtro de Kalman para realizar el seguimiento. La homografía; la cuál fue calculada previamente tal y como se explicó anteriormente, nos permitió obtener los puntos del plano asociados a cada punto de la imagen. De tal manera se pudo seguir al mismo objeto con trackers que trabajaban de forma independiente.

Para comparar el rendimiento de ambos trackers se calculó la distancia media entre los puntos predichos por los trackers y las medidas asociadas, ya que aunque siempre habrá una diferencia, está será menor cuanto mejores sean las predicciones del tracker. En un principio se puede pensar que para medir el error cometido sería más fiable calcular la distancia entre la posición real del objeto móvil y las estimaciones de posición de los trackers; pero existen varios problemas, ya que las estimaciones calculadas a partir de las medidas y la predicción no estarán siempre disponibles; y las posiciones reales de los objetos en movimiento no se conocen. Las comparaciones entre las predicciones y las medidas actuales se realizaron en tres niveles de tiempo:

- Corto plazo:
  En este caso se fue comparando la precisión de las predicciones entre fotogramas consecutivos. Las distancias medias obtenidas entre las medidas y las predicciones se pueden ver en la Figura C.2. Tanto las distancias de los trackers en el suelo como las de los trackers en la imagen fueron medidas en el suelo, usando el milímetro como unidad de distancia para poder compararlas de forma directa. Estas distancias fueron calculadas mediante el uso de la distancia euclidea. Podremos ver las distancias obtenidas entre las predicciones y las medidas en la FiguraC.3. Tal y como se puede ver en dicha figura el seguimiento en el plano del suelo obtiene mejores resultados que el seguimiento en la imagen. Comparativamente, se puede ver que el tracker del plano del

Imagen	Suelo
54,90	18,89

Figura C.2: Distancias medias(en milímetros) entre las predicciones y las medidas para el tracker en la imagen y el tracker en el suelo para fotogramas consecutivos.

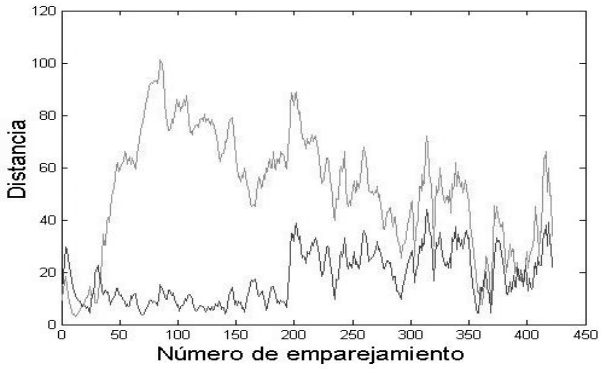

Figura C.3: Distancia obtenida entre predicciones y medidas para ambos trackers(el tracker en el suelo corresponde a la línea oscura y el tracker en la imagen corresponde a la línea clara)

suelo obtiene mejores resultados que el de la imagen si el objeto está cerca de la cámara, ya que en ese intervalo temporal la variabilidad de la velocidad del objeto en la imagen suele ser mayor(es decir, el objeto se mueve en cada fotograma un número mayor de píxeles). Cuando el objeto está lejos de la cámara el tracker del plano del suelo sigue obteniendo mejores resultados que el de la imagen, pero comparativamente la diferencia es bastante menor.

- Medio plazo:
  En el segundo caso, usando la misma secuencia, se realizó una decimación del número de medidas usadas, de manera que se obtuvieron 550/f medidas (donde f es el factor de decimación). Se usaron las mismas matrices de ruido de estado y medida que en el caso de Corto plazo. Las distancias medias obtenidas entre las medidas y las predicciones se pueden ver, para distintos valores de decimación, en la Figura C.4.
  Podemos observar que la distancia media aumenta conforme se incrementa el valor del factor de decimación, ya que el movimiento del objeto se hace menos predecible. Además de ver que el error es menor en el caso del tracker del suelo respecto al tracker de la imagen, podemos observar que conforme aumenta el factor de decimación el aumento proporcional del error es menor en el caso del tracker del suelo.

- Largo plazo:
  Por último se realizó una simulación de oclusión o pérdida del objeto movil durante un gran

f	Imagen	Suelo
1	54,90	18,89
2	78,49	22,89
3	97,75	26,58
5	132,01	31,06
8	180,02	40,33
12	248,08	55,93

Figura C.4: Distancias medias(en milímetros) entre las predicciones y las medidas para el tracker en la imagen y el tracker en el suelo tras aplicacr diferentes factores de decimación f.

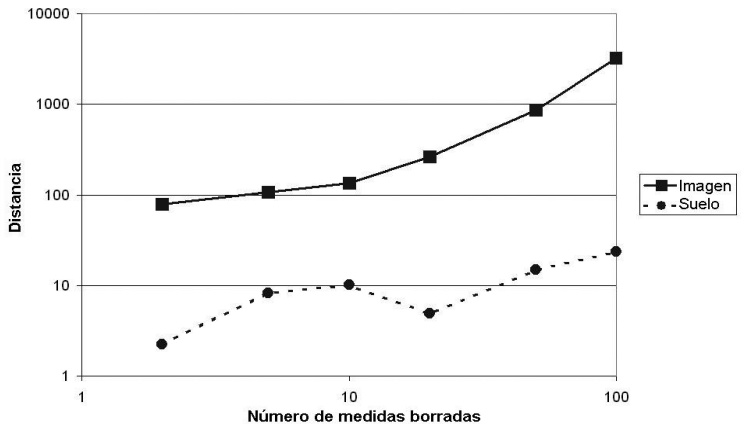

Figura C.5: Distancia obtenida entre predicciones y medidas para ambos trackers en el momento que aparece la primera medida después de eliminar un conjunto de medidas consecutivas(los ejes están representados en escala logarítmica)

período de tiempo; es decir, usando la misma secuencia de fotogramas se eliminó un gran número de fotogramas consecutivos para ambos trackers. En este caso también se usó la misma configuración para ambos trackers respecto a las dos pruebas anteriores. Se realizaron diferentes test en los que el número de medidas borradas de forma consecutiva varió entre 2 y 100. Se tomó como referencia la distancia existente entre la predicción y la primera medida que hubiera a continuación del conjunto de medidas borradas(aunque no fuera emparejada de forma correcta con el tracker). Con la configuración usada se puede apreciar como el tracker del suelo no pierde el objetivo incluso después de 100 fotogramas, ya que la trayectoria seguida es bastante lineal; sin embargo, el tracker de la imagen pierde al objeto depués de oclusiones de un número de fotogramas igual o mayor a 10. Podemos ver una muestra de las distancias obtenidas en la Figura C.5

C.3. Algoritmo de seguimiento MSUKF

C.3.1. Introducción

Nuestros algoritmos de seguimiento se van basar en el filtro de Kalman [17], un algoritmo sencillo y rápido comparado con otros algoritmos de seguimiento. Dicho algoritmo solo es capaz de funcionar con modelos lineales y gaussianos; no obstante, existen versiones de este algoritmo que sí son capaces de trabajar con modelos no lineales ni gaussianos. Actualmente existe un algoritmo, en concreto, el cuál es capaz de tratar con modelos de este tipo de manera eficiente y con un coste computacional no muy alto. Dicho algoritmo de seguimiento es el llamado Unscented Kalman Filter(UKF).

En nuestro caso buscamos trabajar con sistemas multisensor(en este caso sistemas multicámara), de manera que dispongamos de varias cámaras en las que la información procesada por cada una de ellas sea tratada en conjunto con la información del resto de las camáras; ya que se obtienen mejores resultados de esta forma respecto a usar varios filtros UKF independientes(uno para las medidas de cada cámara). En nuestro caso dispondremos de 3 cámaras, de forma que cada una cubra una zona distinta del campo. Aun así podrá haber solapamientos entre las áreas de visión de las cámaras, de manera que haya momentos en los que podamos tener medidas en distintas cámaras para el mismo jugador a la vez.

En primer lugar, deberemos crear una referencia común para poder integrar las medidas de las distintas cámaras, tal y como se explica en el apartado B.2.2. Para ello deberemos de calcular préviamente las matrices de homografía para cada cámara. Tras esto obtendremos las posiciones puntuales de cada jugador -a partir del sistema de detección y clasificación el cuál se ha descrito en el Anexo B -para cada cámara y pasaremos las medidas obtenidas- mediante su transformación proyectiva- al plano común. Una vez disponemos de las medidas en el plano común, alimentaremos con éstas a nuestro algoritmo de seguimiento.

C.3.2. Unscented Kalman Filter(UKF)

Comenzaremos hablando del algoritmo de seguimiento UKF, puesto que nuestro sistema de seguimiento se va ha basar en dicho algoritmo. En un principio definiremos los términos de estado y covarianza. El *estado* indica, en nuestro caso, la *posición media* en la que se detecta al jugador en el campo de futbol; y la *covarianza* indica la *variabilidad respecto a dicha posición media*, en la que el jugador se puede encontrar. Si la covarianza aumenta significa que la medida es más mala que buena; y por tanto existirá una variabilidad alta respecto a la posición media.
Llamaremos estado extendido y covarianza extendida a la concatenación de dichos estados y

covarianzas con las variables de ruido:

$$x_k^a = [x_k^T v_k^T n_k^T] \tag{C.3}$$

$$P_k^a = \begin{bmatrix} P_k & 0 & 0 \\ 0 & R^v & 0 \\ 0 & 0 & R^n \end{bmatrix} \tag{C.4}$$

siendo: R^v: matriz de ruido de estado. R^n: matriz de ruido de medida.

El UKF original se divide en tres etapas:

1. Inicialización de los estados y las covarianzas.

2. Fase de predicción de estado y medida:
   Es similar a la fase de predicción del filtro de Kalman.

$$\hat{x}_k^- = F(x_k, u_k, v_k) \tag{C.5}$$

$$\hat{y}_k^- = H(x_k, n_k) \tag{C.6}$$

   siendo:

   x_k: Estado en el instante de tiempo k(no observable directamente).
   y_k: Medida observada en el instante de tiempo k.
   $\hat{x}_k^-$: Predicción de estado en el instante de tiempo k.
   $\hat{y}_k^-$: Predicción de medida en el instante de tiempo k.
   u_k: Datos observados externamentete en el instante de tiempo k.
   v_k: Ruido de estado en el instante k.
   n_k: Ruido de medida en el instante k.
   F: Modelo dinámico.
   H: Matriz de relación estado-medida.

3. Fase de estimación de estado:

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - \hat{y}_k^-) \tag{C.7}$$

$$\hat{P}_k = \hat{P}_k^- - (K_k P_{yky k} K_k^T) \tag{C.8}$$

   siendo:

   $\hat{x}_k^-$: Predicción de estado en el instante de tiempo k.
   $\hat{x}_k$: Estimación de estado en el instante de tiempo k.
   $\hat{P}_k^-$: Predicción de covarianza en el instante de tiempo k.
   $\hat{P}_k$: Estimación de covarianza en el instante de tiempo k.
   $\hat{y}_k^-$: Predicción de medida en el instante de tiempo k.
   K_k: Ganancia de Kalman en el instante de tiempo k.

La ganancia de Kalman podrá ser calculada de esta manera:

$$K_k = P_{xkyk} P_{ykyk}^{-1} \tag{C.9}$$

donde:

P_{ykyk}: Covarianza de medida

P_{xkyk}: Covarianza de medida condicionada al estado.

A su vez P_{ykyk} y P_{ykyk} son calculadas en el filtro UKF usando puntos sigma. Los puntos sigma son el menor conjunto posible de puntos cuya media y covarianza es la misma que la del estado. Para un estado con n componentes serán necesarios 2n+1 puntos sigma. Por tanto, la media y la covarianza de los puntos sigma, una vez apliquemos el modelo dinámico, representarán la media y la covarianza de estado, con la ventaja de poder usar modelos dinámicos no lineales.

C.3.3. Multi Sensor Unscented Kalman Filter(MSUKF)

En el sistema utilizado dispondremos de la visión de al menos dos cámaras por cada punto del campo, de manera que las medidas obtenidas en cada cámara serán combinadas para obtener una estimación por jugador. Llamaremos Multi Sensor Unscented Kalman Filter(MSUKF) a la modificación del algoritmo UKF que vamos a utilizar [18]. En este algoritmo de seguimiento dispondremos de 5 fases:

1. Inicialización de los estados y las covarianzas.
2. Fase de predicción de estado.
3. Emparejamiento de medidas con trackers.
4. Fase de predicción de medida.
5. Fase de estimación.

Podemos ver un diagrama de bloques asociado a dicho algoritmo en la Figura C.6

Inicialización de los estados y las covarianzas

En esta fase cada tracker se inicializa con el valor de la estimación calculada en la última iteración, o en caso de estar en el primer fotograma, con los valores asociados a las posiciones iniciales de los jugadores.

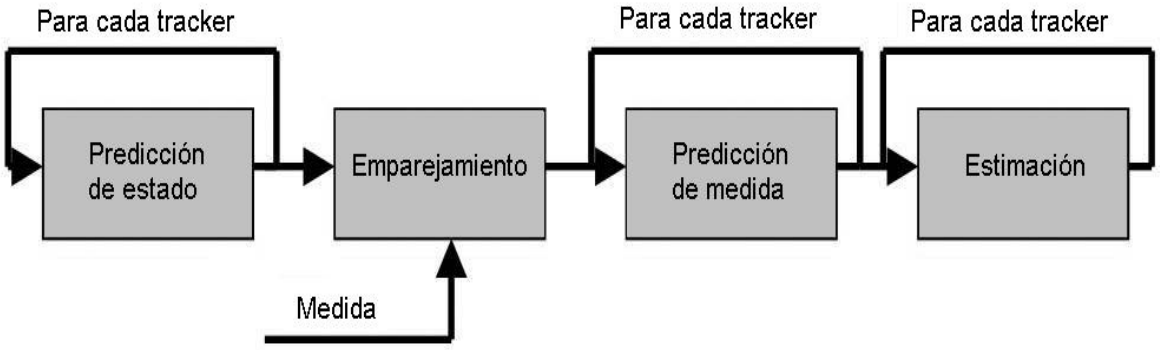

Figura C.6: Diagrama de bloques del algoritmo MSUKF utilizado.

Predicción de estado

Si conocemos el estado previo $\hat{x}_{k-1}$ y su covarianza $\hat{P}_{k-1}$, podremos calcular el estado extendido y la covarianza extendida concatenando la estimación previa con el ruido de estado. En esta fase tan solo se concatena el ruido de estado. El ruido de medida se concatenará posteriormente en la fase de predicción de medida, a diferencia del UKF; en el cuál se concatenan ambos a la vez.

$$\hat{x}^a_{k-1} = [\hat{x}^T_{k-1} \;\; E[v_k]]^T, \qquad E[v_k] = [0 \dots 0] \tag{C.10}$$

$$\hat{P}^a_{k-1} = \begin{bmatrix} \hat{P}_{k-1} & 0 \\ 0 & R^v \end{bmatrix} \tag{C.11}$$

Si disponemos de n componentes para el estado extendido, el número de puntos sigma χ^a_{k-1} creados será de 2n+1. El primer punto sigma se corresponderá con la estimación del estado en el fotograma anterior. Los n puntos sigma siguientes serán la estimación del estado en el fotograma anterior más cada una de las columnas de la matriz de estimación y los últimos n puntos sigma serán la estimación del estado en el fotograma anterior menos cada una de las columnas de la matriz de estimación:

$$\Delta_k = \sqrt{(n+\lambda)\hat{P}^a_{k-1}} \tag{C.12}$$

$$\chi^a_{k-1} = [\hat{x}^a_{k-1} \quad \hat{x}^a_{k-1} + \Delta_k \quad \hat{x}^a_{k-1} - \Delta_k] \tag{C.13}$$

Nuestra agrupación de puntos sigma puede dividirse en dos tipos: los que provienen del estado χ^x_{k-1} y los que provienen del ruido de estado χ^v_{k-1}. Los pesos asignados a cada punto sigma se calcularán de la misma manera que en la Unscented Transform, de manera que el peso 0 será calculado de forma diferente para las medias y las covarianzas:

$$\begin{aligned} W_0^{(m)} &= \lambda/(n+\lambda) \\ W_0^{(c)} &= \lambda/(n+\lambda) + (1+\alpha^2+\beta) \\ W_i^{(m)} &= W_i^{(c)} = 1/[2(n+\lambda)] \quad i = 1, 2, \dots 2n \end{aligned} \tag{C.14}$$

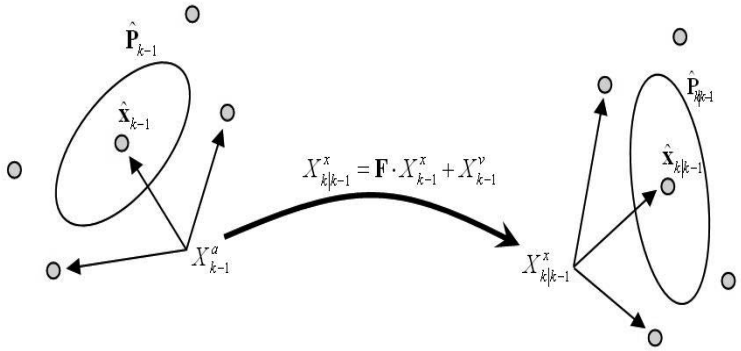

Figura C.7: Esquema gráfico en el que podemos ver la generación de puntos sigma a partir de la media y la covarianza del estado anterior, la predicción de los nuevos puntos sigma a partir de la matriz de transición y la recuperación de la media y la covarianza a partir de los nuevos puntos sigma

Mediante la matriz de transición F podremos predecir el valor de los puntos sigma en el instante de tiempo k-ésimo:

$$\chi^x_{k|k-1} = F \cdot \chi^x_{k-1} + \chi^v_{k-1} \tag{C.15}$$

Dicha matriz será calculada a partir del modelo de predicción elegido. De tal forma, una vez disponemos de la predicción de los puntos sigma y de sus pesos, podremos calcular la predicción para cada estado de cada tracker (de la media y la covarianza) de esta manera:

$$\begin{aligned} \hat{x}_{k|k-1} &= \hat{x}^-_k = F \cdot \hat{x}_{k-1} \\ \hat{x}_{k|k-1} &= \sum_{i=0}^{2n} W_i^{(m)} \chi^x_{i,k|k-1} \end{aligned} \tag{C.16}$$

$$\hat{E}^x_{i,k} = \chi^x_{i,k|k-1} - \hat{x}_{k|k-1} \tag{C.17}$$

$$\hat{P}_{k|k-1} = \sum_{i=0}^{2n} W_i^{(c)} E^x_{i,k} (E^x_{i,k})^T \tag{C.18}$$

Podemos ver una representación gráfica de los pasos realizados en la predicción de estado en la Figura C.7

Emparejamiento

Una vez disponemos de la la predicción de estado para cada tracker, deberemos asociar las medidas obtenidas mediante el algoritmo de detección con los trackers. Para ello nuestro sistema de asociación se basará en dos principios:

- Asignar como máximo una medida a cada tracker (para cada cámara).

- No asignar la misma medida a diferentes trackers.

El método más sencillo y rápido sería el de asignar a cada tracker la medida más cercana, también conocido como Método del Vecino Más Cercano(Nearest Neighbor). Sin embargo, hemos preferido usar otro método, que aunque sea más lento obtiene mejores resultados. Dicho método consiste en minimizar la suma global de distancias entre trackers y medidas.

Para que la cantidad de información a procesar sea menor (y por tanto el método no sea tan lento) utilizaremos la técnica del *gating*; la cuál consiste en crear una zona alrededor de cada tracker, llamada área de gating, de forma que las medidas que queden fuera de esa zona no serán tenidas en cuenta como candidatas a ese tracker. Para ello crearemos una lista con las posibles medidas para cada tracker, seleccionando aquellas cuya distancia de Mahalanobis respecto a la predicción de cada tracker sea menor que un umbral.

Tras esto crearemos una tabla con todas las posibles combinaciones de asociación y escogeremos la combinación de medidas-predicciones cuya suma de distancias sea menor. Deberemos de considerar la medida nula dentro de la lista de posibles medidas para cada tracker, ya que es posible que la medida asignada a un tracker no sea ninguna de las que estén dentro de su área de gating, de manera que no sea posible hallar ninguna matriz de combinaciones que cumpla con las dos condiciones necesarias expuestas anteriormente. Como en esta fase no estamos aplicando ningún tipo de relación entre las medidas de cada cámara, el mismo proceso se aplicará de forma independiente para las medidas obtenidas en cada una de ellas.

Predicción de medida

Tras realizar el emparejamiento entre predicciones y medidas pasaremos a la fase de predicción de medidas. Calcularemos los estados y las covarianzas extendidas para cada par predicción-medida asociada concatenando los ruidos de medida. De esta forma, tanto los estados como las covarianzas extendidas no tendrán una dimensión fija, ya que dependerá del número de medidas que hayan quedado dentro del área de gating del tracker.

$$\hat{x'}^{a}_{k} = [\hat{x}_{k|k-1} \;\; 0 \;\; 0 \ldots]^{T} \tag{C.19}$$

$$\hat{P'}^{a}_{k} = \begin{bmatrix} \hat{P}_{k|k-1} & 0 \\ 0 & R^{n} \end{bmatrix} \tag{C.20}$$

$$R^{n} = \begin{bmatrix} R^{n}_{1} & 0 & \ldots & 0 \\ 0 & R^{n}_{2} & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & R^{n}_{m} \end{bmatrix} \tag{C.21}$$

Si disponemos de S medidas (provenientes cada una de un sensor diferente) para un tracker, dicho tracker tendrá un estado extendido con n'= 2S+4 componentes y se generarán 2n'+1 = 4S+9 puntos sigma.

$$\Delta'_k = \sqrt{(n' + \lambda')\hat{P'}^a_k} \tag{C.22}$$

$$\chi'^a_{k-1} = [\hat{x'}^a_k \quad \hat{x'}^a_k + \Delta'_k \quad \hat{x'}^a_k - \Delta'_k] \tag{C.23}$$

cuyos pesos serán calculados de esta forma:

$$\begin{aligned} W'^{(m)}_0 &= \lambda'/(n' + \lambda') \\ W'^{(c)}_0 &= \lambda'/(n' + \lambda') + (1 + \alpha'^2 + \beta') \\ W'^{(m)}_i &= W'^{(c)}_i = 1/[2(n' + \lambda')] \quad i = 1, 2, \dots 2n' \end{aligned} \tag{C.24}$$

Nuestra agrupación de puntos sigma puede dividirse en dos tipos: los que provienen del estado χ'^x_k y los que provienen del ruido de medida χ'^n_k; los cuáles a su vez pueden ser diferenciados dependiendo de la medida que se ha usado para generarlos: $\chi'^{n(1)}_k$,$\chi'^{n(2)}_k$,...,$\chi'^{n(S)}_k$.

Mediante la matriz de relación estado-medida H, podremos obtener la predicción de medida de los puntos sigma:

$$Y^{(s)}_{k|k-1} = H \cdot \chi'^x_k + \chi'^{n(s)}_k \quad s = 1, 2, \dots, S \tag{C.25}$$

De esta manera, mediante estos nuevos puntos sigma, podremos hallar para cada medida la predicción de medida, la predicción de covarianza y la covarianza cruzada estado-medida:

$$\hat{y}_{k|k-1} = \sum_{i=0}^{2n'} W'^{(m)}_i Y^{(s)}_{i,k|k-1} \tag{C.26}$$

$$\hat{E}'^{x(s)}_{i,k} = \chi^{(s)}_{i,k|k-1} - \hat{x}_{k|k-1} \tag{C.27}$$

$$\hat{E}^{y(s)}_{i,k} = Y^{(s)}_{i,k|k-1} - \hat{y}_{k|k-1} \tag{C.28}$$

$$P^{(s)}_{\hat{y}_k\hat{y}_k} = \sum_{i=0}^{2n'} W'^{(c)}_i E^{y(s)}_{i,k} (E^{y(s)}_{i,k})^T \tag{C.29}$$

$$P^{(s)}_{\hat{x}_k\hat{y}_k} = \sum_{i=0}^{2n'} W'^{(c)}_i E^{x(s)}_{i,k} (E^{y(s)}_{i,k})^T \tag{C.30}$$

Podemos ver una representación gráfica de los pasos realizados en la predicción de medida en la Figura C.8

Estimación

Finálmente llegamos a la fase de estimación de estado. En un principio calcularemos la ganancia de Kalman para cada sensor que haya obtenido una medida asociada a alguno de los trackers:

$$K^{(s)}_k = P^{(s)}_{\hat{x}_k\hat{y}_k} / P^{(s)}_{\hat{y}_k\hat{y}_k} \tag{C.31}$$

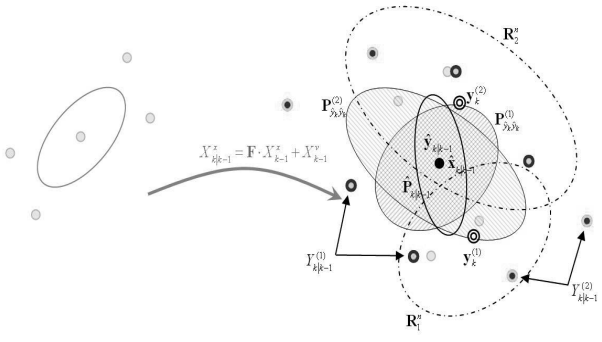

Figura C.8: Esquema gráfico en el que podemos ver una distribución de puntos sigma en la que un tracker es asociado a dos medidas, una proveniente de cada sensor.

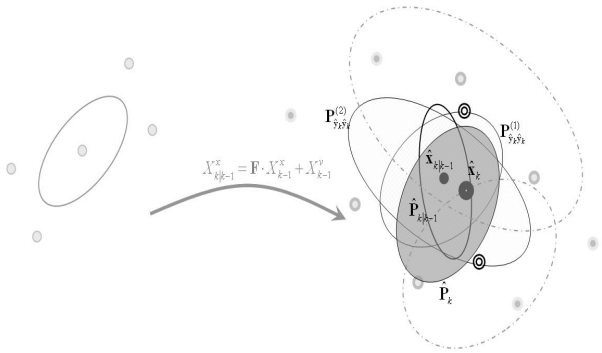

Figura C.9: Esquema gráfico de la fase de estimación para un tracker, en la que dicha estimación se calcula a partir de la predicción de estado, las predicciones de medida asociadas a dicho tracker, la ganancia del filtro de Kalman y el peso asociado a dichas medidas.

Para obtener una estimación conjunta deberemos combinar las medidas de los diferentes sensores. De tal manera, para cada medida calcularemos un peso $\beta^{(s)}$, el cuál dependerá de la distancia entre la predicción y la medida. Dicho conjunto de pesos deberá ser normalizado de forma que su suma sea unitaria. Una vez que dichos pesos hayan sido normalizados, calcularemos la estimación para cada estado de cada tracker (de la media y la covarianza) de esta manera:

$$\hat{x}_k = \hat{x}_{k|k-1} + \sum_{s=1}^{S} \beta^{(s)} K_k^{(s)} (y_k^{(s)} - \hat{y}_{k|k-1}^{(s)}) \tag{C.32}$$

$$\hat{P}_k = \hat{P}_{k|k-1} - \sum_{s=1}^{S} K_k^{(s)} P_{\hat{y}_k \hat{y}_k}^{(s)} (K_k^{(s)})^T \tag{C.33}$$

Podemos ver una representación gráfica de los pasos realizados en la estimación en la Figura C.9

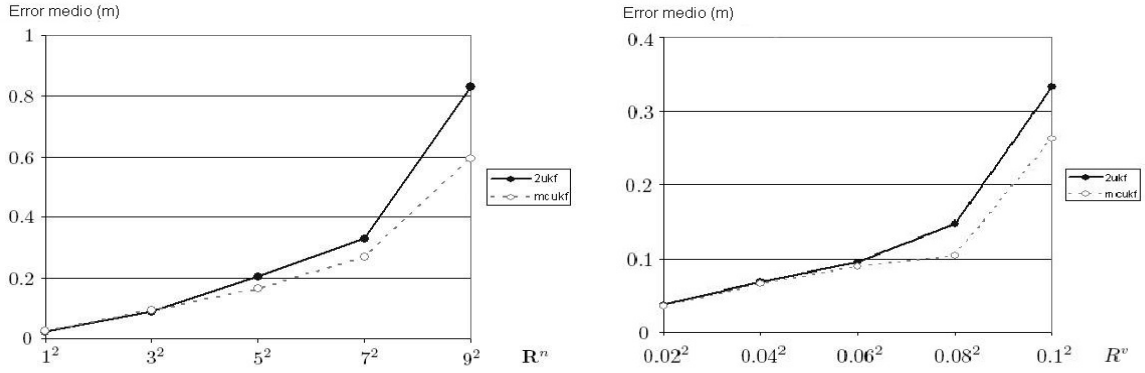

Figura C.10: Error medio obtenido en el seguimiento de una trayectoria virtual, usando un filtro MSUKF (para las medidas de ambas cámaras) y dos filtros independientes UKF(uno para las medidas de cada cámara). Para ello se ha hecho pruebas con diferentes ruidos de medida(izda.) y diferents ruidos de estado(dcha.). Los errores estarán dados en metros y los ruidos de medida en píxels.

C.3.4. Pruebas realizadas

Mediante las pruebas que fueron realizadas en [1](Capítulo 4.3.6.) se consiguió comparar el rendimiento entre usar un algoritmo MSUKF y usar dos algoritmos UKF independientes, uno por cámara(tomando como estimación conjunta la media de las dos estimaciones por separado). Para ello se creó un generador de trayectorias virtuales, de manera que se pudieran hacer diversas pruebas con distintos niveles de ruido de estado y de ruido de medida. Podemos ver los resultados de estas pruebas en la Figura C.10.

La ventaja de este método para probar los algoritmos de seguimiento es es que conocemos la posición real de los jugadores virtuales en cada momento, de manera que se conoce exactamente el error cometido. Además podemos modelar las trayectorias con la cantidad deseada de ruido y de oclusiones. En la Figura C.11 podemos ver un ejemplo de trayectoria virtual, en el que se puede apreciar la trayectoria real, la trayectoria estimada y las medidas observadas en el último frame.

Para que la comparación sea correcta, en ambos casos las medidas obtenidas en las imágenes virtuales han sido trasladadas al plano del suelo. Podemos apreciar que en situaciones de poco ruido, ambas estimaciones obtienen resultados parecidos. Sin embargo, en situaciones ruidosas, podemos apreciar en mayor medida la mejora obtenida al usar el algoritmo MSUKF respecto al considerar las medidas de ambas cámaras de forma independiente.

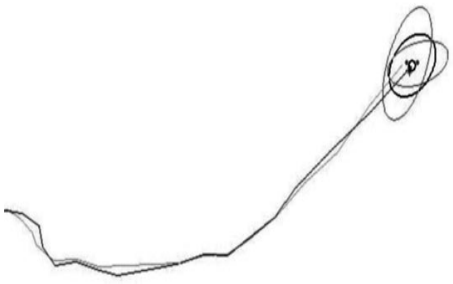

Figura C.11: Ejemplo de trayectoria virtual obtenida mediante el generador de trayectorias virtuales. La línea gris representará la trayectoria real y la línea negra la trayectoria estimada. Las dos elipses grises representan las matrices de correlación de las medidas obtenidas en ambas cámaras en el último fotograma. Por otra parte, la elipse negra representa la matriz de correlación de la estimación en dicho fotograma.

C.4. Algoritmo de seguimiento MHT

C.4.1. Introducción

Uno de los principales problemas que se dan en los algoritmos de Kalman, es que en cada fotograma hay que decidirse por una única medida en cada sensor, de manera que se obtiene una única posición estimada para cada objeto a seguir. Esto puede conducirnos a errores, ya que la opción que más probable en un momento determinado puede no serlo una vez evolucionan las medidas deseadas. Por ejemplo si dos jugadores se cruzan, una vez salgan del cruce podría pasar que hubiera un cambio de identificador entre ellos, de manera que el seguimiento no sería correcto(a pesar de que estamos siguiendo a ambos jugadores los estamos confundiendo).

Por lo tanto, sería interesante disponer de un sistema en el cuál se puedan rectificar las decisiones o postponerlas hasta estar lo suficientemente seguros de la trayectoria real que ha seguido el objeto. Por ello se pretende integrar el sistema de seguimiento multisensor desarrollado previamente en un sistema de evaluación de múltiples hipótesis. El sistema a usar será llamado Multiple Hypothesis Tracking(MHT) [19] y nos permitirá realizar el seguimiento simultáneo de varias hipótesis hasta la toma de decisiones, ampliando su uso a un entorno multicámara. A diferencia del algortimo JPDA básico, el cuál fusiona las aportaciones de diversas medidas del mismo sensor para calcular la estimación del estado; el MHT fusiona las medidas de diferentes sensores, haciendo que el cálculo de combinaciones entre medidas sea necesario. De tal manera, las combinaciones más probables pasarán a ser hipótesis, a partir de las cuáles se decidirá la trayectoria seguida por el objeto en movimiento. Mediante este sistema se eliminará la limitación de que una medida solo pueda ser asignada a un tracker en un momento determinado, ya que si la asignación es dudosa entre varios trackers, se crearán hipótesis para cada uno de ellos que se mantendrán hasta que se resuelva la duda.

Consideraremos que en un instante de tiempo k existirán t trackers($T_1, T_2, ..., T_t$), cada uno con un conjunto de W_t hipótesis: $T_i = (H^{w_1}, H^{w_2}, ..., H^{w_k})$. De tal manera, la suma de hipótesis para cada tracker en cada instante de tiempo será igual a la unidad:

$$\sum_i p(H_k^{w_i}) = 1 \ \ \forall t, \forall k \tag{C.34}$$

Por lo tanto, definiremos un estado x_k^w para cada instante de tiempo k e hipótesis w por tracker. En nuestro caso cada estado estará formado por cuatro componentes , dos correspondientes a posición y otros dos correspondientes a velocidad.

Nuestro sistema se dividirá en cuatro etapas:

1. Predicción:
   En la primera etapa se predice el estado para cada hipótesis de cada tracker a partir de la estimación del estado anterior y del modelo dinámico.

2. Asociación de datos:
   En la segunda etapa se obtienen las posibles asignaciones de medidas a cada hipótesis y se buscan las mejores combinaciones que darán lugar a nuevas hipótesis. Se eliminan las hipótesis menos probables y se recalculan las probabilidades para cada hipótesis.

3. Estimación:
   En la tercera etapa se calculan los estados estimados para cada hipótesis a partir de las predicciones y de las medidas asociadas a cada hipótesis.

4. Probabilidad a posteriori y agrupamiento de hipótesis:
   Por último se aplica una serie de correciones en el cálculo de probabilidades para conseguir un seguimiento efectivo y se reduce el número de hipótesis agrupando las más parecidas.

C.4.2. Predicción

En primer lugar calcularemos la predicción de estado $\hat{x}_{k|k-1}^w$ y de medida $\hat{z}_{k|k-1}^w$ para cada hipótesis de cada tracker, a partir de la estimación del instante anterior, o de la inicialización en caso de tratar con la primera iteración. De tal manera:

$$\begin{aligned} \hat{x}_{k|k-1}^w &= F \cdot \hat{x}_{k-1}^w \\ \hat{z}_{k|k-1}^w &= H \cdot \hat{x}_{k|k-1}^w \end{aligned} \tag{C.35}$$

donde:

$\hat{x}_{k-1}^w$: Estimación de estado en el instante k-1 para la hipótesis w asociada a un tracker.
$\hat{x}_{k|k-1}^w$: Predicción de estado en el instante k para la hipótesis w asociada a un tracker.

$\hat{z}^w_{k|k-1}$: Predicción de medida en el instante k para la hipótesis w asociada a un tracker.

F: Matriz de transición de estado.

H: Matriz de relación estado-medida.

En nuestro caso, debido al estado y al modelo utilizados(modelo de movimiento lineal y velocidad constante):

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.36}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{C.37}$$

Además calcularemos la predicción de covarianza de estado $\hat{P}^w_{k|k-1}$ y la predicción de covarianza de medida $\hat{S}^w_k$ para cada hipótesis de cada tracker:

$$\begin{aligned} \hat{P}^w_{k|k-1} &= F \cdot \hat{P}^w_{k-1} \cdot F^T + Q \\ \hat{S}^w_k &= H \cdot \hat{P}^w_{k|k-1} \cdot H^T + R \end{aligned} \tag{C.38}$$

donde:

$\hat{P}^w_{k-1}$: Estimación de covarianza de estado en el instante k-1 para la hipótesis w asociada a un tracker.

$\hat{P}^w_{k|k-1}$: Predicción de covarianza de estado en el instante k para la hipótesis w asociada a un tracker.

$\hat{S}^w_k$: Predicción de covarianza de medida en el instante k para la hipótesis w asociada a un tracker.

Q: Matriz de ruido de estado.

R: Matriz de ruido de medida.

C.4.3. Asociación de Datos

En segundo lugar realizaremos la asociación de datos. Dicha etapa se divide en cinco fases:

1. Selección de medidas:
   Lo primero que haremos es seleccionar para cada hipótesis de cada tracker aquellas medidas las cuáles se encuentren dentro del área de gating. Para ello, calcularemos la distancia de Mahalanobis entre cada medida z^c_k y la predicción de medida de cada hipótesis $\hat{z}^w_{k|k-1}$. Si dicha distancia está por debajo de un umbral γ entonces la medida asociada a dicha distancia será seleccionada. De tal manera, obtendremos un conjunto de medidas $V^{c,w}_k$ para cada hipótesis w y

para cada cámara c:

$$V_k^{c,w} = \{z_k^c : [(z_k^c - \hat{z}_{k|k-1}^w)^T \cdot (\hat{S}_k^w)^{-1} \cdot (z_k^c - \hat{z}_{k|k-1}^w)] \leq \gamma\} \tag{C.39}$$

El umbral de gating γ estará determinado por la probabilidad de gating P_G, la cuál será considerada igual para todos los trackers. De tal manera obtendremos $m_k^{c,w}$ medidas para cada cámara para cada hipótesis:

$$V_k^{c,w} = \{z_{i,k}^{c,w}\} \quad i = 1, \ldots, m_k^{c,w} \tag{C.40}$$

Para cada una de las medidas seleccionadas calcularemos la innovación asociada a ellas como:

$$v_{i,k}^{c,w} = z_{i,k}^{c,w} - \hat{z}_{k|k-1}^w \quad i = 1, \ldots, m_k^{c,w} \tag{C.41}$$

Para cada hipótesis de cada tracker deberemos añadir la posibilidad de que no exista ninguna medida correcta para dicha hipótesis. Dicha medida tendrá como índice i=0 y la innovación asociada a dicha medida será nula: $v_{0,k}^{c,w}$.

2. Obtención de la matriz de combinaciones:
   Una vez hemos seleccionado las posibles medidas para cada hipótesis, crearemos una matriz con las posibles combinaciones de medidas que puedan pertenecer a dicha hipótesis. Dicha matriz de combinaciones,Π_k^w, se calculará de forma iterativa, tomando en cada iteración las medidas que estén dentro del área de gating de cada cámara. En la primera iteración escribiremos en la primera fila de la matriz las posibles medidas capturadas por la primera cámara, para cada hipótesis w, desde la medida 0 hasta la medida $m_k^{1,w}$:

$$\Pi_k^w \mid_1 = [n_0^1 n_1^1 \ldots n_{m_k^{1,w}}^1] \quad con \;\; n_i^c = \begin{cases} z_{i,k}^{c,w}, & i \neq 0 \\ 0, & i = 0 \end{cases} \tag{C.42}$$

   En la segunda iteración, para las medidas de la segunda cámara, repetiremos $m_k^{1,w}$ veces la matriz $\Pi_k^w \mid_1$, añadiéndole en la segunda fila las medidas asociadas a las hipótesis que provienen de la cámara 2:

$$\Pi_k^w \mid_2 = \begin{bmatrix} n_0^1 n_1^1 \ldots n_{m_k^{1,w}}^1 & n_0^1 n_1^1 \ldots n_{m_k^{1,w}}^1 & \ldots & n_0^1 n_1^1 \ldots n_{m_k^{1,w}}^1 \\ n_0^2 n_0^2 \ldots n_0^2 & n_1^2 n_1^2 \ldots n_1^2 & \ldots & n_{m_k^{2,w}}^2 n_{m_k^{2,w}}^2 \ldots n_{m_k^{2,w}}^2 \end{bmatrix} \tag{C.43}$$

   Repetiremos este paso para cada iteración, de manera que se realicen tantas iteraciones como cámaras(de las que obtengamos datos) existan.
   Las matrices de combinaciones pueden llegar a tener un tamaño demasiado grande, de manera que este método no pueda ser llevado a la práctica. Por ello limitaremos el número de

combinaciones y en cada iteración se guardarán solo las mejores combinaciones. Calcularemos la suma de distancias de cada combinación, de manera que el resultado obtenido aunque no corresponda exáctamente con la probabilidad, será mucho más rápido de calcular. El resultado obtenido mediante este método, no obstante, será bastante parecido al obtenido calculando la probabilidad; sería bastante dificil que una combinación con la mejor probabilidad tuviera una suma de distancias demasiado grande.

3. Cálculo de la probabilidad a priori de cada combinación:
   Tras obtener la matriz de combinaciones, se realizará una selección final de las combinaciones que se van a tener en cuenta. Para ello se usará el criterio de la máxima probabilidad a priori para cada combinación $C^w_{k,y}$. Dicha probabilidad se calculará como el producto de la probabilidad de cada cámara ($\beta^{c,w}_{i,k}$), multiplicada por la probabilidad de la combinación de las n_c medidas de la fila r de la matriz Π^w_k ($p(\Pi^w_k[1,...,n_c;r])$); y multiplicada a la vez por la probabilidad de la hipótesis w en el instante anterior $k-1$ ($p(H^w_{k-1})$) desde la que se ha generado la hipótesis actual w en el instante actual k:

$$p(C^w_{k,r}) = p(H^w_{k-1}) \cdot p(\Pi^w_k[1,...,n_c;r]) \cdot \prod_b \beta^{c,w}_{i,k} \tag{C.44}$$

$$p(H^1_0) = 1 \tag{C.45}$$

$$\beta^{c,w}_{i,k} = \begin{cases} \frac{e^{c,w}_{i,k}}{b^{c,w}_k + \sum_{j=1}^{m^{c,w}_k}} & i = 1,\dots,m^{c,w}_k \\ \frac{b^{c,w}_{i,k}}{b^{c,w}_k + \sum_{j=1}^{m^{c,w}_k}} & i = 0 \end{cases} \tag{C.46}$$

$$\begin{aligned} e^{c,w}_{i,k} &= e^{-\frac{1}{2}\cdot(v^{c,w}_{i,k})^T\cdot(S^w_k)^{-1}\cdot v^{c,w}_{i,k}} \\ b^{c,w}_{i,k} &= (\tfrac{2\Pi}{\gamma})^{\frac{n_z}{2}} m^{c,w}_k c^{-1}_{n_z} \tfrac{1-P^c_D P_G}{P^c_D} \end{aligned} \tag{C.47}$$

   donde:

   $\beta^{c,w}_{0,k}$: Probabilidad de que ninguna de las medidas dentro del área de gating sea la correcta para la hipótesis w en la cámara c en el instante temporal k.

   P^c_D: Probabilidad de detección de la cámara c.

   P_G: Probabilidad de gating.

   n_z: Número de dimensiones de la medida c.

   c_{n_z}: Volumen de la hiperesfera unidad n_z-dimensional.

   Habrá casos en los que un tracker no sea capaz de capturar medidas dentro de su área de gating. En tal caso se guiará por las predicciones realizadas mientras su covarianza crece, de manera que el área de búsqueda del jugador aumente en cada instante(ya que si el área de búsqueda del tracker fuera similar probablemente no volvería a encontrarlo). Al aumentar el área de gating, es posible que el tracker capture medidas de sensores distintos, de manera que estén demasiado alejadas entre sí. Para evitar este problema definiremos una función a la que denominaremos función de coherencia, la cuál será aplicada a cada combinación de medidas, de manera que

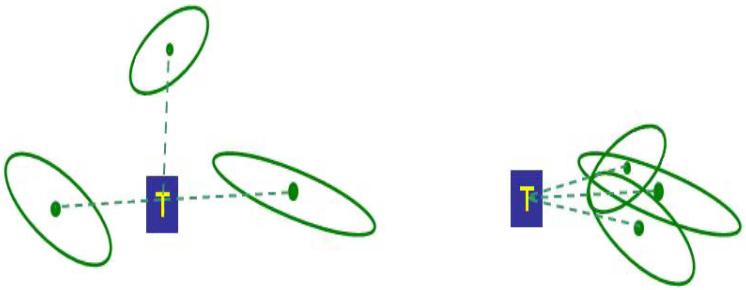

Figura C.12: Ejemplo en el que tres medidas están individualmente a la misma distancia de un tracker en dos casos bien diferentes. En el caso de la izquierda las 3 medidas están alejadas entre sí; y sin embargo en el caso de la derecha están cercanas entre sí. Parece lógico pensar que en el caso de la derecha es más probable que las medidas provengan del mismo jugador, de modo que debería considerarse dicha hipótesis como más probable.

multiplicará a la probabilidad calculada para cada combinación reduciendo dicha probabilidad en caso de que las medidas estén demasiado alejadas. Podemos ver un ejemplo de este tipo de problemas en la Figura C.12.

4. Generación de hipótesis:
   Una vez hayamos seleccionado las combinaciones de medidas que tengan una mayor probabilidad a priori(las cuáles han sido obtenidas a partir del conjunto de hipótesis H_{k-1}^{w} en el instante $k-1$), dichas combinaciones se convertirán en las nuevas hipótesis $H_{k}^{w'}$ en el instante k. El número de hipótesis generadas será mayor o igual al número de hipótesis de partida, ya que de cada hipótesis en el instante $k-1$ siempre se generan una o más hipótesis en el instante k.
   Cada hipótesis estará representada por un vector de medidas, una de cada cámara, asociados a la r-ésima combinación de medidas:

$$C_{k,r}^{w} = z_{i,k}^{c,w} \equiv z_{k}^{c,w'} = H_{k}^{w'} \tag{C.48}$$

   En las hipótesis generadas se mantendrán los valores de la predicción de estado, de medida, de covarianza de estado y de covarianza de medida (además de la innovación respecto de las medidas seleccionadas) respecto a las hipótesis de partida.

$$\begin{array}{lll} \hat{x}_{k|k-1}^{w'_t} = \hat{x}_{k|k-1}^{w_t} & \hat{z}_{k|k-1}^{w'_t} = \hat{z}_{k|k-1}^{w_t} & \\ \hat{P}_{k|k-1}^{w'_t} = \hat{P}_{k|k-1}^{w_t} & \hat{S}_{k|k-1}^{w'_t} = \hat{S}_{k|k-1}^{w_t} & v_{k}^{c,w'_t} = v_{k}^{c,w_t} \end{array} \tag{C.49}$$

5. Renormalización conjunta de hipótesis:
   Finálmente renormalizaremos las nuevas hipótesis, ya que la que influencia mutua que ejercen unos trackers sobre otros podría hacer que varios trackers intentaran seguir al mismo objeto. Sino realizamos esta corrección es bastante probable que cada tracker acabe siguiendo a varios jugadores usando diferentes hipótesis para cada uno de ellos, de manera que permanezcan

siguiéndolos a todos ellos por un tiempo indeterminado haciendo imposible la toma de decisión retardada. Otro caso indeseable que puede suceder, es que si se produce un cruce al final solo sobrevivan las hipótesis de ambos trackers las cuáles se dirijan al jugador con mejores medidas. Para evitar estos casos se creará un algoritmo el cuál rebaje las probabilidades de las hipótesis menos probables que siguen a medidas que otros trackers están siguiendo con hipótesis más probables. Para ello, en un principio calcularemos para cada tracker, la suma de las probabilidades de las hipótesis que usan cada una de las medidas, obteniendo para cada medida:

$$M_{j,k} = \{M_{j,k}^t\} = \sum_{w'} p(H_k^{w'}) \quad \forall z_k^{w'} \in H_k^{w'} \tag{C.50}$$

Acto seguido, se calcularán las probabilidades para cada hipótesis, dividiéndolas por el mayor cociente, para cada una de sus medidas, entre la probabilidad de la hipótesis más probable para esa medida y su propia hipótesis. De esta manera la hipótesis de mayor probabilidad no modificará su valor y el resto disminuirán el suyo proporcionalmente a la diferencia de probabilidades. Por tanto, la nueva probabilidad de cada hipótesis será:

$$p'(H_k^{w'}) = \frac{p(H_k^{w'})}{\underset{z_k^{w'} \in H_k^{w'}}{\max} \left\{ \frac{\max_{t'}\{M_{j,k}^{t'}\}}{M_{j,k}^t} \right\}} \tag{C.51}$$

Después de esto se eliminarán aquellas hipótesis cuya probabilidad esté por debajo de un umbral. En cualquier caso, la hipótesis de que ninguna de las medidas es correcta no deberá ser eliminada, aunque esté por debajo del umbral. Para seleccionar que hipótesis serán eliminadas calcularemos los coeficientes ${\beta'}_k^{c,w'}$; los cuáles serán usados en la fase de estimación de estado como pesos:

$${\beta'}_k^{c,w'} = \frac{e^{c,w'}}{\sum_c e^{c,w'}} \tag{C.52}$$

Por último normalizaremos las probabilidades de las hipótesis de nuevo, teniendo en cuenta tan sólo las seleccionadas:

$${p'}_N(H_k^{w'}) = \frac{p'(H_k^{w'})}{\sum_{w'} p'(H_k^{w'})} \tag{C.53}$$

C.4.4. Estimación

Una vez hayamos realizado la asociación de datos pasaremos a la fase de estimación. Para aquellas hipótesis en las que no ha sido asociada ninguna medida con el tracker, la estimación tanto de estado como de covarianza de estado, será igual a la predicción:

$$\hat{x}_k^{w'} = \hat{x}_{k|k-1}^{w'} \qquad \hat{P}_k^{w'} = \hat{P}_{k|k-1}^{w'} \tag{C.54}$$

Para el resto de las hipótesis la estimación será calculada a partir de la predicción y de las medidas que hayan sido asociadas. En primer lugar se calculará la ganancia:

$$G_k^{w'} = \hat{P}_k^{w'} \cdot H^T \cdot (S_k^{w'})^{-1} \tag{C.55}$$

Por otra parte, la innovación total será igual a la suma de innovaciones de cada cámara ponderadas por los pesos que hemos calculado anteriormente:

$$\hat{v}_k^{w'} = \sum_c \beta'^{c,w'}_k \cdot \hat{v}_k^{c,w'} \tag{C.56}$$

Finálmente, podremos calcular la estimación de estado y covarianza de esta manera:

$$\begin{aligned} \hat{x}_k^{w'} &= \hat{x}_{k|k-1}^{w'} + G_k^{w'} \cdot v_k^{c,w'} \\ \hat{P}_k^{w'} &= \hat{P}_{k|k-1}^{w'} - G_k^{w'} \cdot S_k^{w'} \cdot (G_k^{w'})^T + \tilde{P}_k^{w'} \end{aligned} \tag{C.57}$$

Llamaremos dispersión de las innovaciones a $\tilde{P}_k^{w'}$. Si nos fijamos en la distancia entre las predicciones y las medidas estaremos ignorando las distancias entre sí de las medidas seleccionadas en la combinación. Para corregir esto crearemos esta variable:

$$\tilde{P}_k^{w'} = G_k^{w'} \cdot \left[\sum_c \beta'^{c,w'}_k \cdot v_k^{c,w'} \cdot (v_k^{c,w'})^T - v_k^{c,w'} \cdot (v_k^{c,w'})^T \right] \cdot (G_k^{w'})^T \tag{C.58}$$

C.4.5. Probabilidad a posteriori y agrupamiento de hipótesis

En nuestro caso, consideraremos que las hipótesis con menos variaciones de aceleración serán más probables. Para ello podemos obtener una probabilidad a posteriori para cada hipótesis, la cuál esté basada en el cálculo de la aceleración sufrida en los últimos fotogramas. De este modo, definiremos una función de probabilidad f_A a la que llamaremos función de aceleración, la cuál representará la probabilidad de que se dé una determinada aceleración. Tras aplicar dicha función, tendremos que renormalizar las probabilidades de las hipótesis de nuevo.

$$p''(H_k^{w'}) = p'_N(H_k^{w'}) \cdot f_A(H_k^{w'}) \tag{C.59}$$

$$p''_N(H_k^{w'}) = \frac{p''(H_k^{w'})}{\sum_{w'} p''(H_k^{w'})} \tag{C.60}$$

Por otra parte, al final de cada iteración, también puede ser interesante el hecho de agrupar las hipótesis cuyos estados sean demasiado cercanos; ya que es bastante probable que dichas hipótesis obtengan resultados en paralelo, de manera que no se aproveche la ventaja de usar múltiples hipótesis. Para unir

hipótesis cercanas se usará un agrupamiento jerárquico en cada tracker a partir del estado estimado. Al fusionar dos hipótesis se considerará como válida la trayectoria de la más probable; de manera que su probabilidad sea igual a la suma de las probabilidades de ambas hipótesis.

Bibliografía

[1] J.R. Gómez, *Detección, segmentación y seguimiento de personas en un entorno multicámara: Aplicación en el ámbito deportivo*, Universidad de Zaragoza, Zaragoza (Abril 2010).

[2] S.S. Cheung and C.Kamath, *Robust techniques for background substraction in urban traffic video*, In Proc. Video Communications and Image Processing. vol. 5308 SPIE Electronic Imaging , San Jose, CA, U.S.A., doi:10.1007/12.526886 (Jan. 2004).

[3] I. Haritaoglu, D. Harwood and L.Davis, *W4s: A real time system for detecting and tracking people in 2 1/2d*, European Conference in Computer Vision, vol. LNCS/1406, pp.877-892, doi:10.1007/BFb0055710 (1998).

[4] E.Herrero, C.Orrite and J.Senar, *Detected motion classification with a double-background and a neighborhood-based difference*, Patter Recognition Letters, vol.24. no.12, pp. 2079-2092 (2003).

[5] T.Misu, M.Naemura, W.Zheng, Y.Izurni and K.Fukui, *Robust tracking of soccer players based on data fusion*, In Proc. 16th International Conference on Pattern Recognition (ICPR), vol.1, pp.556-561, doi:10.1109/ICPR.2002.1044792 (Aug. 2002).

[6] M.M.Jlassi, A.Douik, T.Battikh and M.Annabi, *Synthesis of classification supervised algorithms for players identification during a sports meeting*, In Proc. 14th IEEE International Conference on Electronics, Circuits and Systems(ICECS) (2007).

[7] M.Xu, J.Orwell and G.A. Jones, *Tracking football players with multiple cameras*, In Proc. International Conference on Image Processing, vol.5, pp. 2909-2912 (Oct. 2004).

[8] S.Iwase and H.Saito,*Tracking soccer players based on homography among multiple views*, Proceedings of SPIE, vol. 5150 (Visual Communications and Image Processing), pp. 283-292, doi:10.1117/12.502967 (Jun.2003).

[9] R.E.Kalman, *A new approach to linear filtering and prediction problems*, Transaction of the ASME-Journal of Basic Engineering - D, pp. 35-45(1960).

[10] J.J. La Viola, *A comparison of unscented and excented kalman filtering for estimating quaternion motion*, In Proc. American Control Conference, pp. 2435-2440 (2003).

[11] E.A. Wan and R.Van der Merwe. *The unscented kalman filter for nonlinear estimation*, In Proc. IEEE Adaptative Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC), pp.153-158, Lake Louise, AB, Canada, doi:10.1109/ASSPCC.2000.882463 (2000).

[12] A.Doucet, *On sequential simulation-based methods for bayesian filtering*, Technical Report CUED/F-INFENG/TR 310, Cambridge University, Department of Engineering , Cambridge, U.K. (1998).

[13] I.N.Junejo, O.Javed, and M.Shah,*Multi Feature path modeling for video surveillance*, In Proc. International Conference on Pattern Recognition(ICPR), vol.2, pp. 716-719(2004).

[14] S.C. Jhonson, *Hierarchical clustering schemes*, Psychometrika, vol.32, no.3, pp.241-254, doi:10.1007/BF02289588 (Sep. 1967).

[15] J.R.Gómez, J.E.Herrero, M.Montañes, F.Martínez, and C.Orrite,*Detection and classification of football players with automatic generation of models*, Optical Engineering, vol. 49, no. 1, pp. 017005-017005-17, doi:10.1117/1.3294885 (Jan. 2010).

[16] A.Criminisi, I.Reid, and A.Zisserman, *A plane measuring device*, Image and Vision Computing, vol.17, no.8, pp. 625-634, doi:10.101016/S0262-8856(98)00183-8 (1999).

[17] G.Welch and G.Bishop, *An introduction to the Kalman filter*, Technical Report TR 95-041, Computer Science, UNC Chapel Hill(1995).

[18] J.R. Gómez, J.E. Herrero, C. Medrano, and C.Orrite, *Multi-Sensor system based on unscented kalman filter*, In.Proc Sixth IASTED international Conference on Visualization, Imaging and Image Processing (VIIP), Palma de Mallorca, Spain, pp-13-18 (Aug.2006).

[19] P.Willet, Y.Ruan. and R.Streit, *The pmht: Its problems and some solutions*, IEEE Transactions on Aerospace and Electronic Systems, vol.38 no.3, pp.738-754(Jul.2002).

Índice de figuras