UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

Seguimiento de jugadores para detección de eventos en vídeos deportivos

INGENIERÍA SUPERIOR DE TELECOMUNICACIONES
ESPECIALIDAD EN SISTEMAS Y REDES

Autor:
JOSE MANUEL MORENO GARCÍA

Tutor:
Dr. JESÚS CID-SUEIRO

Leganés, a 21 de Mayo de 2010

A mis abuelos Milagros y Manolo

Hace ya 8 años que comencé esta aventura con la mayor de las ilusiones. He de decir que, aunque ha sido más difícil de lo que por aquel entonces pensaba, en ningún caso me he arrepentido de tomar la decisión que tomé en aquel momento. Y gran parte de que tenga esta seguridad la tienen todas las personas que me rodean y que me han hecho sentir arropado en todo momento.

Me gustaría comenzar los agradecimientos con mi familia y especialmente con mis padres, mi hermana y mis abuelos. Por estar ahí, tanto en los buenos como en los malos momentos. Por apoyarme independientemente de la decisión que tomara, ya fuera buena o mala. Por aconsejarme, tanto en las decisiones sencillas como en las importantes. Por conseguir que terminar la carrera fuera un objetivo de todos y no solo mío. Por no dudar cuando yo sí que lo hice durante el primer año de carrera. Simplemente por darme toda vuestra ayuda durante estos largos 8 años. Sin vosotros esto no habría sido posible. Gracias por confiar en mí.

Me gustaría aprovechar estas líneas tan importantes para mí para decirle a mi hermana Cristina que no deje pasar la oportunidad que tiene y que la aproveche. Como consejo le diría que todo esfuerzo tiene su recompensa.

También dar las gracias a Zaida por su apoyo incondicional. Por todo lo que hemos pasado juntos desde que nos conociéramos en el instituto y lo que hemos aprendido el uno del otro. Por aguantar mis cabreos en épocas de exámenes y por entender que, en algunos momentos, decidiera estudiar en lugar de verla. Son muchas las alegrías que hemos vivido juntos y esta no deja de ser una de las que nos quedan por vivir. También agradecer a sus padres y su hermana todo el cariño que he sentido durante todos estos años. Hace ya mucho tiempo que forman parte de mi familia.

A mis amigos, los que siempre han estado ahí. Los que me han dado siempre mucha más ayuda de la que realmente les pedía. Los que han estado a mi lado en los malos momentos y han conseguido que saliera adelante. Los que en septiembre del año pasado me animaban diciendo que "Microondas" solo era un aparato para calentar la comida y no una asignatura. Muchos de ellos saben lo que siento en estos momentos porque han pasado por lo mismo. Negaré en todo momento que yo he escrito estas palabras sobre vosotros.

Mis compañeros de la universidad. Estoy orgulloso de que muchos de ellos se hayan sentido identificados en el párrafo anterior. Y es verdad, podría haberlos incluido en esa parte. Sin embargo, un día comenté que determinadas historias iban a ser, como mínimo, nombradas en los agradecimientos. Estas historias tienen diversos nombres y protagonistas. Algunas de ellas son *¿Dónde está el cable rojo?*, *Dale con la cámara ya verás como sale brillo*, *Cómo ir a por componentes electrónicos y acabar en la feria de abril* ó *Yo medí los ángulos de la carta de Smith con el transportador*. No me gustaba la idea de nombrar a nadie en concreto pero he vivido muchas cosas con Fer y Fran y no estaría del todo satisfecho si no los nombrara. A ambos les diría que todo el esfuerzo que hicimos ha merecido la pena (¡Ánimo Fer!)

Por supuesto agradecer también a Jesús que aceptase dirigirme en este proyecto. También que tuviera la paciencia de aguantar todas las citas que he cancelado con él durante este año y medio pero sobre todo por toda la ayuda aportada durante este tiempo. Gran parte de este trabajo es tuyo también. Gracias.

Título: SEGUIMIENTO DE JUGADORES PARA DETECCIÓN DE EVENTOS EN VIDEOS DEPORTIVOS.

Autor: JOSE MANUEL MORENO GARCÍA

Tutor: DR. JESÚS CID-SUEIRO

La defensa del presente Proyecto Fin de Carrera se realizó el día 21 de Mayo de 2010, siendo calificada por el siguiente tribunal:

Presidente: EMILIO PARRADO HERNÁNDEZ

Vocal: JERÓNIMO ARENAS GARCÍA

Secretario: PATRICIA ARIAS CABARCOS

Habiendo obtenido la siguiente calificación:

Presidente: Vocal: Secretario:

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

ÍNDICE DE TABLAS

1. RESUMEN

1.1 Resumen

El objetivo principal marcado en este proyecto es conseguir obtener un algoritmo automático de seguimiento de jugadores sobre vídeos de partidos de fútbol en formato televisivo. El resultado obtenido en este algoritmo podría ser usado, junto a los resultados de otros algoritmos, para una posterior detección de eventos de alta importancia ("*highlights"*) de cara al procesado y etiquetado de estos vídeos de forma automatizada.

A modo de introducción se va a repasar un poco la historia y origen del fútbol y el estado actual del deporte para conocer, de primera mano, el terreno sobre el que se va a trabajar. También vamos a comentar la motivación que nos lleva al estudio de este proyecto y los objetivos que nos marcamos inicialmente.

Después continuaremos marcando las tendencias actuales del estado del arte tanto de forma genérica, desde el estudio de las diferentes características usadas para el procesado y el etiquetado automático de los videos, como de forma particularizada, con la situación actual de los algoritmos de seguimiento de jugadores.

Una vez hayamos entendido las motivaciones y objetivos que nos llevan a realizar este proyecto, nos meteremos directamente en el estudio del algoritmo propuesto por nosotros. Comenzaremos explicando de forma generalizada la arquitectura del algoritmo y las partes en que vamos a dividir el desarrollo del mismo. Durante los primeros capítulos se detallarán los problemas que nos vamos a encontrar al usar vídeos adquiridos directamente de la retransmisión televisiva, de que forma vamos a obtener estos vídeos para lograr obtener una base de datos sólida sobre la que realizar pruebas y como vamos a conseguir utilizar estos vídeos sobre un software específico.

Posteriormente se especificará con el mayor rigor posible el algoritmo de seguimiento desarrollado por nosotros. Esta parte estará compuesta de tres bloques básicos siguiendo una linealidad en el desarrollo:

- Segmentación de los frames, donde obtendremos a partir de cada frame original un frame alternativo en el que distingamos dos partes: el césped y el resto de la imagen. El objetivo principal de este bloque es obtener las figuras de los jugadores sobre el césped y así poder localizarlos.

- Detección de los jugadores, mediante procesado de la imagen obtenida en el punto anterior y con apoyo en algoritmos del software utilizado obtendremos las posiciones de los jugadores en el frame.

- Seguimiento de los jugadores, con el desarrollo de un algoritmo que calcula los movimientos más probables entre frames de los jugadores.

Una vez explicados los algoritmos, los problemas que nos hemos encontrado y las decisiones que se han tomado procedemos a evaluar los resultados. Como se podrá ver en este apartado, la evaluación de funcionamiento no ha sido la habitual ya que era inviable obtener unos resultados de forma automatizada y se ha tenido que detallar los

resultados de forma subjetiva. Para este apartado se han usado fragmentos de vídeos distintos al usado como vídeo de pruebas y los resultados obtenidos han sido muy satisfactorios obteniendo un porcentaje de acierto en el seguimiento por encima del 80% en la mayoría de los casos. También se han citado los problemas más habituales y de que forma nos afectan en los resultados.

Una vez mostrados los resultados obtenidos de forma numérica mostraremos las conclusiones obtenidas con la realización de este proyecto tanto a nivel de estudio y desarrollo como a nivel personal. También se darán unas pinceladas sobre las líneas futuras a seguir para posteriores estudios sobre lo desarrollado por nosotros.

Finalmente, se realizará un presupuesto aproximado del coste que nos hubiera supuesto la creación de este proyecto de forma profesional y comercial, teniendo en cuenta tanto el gasto material como la remuneración profesional.

2. INTRODUCCIÓN

2.1 Introducción

El fútbol. El deporte rey por excelencia sobre todo en gran parte de Europa y Sudamérica aunque practicado y visionado en todo el mundo. Este deporte implica a 270 millones de personas en todo el mundo [*1*] (estudio realizado en el año 2006) siendo esta cantidad un 4% de la población mundial. Viendo estos números, nos podemos hacer una idea de la importancia de este deporte en la actualidad pero, volvamos a los inicios de este deporte.

2.1.1 El origen del fútbol

Para remontarnos al origen del fútbol debemos irnos muy atrás en la historia, hacía finales de la Edad Media y sus siglos posteriores. Durante esta época se desarrollaron en las Islas Británicas lo que entonces se conocía como "Códigos de Fútbol", en los que se detallaban las reglas a seguir para practicar este deporte. Fue en el siglo XVII cuando se realizaron las primeras reglas unificadas.

Durante esta época hubo una gran agitación en los colegios británicos por el posicionamiento que tenían frente a como entenderlo: fútbol con la mano (rugby) o fútbol con el pie. Fue a mediados del siglo XIX cuando se comenzaron a unificar todas las reglas en el código Cambridge (1848) en el que se limitó el deporte al uso del píe y cuyas reglas eran similares a las actuales.

Sin embargo, se considera el 26 de Octubre de 1863 como el día en el que se creó el fútbol moderno siendo el 30 de Noviembre de 1872 cuando se disputó el primer partido (Escocia e Inglaterra) de selecciones con las reglas modernas.

2.1.2 La expansión del fútbol

Desde entonces, el interés por el fútbol se extendió a los demás países, creándose en 1904 la FIFA (*Fédération Internationale de Football Association*), asociación que actualmente perdura.

Durante la Primera Guerra Mundial, el crecimiento del fútbol se paralizó pero con la realización de los juegos olímpicos posteriores el deporte se reactivó, jugándose en 1930 la primera copa mundial de selecciones.

También en la Segunda Guerra Mundial ocurrió algo similar, resurgiendo con un partido amistoso en 1947 entre la selección británica y un combinado de estrellas europeas en el que la recaudación fue destinada a reactivar la FIFA. También parte de este resurgimiento fue gracias a copa mundial de 1950 que se jugó en Brasil con el famoso "Maracanazo" en el que la selección uruguaya ganó a la brasileña en su estadio la final de la copa del mundo.

2.1.3 El afianzamiento del fútbol

La mitad del siglo XX fue la época de consolidación del fútbol moderno con la creación de la UEFA (*Union of European Football Associations*) y las confederaciones continentales. Después llegaron las diferentes copas a nivel de continentes (incluida la europea) y los primeros campeonatos europeos a nivel de club.

Todo esto hizo que el fútbol se afianzara como el deporte por excelencia siendo la copa mundial de fútbol el evento deportivo más importante estando incluso por delante de los juegos olímpicos.

2.1.4 El fútbol en la actualidad

Con esta pequeña introducción se ha querido dotar al lector de la importancia que tiene en la actualidad el fútbol y todo lo que le rodea. Para hacernos una idea, según la LFP (*Liga de Fútbol Profesional en España)* el fútbol aporta a la economía española 9.000 millones de euros (siendo el 3.45% del PIB en España [2]) y genera 85.000 empleos directos e indirectos. Viendo estos datos, es fácil entender por qué cualquier tipo de estudio, evolución o innovación relacionado con este deporte puede tener tanto interés, tanto a nivel económico como social. Y también nos hace entender que el fútbol haya evolucionado de deporte a espectáculo deportivo en el que se busca la diversión y deleite del espectador.

Si nos centramos en el espectáculo, un aspecto importante a tener en cuenta es la cantidad de personas que lo contemplan vía digital, ya sea por televisión o a través de Internet. Si nos ceñimos simplemente a España y, a modo de ejemplo, podemos ver que la final de la copa de Europa del año 2009 fue vista por 11.310.000 personas y la final de la Eurocopa 2008 por 14.482.000. Con estas cifras, se puede llegar a la conclusión de que, aparte de espectáculo, el fútbol se ha convertido también en un negocio. Y aquí es donde colocamos nuestro punto de partida para exponer los motivos que nos han llevado a realizar este proyecto.

2.2 Motivación

Cada vez hay más televisiones que ofrecen fútbol de pago y en las que el usuario ha dejado de ser un mero espectador. El usuario quiere más. Quiere saber cuantos kilómetros recorre el lateral derecho de su equipo o cuanta posesión ha tenido el contrario en los primeros 30 minutos de partido. Quiere ver un resumen del partido instantes después de finalizar o cuantas veces remata el delantero de su equipo desde el área pequeña.

Realizar esta serie de acciones de forma manual se hace inviable por lo que el procesado digital de dichos partidos se antoja como una opción a tener en cuenta. Es imposible que una persona pueda calcular cuantos kilómetros recorre un jugador en un partido, pero sí podría desarrollarse un algoritmo digital que lo calculara mediante el procesado del vídeo del partido. También podría determinarse un resumen automático del partido procesando las imágenes y obteniendo aquellas jugadas que cumplan con un patrón preestablecido. Estos procedimientos no parecen sencillos de realizar, pero sí más viables que obtener esta información de forma manual.

En la actualidad, numerosas cadenas ofrecen ya este tipo de estadísticas. Podemos comprobar como la cadena de televisión española "La Sexta" [3] ofrece una gran cantidad de estadísticas de este tipo en directo en sus retransmisiones.

Imagen 1 - Ejemplo estadísticas[1]

Este tipo de servicios están en auge, ya que la mayoría de las cadenas que ofrecen retransmisiones deportivas quieren contar con ellos.

Una de las primeras empresas que se dedicó a dar este servicio fue SUP [4] (Sport Universal Process) con su programa Amisco. Este software necesita 8 cámaras enfocadas al campo para poder ofrecer su servicio que se basa en 4 etapas:

- Captura de imágenes
- Modelado del terreno de juego
- Análisis de imágenes
- Almacenamiento de datos

[1] Imágenes obtenidas de www.futbolsexta.com/estadisticas

Este servicio es utilizado por varios equipos de fútbol de alto nivel (Real Madrid, Chelsea, Liverpool, etc.) que lo utilizan para realizar un seguimiento profesional del movimiento de sus propios jugadores y muchas más estadísticas con un software adicional muy completo. En la imagen podemos ver un seguimiento de las líneas de defensa de ambos equipos.

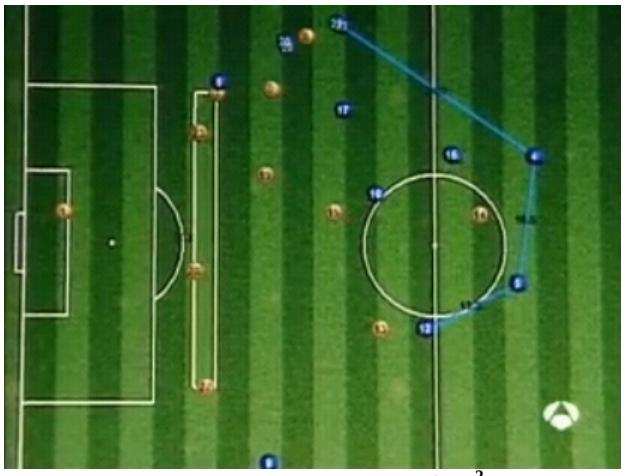


Imagen 2 - Programa Amisco[2]

Dentro de las estadísticas habituales y a modo de ejemplo, en las retransmisiones de la UEFA Champions League se lleva incluyendo desde el año 2008 el estudio de los kilómetros recorridos por cada jugador durante el partido mediante un sistema de tracking. Este servicio es ofrecido por la compañía italiana Delta Tre [5] mediante el uso de tecnología militar basado en el seguimiento de misiles (missile-tracking).

Se basa en la instalación de 16 cámaras por el estadio y a una distancia mínima de 15 metros de altura para poder conseguir un tracking con un margen de error muy pequeño. Además, para poder realizar el servicio es necesario que 2 operadores estén realizando comprobaciones durante el partido.

Imagen 3 - Cámaras Delta Tre[3]

Ambas empresas basan todo su potencial en la utilización de cámaras de alto nivel (y alto coste) para poder realizar un seguimiento de todos y cada uno de los jugadores

[2] Imagen obtenida del video "Amisco a fondo Antena 3" en youtube.com
[3] Imagen obtenida de sport.es

(*tracking*). Esta estadística se antoja demasiado costosa para la información que aporta ya que conocer los kilómetros recorridos por cada jugador o por el equipo al completo es poco más que una curiosidad. Sobre todo teniendo en cuenta que recorrer más o menos kilómetros no te hará ganar el partido.

Si nos basamos en el post-procesamiento seguido por la compañía SUP en su programa Amisco, podemos observar que su método de detección de cada uno de los jugadores es manual [6]. Es decir, para seguir a cada jugador van pinchando en las piernas de ese jugador durante el tiempo que dure el encuentro en cada una de las cámaras que tienen. Esto les supone, en palabras de la propia empresa, unas 8 horas por jugador. Si a esto le añadimos que son 22 jugadores en el campo, saldrían 176 horas, lo que supone un mes de trabajo.

Imagen 4 - Detección manual del tracking Amisco[4]

En esta primera parte podemos comprobar como el seguimiento de los jugadores podría aplicarse en un sistema real. Se ha mostrado como muchos equipos de fútbol profesional utilizan este tipo de servicios y como varias empresas obtienen un beneficio de ello a pesar de necesitar un hardware específico y muchas horas de trabajo. Por lo tanto, podemos hacernos una idea del interés que puede suscitar realizar un sistema de seguimiento de jugadores y del beneficio comercial que se podría obtener de él. Pero, como acabamos de comentar, es necesaria una maquinaria muy concreta y una gran carga de trabajo humano para poder obtener unos resultados óptimos, que serán necesarios ya que el sistema será probado por un cliente y nuestro beneficio económico dependerá de la calidad del sistema desarrollado. Y la calidad en este caso es crítica.

Dejando de lado el uso comercial que se le podría dar al sistema de tracking en algunos casos, este proyecto va a ir enfocado a la realización de otro tipo de sistemas de tracking en el que la calidad de los resultados sea menor pero que sean usados en aplicaciones en los que la calidad no sea crítica. Una de las aplicaciones a la que nos podríamos estar refiriendo sería la detección de eventos en partidos de fútbol para indexado automático de videos. Y para la realización de este tipo de tracking actualmente se investigan dos alternativas:

Imagen [4] Imagen obtenida del video "Amisco a fondo Antena 3" en youtube.com

Como primera alternativa sobre la que se trabaja es el procesamiento “en bruto” de las imágenes usadas en la televisión. En un partido de fútbol debe haber entre 5-10 cámaras y es el realizador el que decide en cada momento que vista se retransmite. Sin embargo, todas estas cámaras siguen grabando por lo que si se pudiera obtener la retransmisión de varias de ellas (siempre que fueran cámaras lejanas) se podría llevar un seguimiento bastante completo. Debemos tener en cuenta que nunca tendrá tanta calidad como el anterior ya que nunca vamos a ver los 22 jugadores a la vez (el seguimiento de muchos de ellos se pierde).

Y ya como tercera opción llegamos a la motivación de este proyecto que consiste en la detección de los movimientos de los jugadores por medio de la retransmisión propia de una televisión. Esta opción consistirá en un detector automatizado basado en el procesamiento de frames que forman el vídeo. Habrá que tener en cuenta que este seguimiento estará limitado y nunca llegará a obtener los mismos resultados que en ninguna de las otras dos opciones ya que los medios utilizados son menores.

2.3 Objetivos

Como primera toma de contacto, se va a intentar realizar un sistema de seguimiento de los jugadores por el campo mediante el procesamiento del vídeo de la retransmisión del partido.

Para ello, vamos a necesitar la propia retransmisión del partido y un software adicional para procesar dichas imágenes, poder realizar el sistema de tracking y obtener una salida válida. En principio, este sistema solo podrá obtener el tracking de aquellos jugadores que aparezcan en la pantalla aunque un sistema mejorado del que vamos a obtener podría utilizar una sola cámara que abarcara todo el campo y poder realizar el seguimiento de forma completa.

Cómo retransmisiones de prueba, vamos a descargar de la red [7] una serie de partidos con los que podamos comprobar el funcionamiento. Estos partidos intentarán contener todas las posibles condiciones (día, noche, sombras, clima, etc.) de forma que se pueda hacer el mayor número de pruebas y cerciorarnos de que el sistema es lo más robusto posible.

La sucesión de objetivos de este proyecto es la siguiente:

1. Lectura de las retransmisiones: Este objetivo contemplará la comunicación entre el software utilizado y los partidos obtenidos. Para ello, es necesario el codec de DivX [8] ya que todos los partidos están codificados con este sistema de codificación.

2. Segmentación del campo: durante este proceso, se va a conseguir eliminar de las imágenes el color dominante (el césped) de forma que el posterior procesamiento sea mucho más sencillo.

3. Sistema de Tracking: este será el objetivo principal del proyecto y que explicaremos mucho más detalladamente en el resto de la memoria.

2.4 Estructura de la memoria

En este apartado, vamos a ver la estructura general del proyecto y una breve descripción de cada uno de los capítulos en los que está formado.

Los capítulos generales son los siguientes:

1. **Resumen.** En este capítulo se intenta dar una idea global del proyecto y del desarrollo del mismo.

2. **Introducción.** Se desarrolla una pequeña introducción histórica acerca del fútbol para situar al lector en el contexto actual. Este capítulo también explica la motivación que nos ha llevado a realizar este proyecto, los objetivos que se quieren conseguir y la estructura general de la memoria.

3. **Revisión del estado del arte.** Se realiza un breve estudio acerca del estado actual del arte. Este estudio engloba tanto la situación general como la situación específica en lo que se refiere a este proyecto.

4. **Algoritmos propuestos.** Este capítulo es el más extenso puesto que se explica al más alto nivel de detalle los algoritmos propuestos y las decisiones tomadas a la hora de realizar el estudio.

5. **Resultados experimentales.** Se defienden los resultados obtenidos y se intenta obtener una evaluación de la forma más objetiva posible.

6. **Conclusiones y líneas futuras.** Tras el apartado anterior, se llega a unas conclusiones y a unas posibles mejoras del sistema o estudios futuros.

7. **Presupuesto.** En este apartado, se realiza un posible presupuesto del proyecto.

8. **Bibliografía.** Referencias bibliográficas usadas en la realización del proyecto.

También y a nivel de índices se ha realizado un índice general, un índice de imágenes y un índice de ecuaciones. Se recomienda usar el índice general para un fácil acceso y lectura de los distintos apartados.

3. REVISIÓN DEL ESTADO DEL ARTE

3.1 Revisión del estado del arte

En este apartado se va a revisar el estado del arte en lo que se refiere a la extracción de características de bajo, medio y alto nivel de partidos de fútbol mediante el procesado de imagen y vídeo de dichos partidos de forma automatizada.

Se entiende por características de bajo nivel a las diferentes situaciones que se pueden dar en un partido de fútbol y que se pueden identificar de forma automatizada mediante el procesado de esta información. Como ejemplo, podemos incluir dentro de estas características el color dominante de la imagen (en este caso, el césped), el audio, los tipos de enfoque, etc. Estas características de bajo nivel van a carecer de valor semántico y van a ser utilizadas en conjunto para obtener las características de medio nivel.

Las características de medio nivel son aquellas que nos van a servir en algunos casos para obtener un valor semántico sobre el partido de fútbol. Estas características de medio nivel se pueden obtener directamente o mediante el procesado de las de bajo nivel. Dentro de las características de medio nivel podemos tener la detección del árbitro del partido o la propia detección del movimiento de los jugadores.

Son características de alto nivel las que nos van a dar en la mayoría de los casos un valor semántico que sea de utilidad para distintas acciones, como por ejemplo, la detección de un gol u de un penalti y que nos serviría, por ejemplo, para la indexación automática de los partidos.

Durante este capítulo vamos a intentar obtener un resumen del estado del arte en cuanto a la detección de estas características. Por un lado, se va a obtener una visión del estado del arte de forma más general y por otro lado, se revisará de una forma más específica en el uso y procesado de estas características para la obtención de un resultado de mayor utilidad: el movimiento relativo de los jugadores en el campo.

3.1.1 Revisión general del estado del arte

En la mayoría de los estudios realizados hasta la fecha, el objetivo principal ha sido el intentar obtener una serie de evidencias de bajo nivel que faciliten la extracción de características de alto nivel y generar un significado semántico.

Esto es, utilizando el caso que nos ocupa en este proyecto, obtener una serie de propiedades de bajo nivel como pueden ser el color dominante, la intensidad de imagen o el audio para realizar una detección de los principales eventos de forma automatizada. Dentro de los principales eventos se puede incluir los goles, los penaltis, saques de esquina o tarjetas amarillas y rojas.

Para la detección de eventos de bajo nivel, el tipo de algoritmo utilizado depende del tipo de evento a detectar ya que son muchos y muy variados. A modo de ejemplo, se van a citar los eventos de bajo nivel más utilizados en los estudios revisados:

- El evento de mayor uso en estos estudios es el **color dominante** ya que sirve de ayuda para poder diferenciar las distintas partes que no son césped y que serán de gran utilidad.
- Otro evento muy usado es la **intensidad de la imagen**, de forma que se podrá diferenciar que partes de la imagen tienen mayor detalle.
- También se pueden obtener las variaciones del **canal de audio**, ya que durante los eventos de alto nivel el volumen suele aumentar.

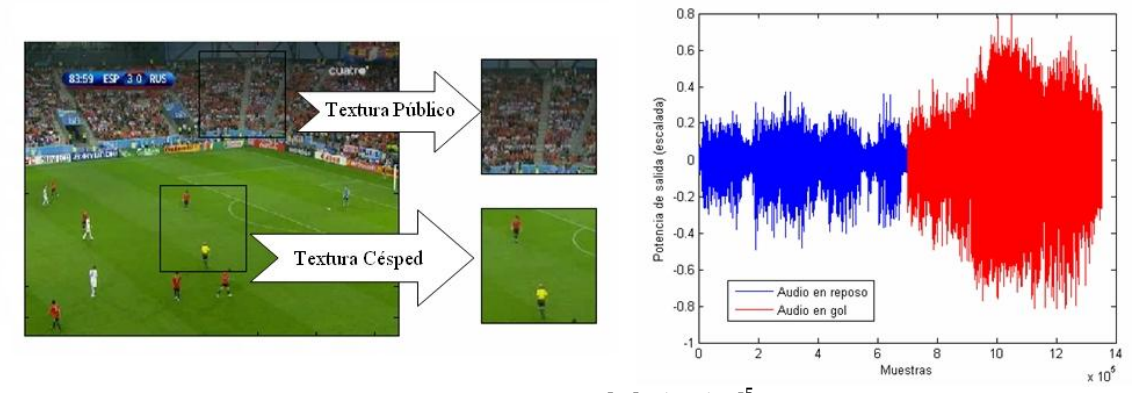


Imagen 5 - Eventos de bajo nivel[5]

En esta imagen podemos comprobar estos eventos de bajo nivel. En este caso, en la figura de la izquierda tenemos el color dominante y a la derecha el aumento de volumen que ocurre en el canal de audio con un gol.

Respecto a la detección de eventos de alto nivel, hay diferentes tipos de algoritmos a la hora de detectarlos:

- Algoritmos que usan varias características (portería, audio, texto, etc.) de bajo nivel unidas mediante una **red Bayesiana** [9].
- Algoritmos basados en obtener **propiedades visuales** de la imagen (como pueden ser posición de los jugadores, zona de juego habitual o los colores de los uniformes) [10].
- Algoritmos establecidos a partir de la posesión de la pelota y de la **correlación** de ésta con la vista general de la cámara [11].
- Algoritmos que utilizan **redes Bayesianas dinámicas** basadas en las redes Bayesianas y sus extensiones [12].

Estos son solo algunos de los tipos de algoritmos utilizados, hay muchos y muy diversos. Algo que en común en muchos de los estudios es la utilización de las redes bayesianas o diagramas de estados una vez que se han obtenido las distintas propiedades de bajo nivel.

[5] Imagen obtenida del video España-Rusia

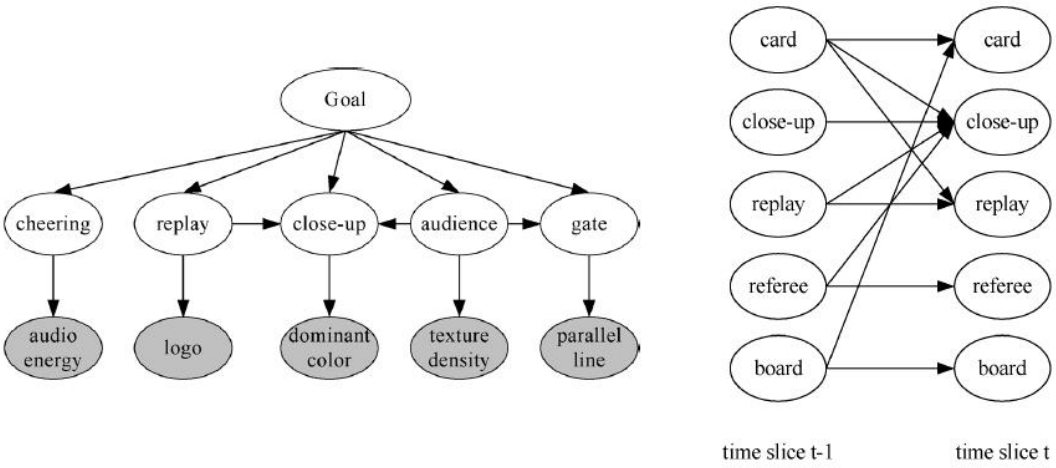
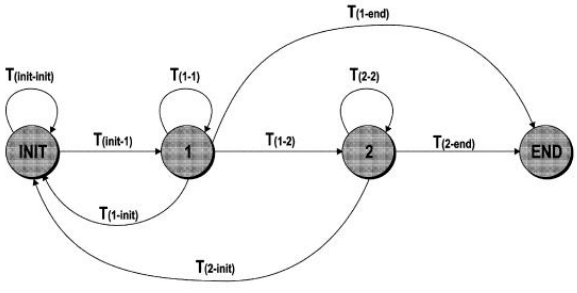


Imagen 6 - Red Bayesiana y red Bayesiana dinámica[6]

En esta imagen podemos ver un diagrama de estados en el que un gol estaría representado por los siguientes eventos de medio nivel:

- El nivel de aplausos aumenta, que se comprueba con el evento de bajo nivel de energía de audio.
- Se produce una repetición, un cierre de la imagen, se enfoca a la audiencia y vuelve a aparecer. Estos eventos de medio nivel se consiguen con la detección del logo, el color dominante y la densidad de la imagen.
- Representación de la portería, que se obtiene con la detección de líneas paralelas.

A la derecha podemos ver una transición de estados más recurrente en un partido de fútbol. Por ejemplo, después de una repetición, los eventos más comunes son la continuación de la repetición o una transición del video.



highlight / state transition	Shot on goal	Penalty Corner Free kick	Forward pass	Kick off	Turnover	Counterattack
$T_{(init-init)}$	$\neg T_{(init-1)}$	$\neg T_{(init-1)}$	$\neg T_{(init-1)}$	$\neg T_{(init-1)}$	$\neg T_{(init-1)}$	$\neg T_{(init-1)}$
$T_{(init-1)}$	$(\mu_1 < M < \mu_2) \wedge Z2$	$(M < \mu_1) \wedge Z1$	$\mu_1 < M < \mu_2$	$(M < \mu_1) \wedge Z7$	$Z1$	turnover
$T_{(1-end)}$	$(M < \mu_1) \vee (\neg Z1 \wedge \neg Z2)$	$\neg((M < \mu_1) \wedge Z1)$	$M < \mu_2$	$\neg((N > v) \wedge (P04))$	$\neg(Z1 \vee Z5 \vee Z6)$	$N > v$
$T_{(1-1)}$	$\neg(T_{(1-1)} \vee T_{(1-2)})$	$\neg(T_{(1-end)} \vee T_{(1-2)})$	$\neg(T_{(1-end)} \vee T_{(1-2)})$	$\neg(T_{(1-end)} \vee T_{(1-2)})$	$\neg(T_{(1-end)} \vee T_{(1-2)})$	$\neg(T_{(1-end)} \vee T_{(1-end)})$
$T_{(1-2)}$	$(M > \mu_2) \wedge Z2$	$N > v$	$Z6 \vee Z7$	not allowed	$Z5 \vee Z6$	not allowed
$T_{(1-end)}$	not allowed	not allowed	not allowed	$(N > v) \wedge (P04)$	not allowed	$(N < v) \wedge$ (forward pass)
$T_{(2-init)}$	$\neg Z1 \wedge \neg Z2$	$\neg(Z1 \vee Z2)$	$M < \mu_2$	not allowed	$\neg(T_{(2-end)})$	not allowed
$T_{(2-2)}$	$\neg(T_{(2-init)} \vee T_{(2-end)})$	$\neg(T_{(2-init)} \vee T_{(2-end)})$	$\neg(T_{(2-init)} \vee T_{(2-end)})$	not allowed	$\neg(T_{(2-init)} \vee T_{(2-end)})$	not allowed
$T_{(2-end)}$	$M < \mu_1$	$\neg(M < \mu_1) \wedge Z2 \wedge P01$ $\neg(M < \mu_1) \wedge Z2 \wedge P02$ $\neg(M < \mu_1) \wedge Z2 \wedge P03$	$Z1$	not allowed	$(\mu_1 < M < \mu_2) \wedge (N > v)$	not allowed

Imagen 7 - Diagrama de estados[7]

En esta imagen podemos ver una transición de estados temporales y su correspondiente tabla con las distintas transiciones y sus consecuencias.

[6] Imágenes obtenidas del estudio [12]
[7] Imagen obtenida del estudio [10]

3.1.2 Revisión específica del estado del arte: Movimiento relativo de jugadores

La mayoría de los estudios realizados hasta el momento sobre este aspecto en concreto tienen un funcionamiento muy similar y su diferenciación se basará en la forma de captura de la imagen o el procesado posterior. Pero si algo tienen en común todos los estudios revisados es **la segmentación del campo**.

El primer paso a tener en cuenta a la hora de detectar el movimiento relativo de los jugadores es la forma en que capturamos la imagen. Podemos realizar un tipo de captura económica con cámaras de bajo coste y baja resolución o cámaras de alto nivel como las usadas por la empresa Delta Tre. También es importante el número de cámaras usadas y los ángulos en los que se disponen. Podemos tener las habituales laterales, una única cámara o algunas en formato 'vista de pájaro'[13], aunque este caso es válido únicamente para deportes "indoor".

El siguiente paso es el proceso de segmentación en el que hacemos uso del color dominante de la escena. Para el caso que nos ocupa, el color dominante es el verde del césped aunque puede variar en función del deporte seleccionado. Los problemas habituales en esta parte suelen ser los mismos en todos los estudios:

- Umbral de decisión: este umbral es el que decide si un determinado píxel se puede considerar color dominante.
- Color dominante inicial: al comenzar la segmentación se debe elegir un color inicial para poder realizar la comparativa entre píxeles.

Como podremos comprobar más adelante, este proceso de segmentación es el más costoso a nivel temporal, ya que llevará asociado la mayor carga procesal de todo el algoritmo. Este es el mayor inconveniente que encuentran todos los estudios realizados hasta la fecha.

Una vez realizada la segmentación, el procesamiento posterior es muy diverso en función del estudio investigado. Algunos necesitan una calibración temporal y espacial inicial para unir las imágenes de las cámaras (si tiene más de una) mientras que otros se basan en predicciones y modelos de transición de estados. Otro de los problemas habituales con los que se encuentran es la superposición de los jugadores por lo que, en algunos casos, se vuelve a realizar un muestreo más exhaustivo para intentar paliarlo [14].

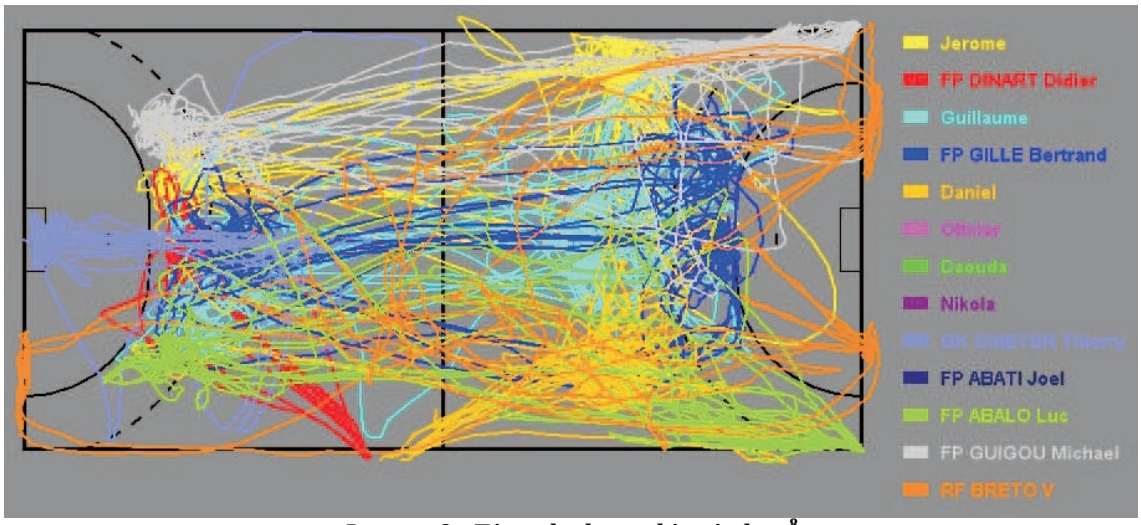


Imagen 8 - Ejemplo de tracking indoor[8]

También existen otro tipo de estudios en los que se usan las propiedades del tracking de los frames anteriores en el cálculo [15]. Para este caso concreto, en el cálculo del color del césped se usa el color dominante obtenido hasta ese frame y calculan una estimación del movimiento para el frame posterior. El esquema es el siguiente:

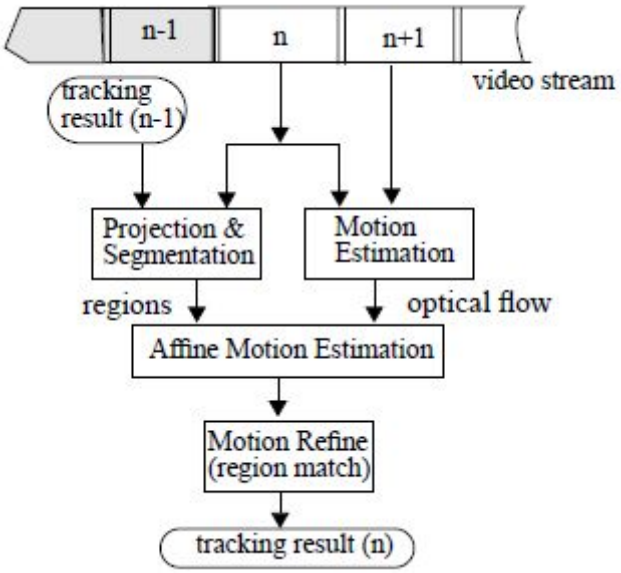


Imagen 9 - Esquema procesamiento frames[9]

En lo que respecta al cálculo del tracking sobre videos de televisión, también hay diferentes estudios hasta la fecha. Podemos encontrar estudios realizados con segmentación por color, detección por contornos y seguimiento similar a lo ideado por nosotros [16]. También podemos encontrar variantes del seguimiento como por ejemplo, seguimiento del balón en lugar de los jugadores [17][18] analizando la propia trayectoria del balón. También hay diversos estudios con métodos muchos más complejos como por ejemplo el que encontramos en [19] con fase de entrenamiento y test y uso de distribuciones de probabilidad para la detección y seguimiento de los jugadores.

[8] Imagen obtenida del estudio [13]
[9] Imagen obtenida del estudio [15]

Con estos ejemplos se ha podido observar que el ámbito del procesado de videos para la obtención del "tracking" no es algo novedoso, ya que realizando búsquedas en la web se encuentran numerosos y variados estudios hasta la fecha. También hemos podido comprobar que, en lo que respecta a la detección y seguimiento de jugadores, existen varios estudios con diversos métodos. Algunos de ellos incluso se aproximan con bastante claridad al algoritmo ideado por nosotros por lo que serán utilizados a modo de referencia tanto en problemas encontrados como en resultados obtenidos. En la siguiente imagen podemos ver un frame de un video tras la segmentación usada en [16].

Imagen 10 - Ejemplo de segmentación de [16][10]

Como podremos comprobar más adelante, esta imagen es prácticamente igual a las calculadas por nosotros en nuestro algoritmo de segmentación.

3.1.3 Conclusiones

Durante este capítulo se ha querido mostrar el estado del arte actual en lo que se refiere a la detección de eventos de distintos niveles (bajo, medio y alto) de partidos de fútbol. Se ha comenzado por el estado actual del arte de forma genérica y se ha continuado por el estado actual específico de sistemas de tracking automático, que es de lo que trata este proyecto.

En cuanto a la revisión general del estado del arte, este proyecto se enmarca tanto en la detección de eventos de bajo nivel con la segmentación del campo como en la detección de eventos de medio nivel con el seguimiento de los jugadores durante el partido. El uso de la segmentación del campo como sistema discriminante de eventos de bajo nivel no es nuevo, y además es usado en la mayoría de estudios investigados por lo que, en este aspecto, no se introduce un sistema novedoso.

Sin embargo, si nos centramos en el estado del arte de forma específica podemos comprobar que en este proyecto se va a intentar realizar un sistema de tracking automático en el que las imágenes son tomadas de la propia televisión. Esta es la característica principal que lo difiere de la mayoría de sistemas mostrados en el capítulo. Salvo en el último párrafo, en los estudios mostrados en el estado del arte las imágenes obtenidas del sistema de captura son fijas, tienen una única cámara global o son colocadas "a vista de pájaro". Esto hace que el sistema de seguimiento de los jugadores sea más fácil en comparación con el nuestro, ya que nuestra imagen depende de la realización de la televisión y tiene numerosos problemas que con las anteriores no tendríamos. Estos problemas serían, por ejemplo, el intercambio de planos largos y cortos, enfoques a las gradas, repeticiones y un largo etcétera.

[10] Imagen obtenida del estudio [16]

En el último párrafo, podemos comprobar que también existen estudios dedicados al seguimiento de jugadores o del balón mediante el procesado de frames provenientes de videos de televisión. Si comparamos nuestro sistema con los estudiados, podemos ver que el algoritmo es muy similar al utilizado en [16] (se muestra en la imagen 10) pero toma aspectos del [13] (con esquema en la imagen 9) en lo que se refiere a la estimación del movimiento. Podemos ver como a partir del vídeo vamos obteniendo la segmentación de la imagen y se realiza una estimación del movimiento. Con estos dos cálculos se realiza un refinamiento y se obtiene el tracking. Nuestro sistema es muy similar a este, pero en nuestra parte la estimación del movimiento se realiza teniendo en cuenta el movimiento de la cámara (problema que no tienen los sistemas de cámara fija).

Por lo tanto, podemos concluir que nuestro sistema no es novedoso en cuanto a la detección de movimiento de jugadores sobre vídeos sin ningún tipo de tratamiento ni necesidades específicas. Los resultados obtenidos serán de peor calidad en comparación con los estudios sobre cámaras fijas ya que las restricciones que nosotros ponemos son menores. Sin embargo, al existir otro tipo de estudios sobre cámara de televisión, tenemos una base sobre la que apoyarnos y realizar algún tipo de comparativa. De esta forma, podremos ver si nuestro algoritmo alternativo de seguimiento es válido y tiene la calidad suficiente para lograr resultados apreciables. Esta alternativa mostrada por nosotros pensamos que puede aportar bastante en el campo de la indexación y búsqueda de contenido en partidos de fútbol y por esto pensamos que merece la pena realizar este proyecto.

4. ALGORITMOS PROPUESTOS

4.1 Introducción

Durante este capítulo vamos a describir el algoritmo de tracking propuesto, los problemas encontrados y las soluciones tomadas. A modo de guión, vamos a seguir los puntos marcados en los objetivos citados anteriormente para poder seguir un orden lógico. Como primer paso se va a mostrar la arquitectura general del sistema para después detallar cada uno de los distintos bloques en apartados posteriores.

Como ya se comentó en los objetivos del proyecto, ha sido necesaria una base de datos formada por una serie de partidos para poder realizar las pruebas oportunas y hacer una evaluación aceptable. Además, como software de análisis y procesamiento de estos vídeos se ha utilizado Matlab [20]. Este software ha sido elegido debido a su gran potencia en lo que se refiere a procesamiento multimedia y porque su lenguaje de programación nos ha permitido realizar el posterior procesamiento de estos datos obtenidos. La versión usada de Matlab ha sido la 7.7 (R2008b) ya que es la primera versión que posee las rutinas para decodificar nuestra base de datos de vídeos (Codec XviD).

4.2 Arquitectura del sistema de tracking

El diagrama de bloques del sistema es el siguiente:

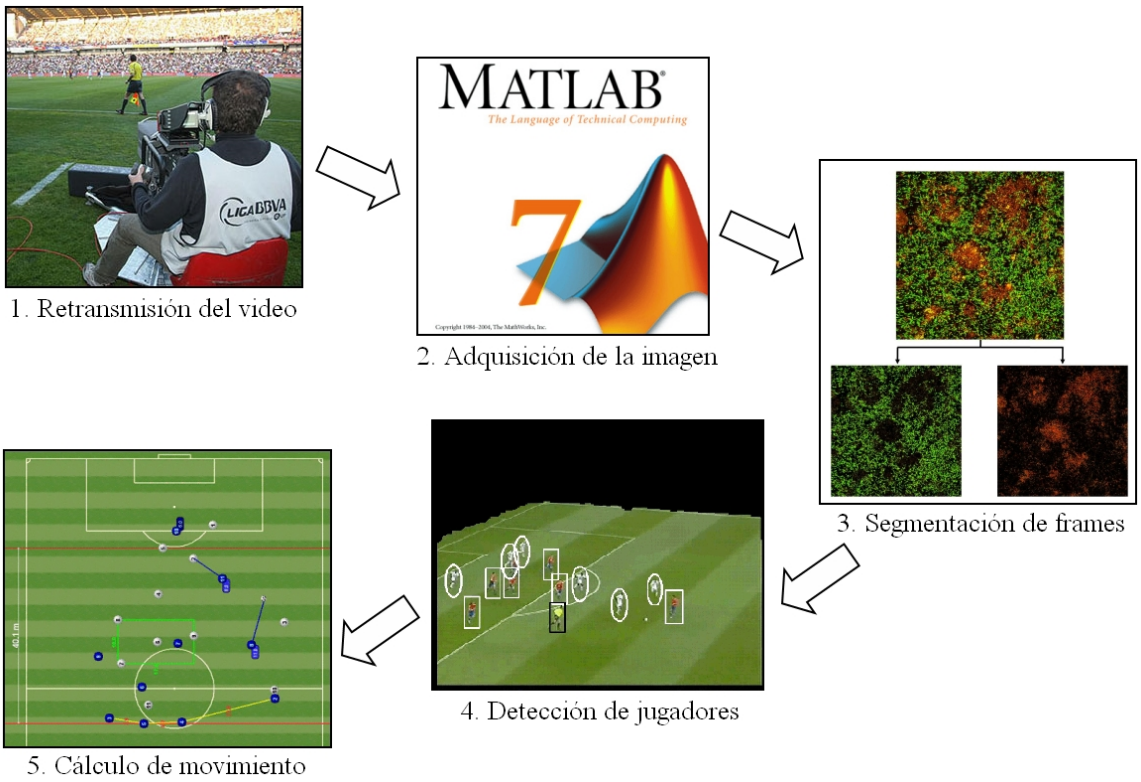


Imagen 11 - Diagrama de bloques[11]

Las funcionalidades de cada bloque son las siguientes:

1. **Retransmisión del vídeo**

   Captura de retransmisión por parte de las cámaras de TV. En nuestro caso este bloque estará compuesto por la base de datos de partidos obtenida de la web.

2. **Adquisición de la imagen**

   Mediante el software utilizado, captura de los distintos frames que componen un vídeo y decodificación para su posterior procesamiento digital. En este apartado haremos uso del sistema de codificación XviD.

3. **Segmentación de frames**

   En este bloque se hará una segmentación de cada frame para intentar dividirlo en dos partes diferenciadas: la parte dominante de la imagen (el césped) y el resto (jugadores, árbitro y público).

[11] Imágenes obtenidas de images.google.es

4. **Detección de jugadores**

   En este bloque se mejora la calidad de la segmentación para eliminar posibles fallos y se realiza la detección de los propios jugadores mediante un mapeo de la imagen.

5. **Cálculo de movimiento**

   Para calcular el movimiento haremos uso de la detección anterior y una posible estimación del movimiento de la cámara de retransmisión

Las imágenes usadas en el diagrama de bloques no se corresponden con la salida de nuestro algoritmo, simplemente se ha querido dotar al lector de una idea gráfica genérica del funcionamiento del algoritmo. En los siguientes apartados se mostrará el funcionamiento de una forma más extensa y se podrán ver las salidas reales de nuestro algoritmo.

4.3 Retransmisión del vídeo

En este primer apartado nos encontramos con el primer problema: el tipo de retransmisión que usamos para el seguimiento.

Como comentamos en la introducción, el único sistema conocido hasta el momento de tracking es el usado por las empresas citadas: por medio de una serie de cámaras **fijas** (desde 8 hasta 16) podemos obtener un seguimiento con un nivel de error ínfimo de todos y cada uno de los jugadores. Estas cámaras están fijas y a una altura en la que podemos obtener una vista total del campo lo que facilita la obtención del seguimiento de los jugadores.

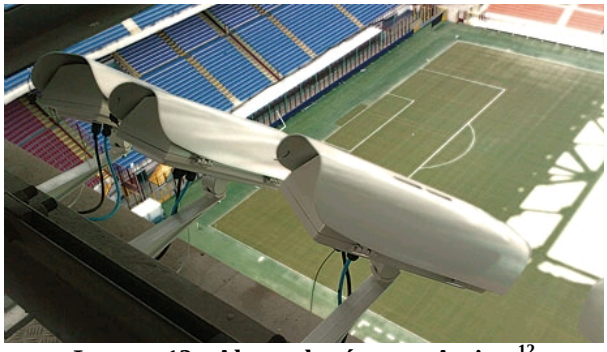


Imagen 12 - Altura de cámaras Amisco[12]

Debido al tipo de imagen que nosotros vamos a utilizar (la propia retransmisión de televisión) nos enfrentamos a dos problemas que no tienen en los anteriores sistemas:

- **Visión parcial del campo**, ya que las cámaras de retransmisión se centran en la zona de juego (normalmente donde se encuentra el balón).

- **Movimiento de la cámara,** ya que la cámara puede estar moviéndose, lo que nos daría un movimiento erróneo de los jugadores.

En nuestro caso, debido a las limitaciones temporales y económicas del proyecto no nos podemos centrar en el primer problema y, como se explicó en los objetivos, se comentará como posibles estudios futuros o evoluciones de nuestro sistema.

Sin embargo, para el segundo error sí que intentaremos reducir la afectación que produce en nuestro sistema de seguimiento mediante una estimación del movimiento de la cámara, aunque esto se verá más adelante en los siguientes apartados.

[12] Imagen obtenida de sport-universal.com

4.4 Adquisición de la imagen

Una vez que obtenemos el vídeo del partido, es necesario procesarlo para poder obtener los distintos frames y tratar las imágenes resultantes.

Para conseguir esto, como se ha mencionado en la introducción, vamos a utilizar el software matemático Matlab. Ha sido necesario obtener la versión R2008b de este software debido a las codificaciones actuales de vídeo utilizadas.

Como ya se ha mencionado, para crear una base de datos nos hemos descargado una serie de partidos de la página Web http://www.rojadirecta.org. Los partidos descargados son los siguientes:

- Liverpool - Real Madrid (10 de Marzo de 2009)
- España – Rusia (10 de Junio de 2008)
- Atlético de Madrid – Villareal (15 de Marzo de 2009)
- Fulham – Manchester United (7 de Marzo de 2009)
- Almería – Valencia (14 de Septiembre de 2008)
- Barcelona – Lyon (10 de Marzo de 2009)
- Juventus– Bologna (14 de Marzo de 2009)

Como podemos comprobar, la base de datos de partidos es extensa ya que cada uno de los partidos tiene 90 minutos de movimientos. Además, hay distintas tonalidades de césped de los distintos campos y partidos de diferentes ligas.

Si nos centramos en el procedimiento seguido, al intentar obtener un objeto lector de cada partido nos encontramos con el problema de la codificación. Al usar una rutina específica de lectura de vídeos en formato AVI [21](Audio Vídeo Interleave) obteníamos el siguiente error:

Unable to locate decompressor to decompress vídeo stream

Estos partidos están codificados mediante el sistema de compresión de vídeo XviD. Este software es de código libre por lo que no tuvimos problemas en instalarlo y que funcionara con Matlab en la adquisición de la imagen.

Cualquiera de los partidos tiene una duración de 95-100 minutos que equivalen a 5700-6000 segundos. Si usamos la codificación europea de televisión (PAL [22] - *Phase Alternating Line*) tenemos una frecuencia de fotograma de 25Hz, lo que obtiene un periodo de 0.04 segundos por frame.

$$f = \frac{1}{T} \Rightarrow T = \frac{1}{f} = \frac{1}{25} = 0.04 sg / frame$$

Ecuación 1 - Frecuencia de fotograma

Esto se traduce en 25 frames/segundo, por lo que un partido normal estará compuesto por 142500-150000 frames. Este número tan alto de frames es un problema para nosotros, ya que el software utilizado no permite realizar objetos de lectura de más de

100 frames por problemas de memoria. A causa de esto, deberemos realizar un sistema cíclico que obtenga los objetos lectores de 100 en 100 frames.

Para poder seguir con el resto de la descripción del algoritmo, es necesario que se explique como trata Matlab las imágenes. Una imagen es representada mediante una matriz. Es fácil pensar en una imagen como una matriz, ya que las televisiones actuales están compuestas de píxeles y cada uno de estos píxeles podría estar compuesto por un valor concreto. Si nos centramos en una imagen en blanco y negro, esta suposición sería acertada. Una imagen en blanco y negro podría describirse como una matriz de M filas por N columnas en el que cada valor correspondería a la codificación del color en un byte (8 bits), es decir, un valor entre 0 y 255.

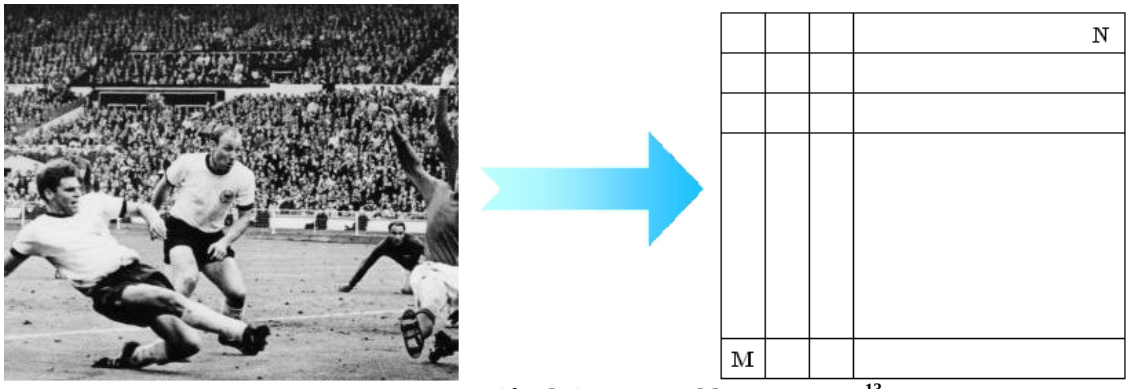


Imagen 13 - Conversión de imagen en blanco y negro[13]

Pero este no es el caso que nos ocupa, debemos procesar imágenes en color por lo que una representación de una matriz en 2 dimensiones no es válida. Para convertir las imágenes en color en datos, Matlab utiliza las matrices en tres dimensiones y se apoya en la representación RGB (Red, Green and Blue) de cada píxel. Esta representación se basa en la síntesis aditiva, en la que un color puede representarse como la suma de los tres colores de luz primarios (rojo, verde y azul). Para poderlo representar con valores concretos, al igual que en el caso de la escala de grises, cada color va a tener un valor entre 0 y 255 que va a significar la intervención de cada color en la mezcla, siendo 0 intervención nula y 255 intervención máxima. Para entenderlo de forma más clara podemos pensar simplemente en el color amarillo: este color está formado por la suma del verde y del rojo por lo que su representación RGB sería (255, 255, 0).

Una vez introducida la representación RGB (volveremos a ella en la segmentación de la imagen) ya nos podemos hacer una idea de cómo Matlab representa las imágenes en color.

[13] Imagen obtenida de images.google.es

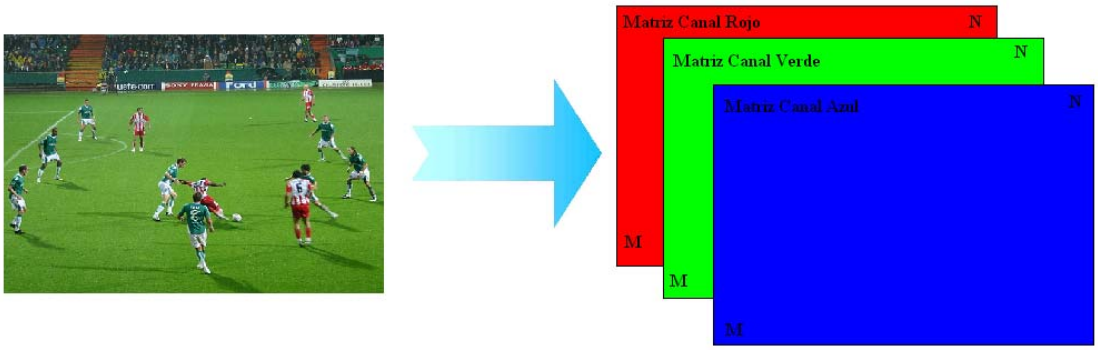


Imagen 14 - Conversión de imagen en color[14]

Cada uno de estos canales de color va a ser una matriz, siendo la imagen la combinación de los tres canales y con un tamaño de:

$$Tamaño\ de\ la\ imagen = M \cdot N \cdot 3\, píxeles$$

Ecuación 2 - Tamaño imagen en Matlab

Si observamos con mayor detenimiento el tipo de conversión, podemos empezar a pensar en la magnitud de los datos que vamos a procesar: un frame aleatorio de uno de los partidos está compuesto por M = 480 filas y N = 640 columnas. Esto resulta en 307200 píxeles por canal. Al tener tres canales (rojo, verde y azul), cada frame está compuesto por 921600 píxeles. Esta cantidad de datos se va a obtener cada 0.04 sg, es decir, cada 40 msg lo que supone una capacidad de procesamiento muy alta. Como veremos más adelante, este tiempo no será el utilizado ya que para el cálculo del seguimiento de los jugadores se podrá utilizar un intervalo mayor que será de 5 frames/segundo, o lo que es lo mismo, un frame cada 0.2 segundos.

Mediante este cálculo, se ha podido comprobar en que orden de magnitud de datos vamos a trabajar y, de esta forma, servir de introducción para el siguiente capítulo. En este siguiente capítulo se va a desarrollar la segmentación del campo en dos partes: el césped y el resto de la imagen. Como primer paso, se piensa en la segmentación por distancia umbral píxel a píxel a un valor medio del césped. Como se comprobará más adelante, la segmentación píxel a píxel es inviable ya que el tiempo de respuesta del sistema es mucho mayor al esperado y hace que este tipo de segmentación se descarte.

[14] Imagen obtenida de images.google.es

4.5 Segmentación de frames

Como primer paso, vamos a explicar que se entiende por "segmentación" y que se quiere conseguir en este apartado que nos ayude en el proceso de seguimiento de los jugadores.

4.5.1 ¿Qué es la segmentación?

La definición exacta de segmentación según *wordreference* [23] es "*la* división de algo en segmentos*"*. Para el caso que nos ocupa, esta definición es un poco incompleta e inexacta. Nuestro objetivo principal en el proyecto es conseguir realizar un seguimiento de los jugadores sobre el campo. Para ello, nuestro primer paso será intentar poder diferenciar que parte de la imagen se corresponde con los jugadores y que parte no. Realmente es algo sencillo de diferenciar desde el punto de vista de un observador humano, pero adquiere mucha más dificultad cuando el procesamiento se realiza de forma informática y automatizada. Por tanto, podemos redefinir concretamente para este caso el término "segmentación" como la *división de cada uno de los frames en que se divide el vídeo objetivo en dos bloques bien diferenciados: por un lado el césped y por el otro los jugadores y el resto de la escena.*

4.5.2 ¿Cómo vamos a realizar esta segmentación?

A tenor del estudio previo realizado sobre el tema y comparando los métodos usados en la actualidad, observamos que **el color** es la característica más utilizada en la mayoría de los algoritmos.

Los motivos principales que nos hacen elegir el color para realizar la segmentación son los siguientes:

- Un vídeo no es más que la superposición de imágenes a una frecuencia determinada. Como se ha explicado en el punto anterior, el software utilizado representa las imágenes capturadas como la unión de 3 matrices de color siguiendo la representación RGB. Por lo tanto, después de la adquisición de la imagen, los datos con los que vamos a poder trabajar directamente son una representación numérica de los colores. Podremos filtrar entonces la imagen mediante el procesamiento de estos números.

- Si observamos las imágenes típicas de un partido de fútbol, podemos comprender que el color dominante en todas las escenas es el verde del césped. Este verde puede tomar distintas tonalidades en función del campo en el que se juegue, la luminosidad o el estado en el que se encuentre. Podemos usar este color dominante para realizar la segmentación deseada de forma que podamos filtrar, dentro de la imagen, las partes que podrían ser césped.

- Dentro de la imagen, el color no varía de forma significativa con cambios de orientación, de vista o de forma. Podríamos decir que el color tiene un comportamiento "homogéneo" para nuestras necesidades por lo que su uso sería óptimo.

Para ilustrar el segundo motivo, vamos a mostrar diferentes tipos de césped. Como podemos observar, hay cambio de tonalidad incluso en la misma imagen debido al diferente tipo de corte del césped que sirve de ayuda a los jueces de línea.

Imagen 15 - Tipos de césped[15]

Se puede comprobar en estas capturas la diferencia de color entre los distintos partidos tanto a nivel de tono como a nivel de densidad de imagen.

4.5.3 El algoritmo de segmentación

Una vez explicado la característica principal que vamos a usar para realizar la segmentación, vamos a desarrollar el algoritmo que lo realiza. Como primer paso, debemos conocer el tamaño de la imagen. En la mayoría de los casos utilizados por nosotros vamos a tener imágenes formadas por 480 filas (M=480) y 640 columnas (N=640). Esto será importante ya que deberemos conocer la matriz que vamos a recorrer aparte de influir en el coste temporal de procesamiento.

Como primera idea, se piensa en utilizar las distancias de los componentes RGB entre cada píxel para poder realizar la segmentación. Según los estudios investigados, una de las distancias entre píxeles más usadas para realizar una segmentación por color es la distancia de Mahalanobis [24]. Esta distancia obtiene la similitud entre dos variables multidimensionales y se diferencia de la distancia euclídea en que tiene en cuenta la correlación entre ambas variables. La ecuación que representa dicha distancia es la siguiente, siendo S la matriz de covarianza:

$$d_m(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}$$

Ecuación 3 - Distancia de Mahalanobis

[15] Imágenes capturadas de los videos utilizados para el proyecto.

Si comparamos esta fórmula con la de la distancia euclídea no podemos ver, de forma gráfica, que diferencia puede haber en el cálculo de la segmentación.

$$d_e(\vec{x},\vec{y}) = \sqrt{\|\vec{x}-\vec{y}\|^2}$$

Ecuación 4 - Distancia Euclídea

Si embargo, en ambos casos podemos ver como existen dos puntos $\vec{x}$ e $\vec{y}$ que serían las componentes RGB de los dos puntos entre los que se quiere calcular la distancia. Como se comentó en el punto de adquisición de la imagen, estas componentes RGB estarían formadas por un vector de 3 componentes por lo que el utilizar estas ecuaciones con estas componentes sería relativamente sencillo.

Para comprobarlo de forma visual, vamos a utilizar un ejemplo realizado por el departamento de electrónica de la Universidad Técnica Federico de Santa María [25] en el que utilizan ambas distancias y así poder ver claramente la diferencia.

Imagen 16 - Diferencia uso distancias[16]

Como podemos comprobar, con la distancia de Mahalanobis la gama de colores se amplía de forma que la segmentación la realiza sobre las distintas tonalidades de un color mientras que para la distancia Euclídea se centra en un color y la variación permitida es menor. Para el caso que nos ocupa, es mucho más interesante el uso de la primera (Mahalanobis) ya que, como hemos podido comprobar en la imagen anterior, nuestro espectro de verdes será muy amplio. Sin embargo, como veremos más adelante, la distancia utilizada será la distancia Euclídea.

Otro de los aspectos a tener en cuenta para el cálculo de la distancia es el valor de referencia. En la ecuación de la distancia de Euclídea, esta variable está representada por $\vec{y}$ y es el valor del color que vamos a usar como referencia en el cálculo de la distancia. Por ejemplo, para el partido Liverpool-Madrid, este valor está inicializado a [101,134,23], para el partido España-Rusia este valor es [90,139,20] y para el Barcelona-Lyon es [84, 119, 73]. Estos colores, como podemos observar, aún siendo verdes son bastante diferentes.

Imagen 17 - Verdes iniciales

[16] Imagen obtenida del estudio [21]

Si comparamos estos colores con los de las imágenes de los tipos de césped, podemos ver que este valor inicial se aproxima al tono principal de cada imagen. Este valor de referencia influye bastante en el resultado final ya que una mala elección podría llevarnos a una segmentación errónea como veremos en los resultados de la segmentación.

En este proyecto se ha decidido que a la hora de ejecutar el algoritmo se conozca este valor, que es representado por un valor aleatorio del césped de ese mismo partido. En teoría, ya que nuestro sistema debería ser totalmente automático se podría discutir la introducción manual de este parámetro pero, ya que nos hemos centrado en el seguimiento posterior de los jugadores, se ha optado por la solución más sencilla. Sin embargo, propondremos una solución teórica a este problema en el apartado de líneas futuras. Debido a esto, es mucho más sencillo la introducción de un solo valor de referencia en lugar de introducir varios (necesarios para calcular la covarianza de la distancia de Mahalanobis) por lo que se decide usar la distancia Euclídea. Además, como el algoritmo se basará en la detección por umbrales, el espectro de verdes posibles también se verá ampliado (que era la principal ventaja de la distancia Mahalanobis) y no estará restringido al verde de referencia. También comentar que el cálculo de la distancia Euclídea es menos costoso computacionalmente que la distancia de Mahalanobis.

Por lo tanto, la idea principal va a ser obtener la distancia entre todos y cada uno de los píxeles del frame y el valor de referencia elegido por nosotros para el partido en concreto. Después vamos a comparar esta distancia con una distancia umbral para decidir si el píxel del frame es "similar" al de referencia. A partir de esta regla, vamos a crear una imagen alternativa del frame en blanco y negro donde el negro corresponderá al césped y el blanco al resto. Esta imagen alternativa nos será de mucha utilidad posteriormente para la detección de bloques mediante rutinas de Matlab.

El diagrama de bloques inicial de este algoritmo sería el siguiente:

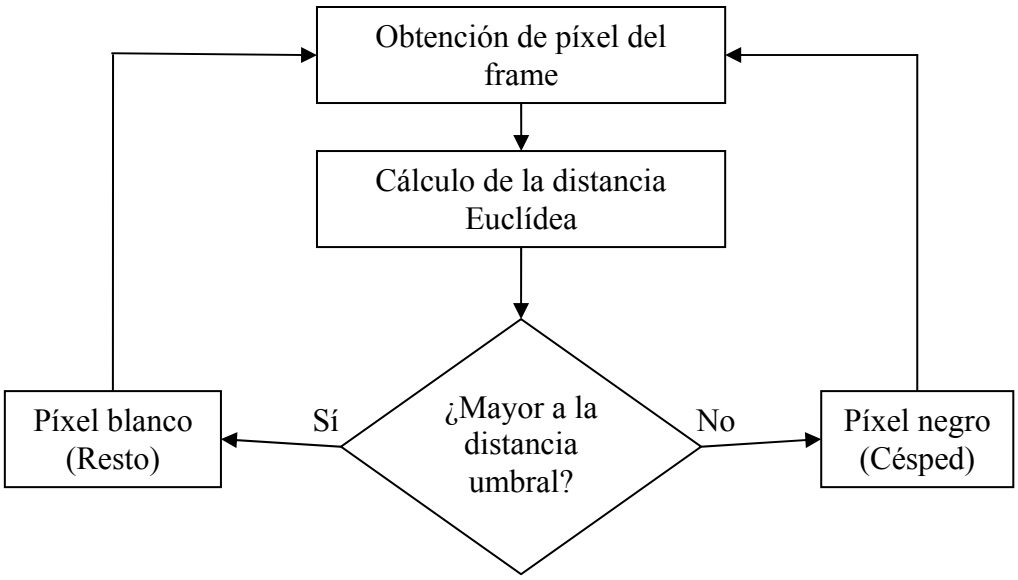


Imagen 18 - Diagrama de bloques segmentación inicial

4.5.4 Problemas encontrados y decisiones tomadas

4.5.4.1 Umbrales de decisión

El primer problema que vamos a encontrar usando el modelo de las distancias va a ser decidir que umbral elegimos para una segmentación correcta. Para hacernos una idea de orden de magnitud de este umbral probamos con el valor utilizado por la Universidad Técnica Federico de Santa María en su estudio, un valor de 0.12. El uso de este valor en nuestro algoritmo no es aceptable ya que los resultados nos son válidos aunque en principio el procedimiento de cálculo de la distancia es similar y solo cambia el color usado, cosa que no debería influir. Por ello, realizamos una serie de pruebas con distintos umbrales para conocer hasta que punto este umbral podría afectar el filtrado.

En la siguiente imagen vamos a mostrar una prueba lanzada sobre una imagen de un partido con distintos umbrales de decisión. Los píxeles negros son los que tienen una distancia menor al umbral mientras que los blancos tienen una distancia mayor:

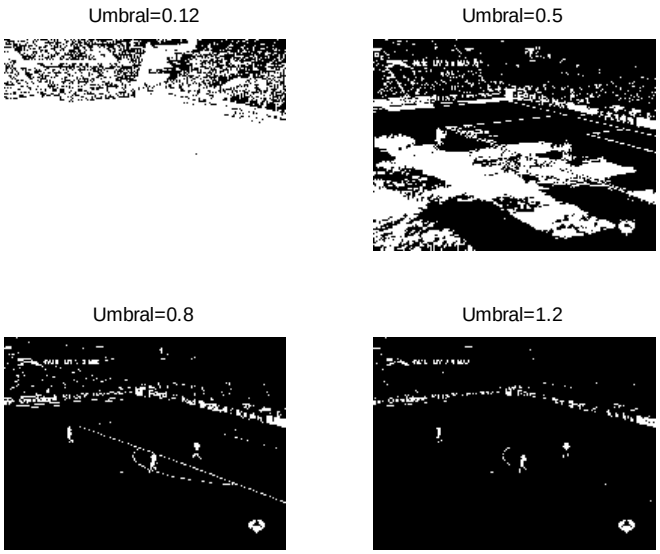


Imagen 19 - Prueba umbrales RGB

En esta imagen podemos comprobar como el valor más aceptable de los probados oscila entre 0.8 y 1.2. Como se puede comprobar a simple vista, este resultado no es viable para nuestros intereses ya que, aparte de encontrar los jugadores en el campo, no hace distinción entre estos y las propias líneas blancas del campo. Para hacer frente a este problema vamos a hacer uso de la representación HSV [26] (Hue-Saturation-Value) de la imagen. Este modelo de color se basa en tres componentes principales:

- Hue (Tonalidad) que nos define el tipo de color empleado en el píxel. Estos valores están comprendidos entre el 0 y el 360 ya que es una representación de tipo angular. Como ejemplo el 120 sería el verde.
- Saturation (Saturación) que se define como la distancia al eje de brillo negro-blanco. De este componente podríamos obtener la decoloración de cada píxel. Los valores que adquiere van del 0 al 100%.

- Value (Valor del color) que es el brillo del color. Los valores normales van del 0 al 100% siendo 0 el valor negro.

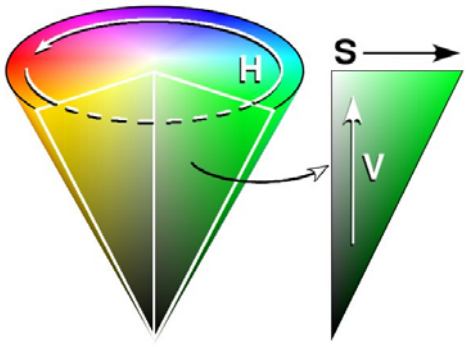


Imagen 20 - Modelo de color HSV[17]

La representación HSV será totalmente complementaria a la representación RGB, de forma que una vez filtrado por RGB volveremos a filtrar por HSV siendo el sistema mucho más robusto que usando solo la primera representación. La forma en la que vamos a realizar este filtrado es distinta a la usada con la representación RGB ya que, en este caso, solo vamos a usar la componente de tonalidad (Hue). Al igual que con el modelo de color RGB, será necesario un umbral de decisión sobre esta representación HSV. Este umbral representará un porcentaje de la desviación que puede cometer la tonalidad de cada píxel siguiendo los siguientes límites:

$$\begin{cases} (1 - umbral) \cdot HUE_Y \geq HUE_X \\ (1 + umbral) \cdot HUE_Y \leq HUE_X \end{cases}$$

Ecuación 5 - Límites de filtrado en representación HUE

Es decir, si la tonalidad del píxel a tratar se encuentra por debajo del valor inferior de referencia o se encuentra por encima del valor superior de referencia será considerado como un píxel que no representa al césped ya que no está dentro de los valores válidos. Hay que darse cuenta de que el píxel no puede cumplir ambas condiciones ya que sería imposible, con cumplir cualquiera de las dos bastaría.

Por lo tanto y a modo de resumen, para que un píxel sea identificado como parte de un jugador deberá obtener una distancia de componentes RGB entre el píxel tratado y el de referencia mayor a la distancia umbral y después su tonalidad (hue) deberá encontrarse fuera de los límites de filtrado de la ecuación anterior (en alguno de los dos casos). Estas condiciones se podrán ver con mayor claridad en el esquema de bloques final de la segmentación.

Para poder comprobar como afecta este cambio en la segmentación a los frames, vamos a realizar la misma prueba que anteriormente pero modificando el umbral de tonalidad en distintos valores. Como podemos observar en la prueba lanzada sobre la imagen (umbral RGB de 0.8), la calidad de la segmentación aumenta al usar este tipo de decisión:

[17] Imagen obtenida de Wikipedia (http://es.wikipedia.org/wiki/Archivo:HSV_cone.jpg)

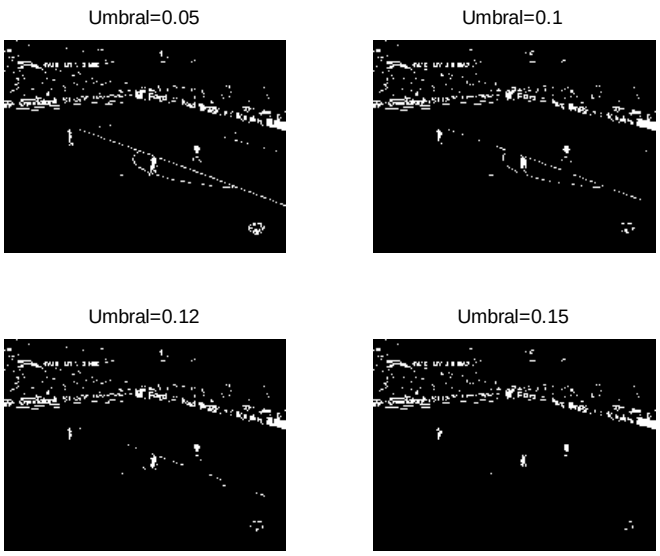


Imagen 21 - Prueba umbrales HSV (Con RGB=0.8)

Si observamos con detenimiento la primera imagen (umbral HSV=0.05) podemos comprobar que apenas varía con respecto a la imagen en la que no realizábamos el filtrado por HSV. Sin embargo, para un valor cercano a 0.15 podemos ver que las líneas de campo desaparecen, de forma que este valor es el que resulta válido para nuestro algoritmo.

Este tipo de representación también nos valdrá para poder diferenciar las sombras de los jugadores sobre el campo en partidos muy soleados o con mucha iluminación, ya que aunque la tonalidad es verde, el color es más oscuro.

4.5.4.2 Segmentación píxel y segmentación bloque

El segundo problema al que nos enfrentamos al realizar la segmentación es el tiempo de cómputo. Como hemos mencionado antes, cada frame obtenido está compuesto de aproximadamente 921.600 píxeles. El algoritmo de segmentación recorre los píxeles uno a uno y calcula una distancia por lo que este proceso es muy costoso tanto a nivel temporal como a nivel computacional. A modo de ejemplo, podríamos obtener un tiempo de procesamiento de 45 segundos/frame.

Debido a la necesidad de realizar un algoritmo más eficiente, se desarrolla otro tipo de segmentación basada en bloques de píxeles. De esta forma, el número de operaciones disminuye proporcionalmente al número de píxeles que forme el bloque. Para nuestro caso, vamos a formar bloques de 4 píxeles en cuadrícula lo que hará que realicemos un cuarto de las operaciones que se ejecutaban anteriormente. Además, vamos a calcular la distancia simple entre bloques (menos costosa que la Mahalanobis) por lo que el tiempo de cómputo se debería reducir aún más.

Al realizar esta modificación en la forma de segmentar, el único inconveniente que vamos a obtener es la calidad del segmentado. Por calidad entendemos que un procesamiento que se realizaba píxel a píxel siempre obtendrá mejores resultados que

un procesamiento bloque a bloque ya que el error mínimo que podemos cometer en la primera opción es un píxel mientras que en la segunda sería un bloque. Para nuestro caso, un error en un bloque estaría afectando a 4 píxeles y, por lo tanto, mayor cantidad de imagen.

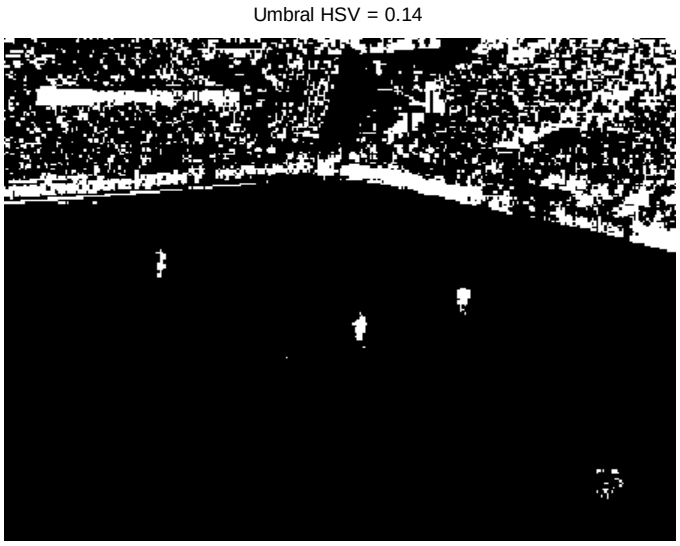


Imagen 22 - Segmentación bloque (Umbral RGB=0.8)

En esta imagen podemos comprobar como la segmentación de los jugadores sigue siendo válida a pesar de haber reducido la calidad del algoritmo. También podemos comprobar como esta modificación también afecta al público de forma que es menos permisivo a la hora de detectarlo llegando incluso a funcionar mejor que en la segmentación píxel. Esto, como veremos después, no nos afectará ya que al pasar por el filtrado eliminaremos la influencia en la imagen del público.

Otra de las formas de ver las ventajas de esta segmentación son los tiempos de respuesta. Para este ejemplo, el tiempo de procesamiento de cada frame se reduce a 12 segundos/frame. Este es un valor muy cercano al valor que esperábamos ya que, usando la proporcionalidad antes citada:

$$45sg\ /\ frame \cdot \frac{1}{4} = 11.25sg\ /\ frame \approx 12sg\ /\ frame$$

Este pequeña diferencia puede deberse al cálculo manual de los tiempos de procesamiento.

Por lo tanto, el nuevo diagrama de bloques de este algoritmo sería:

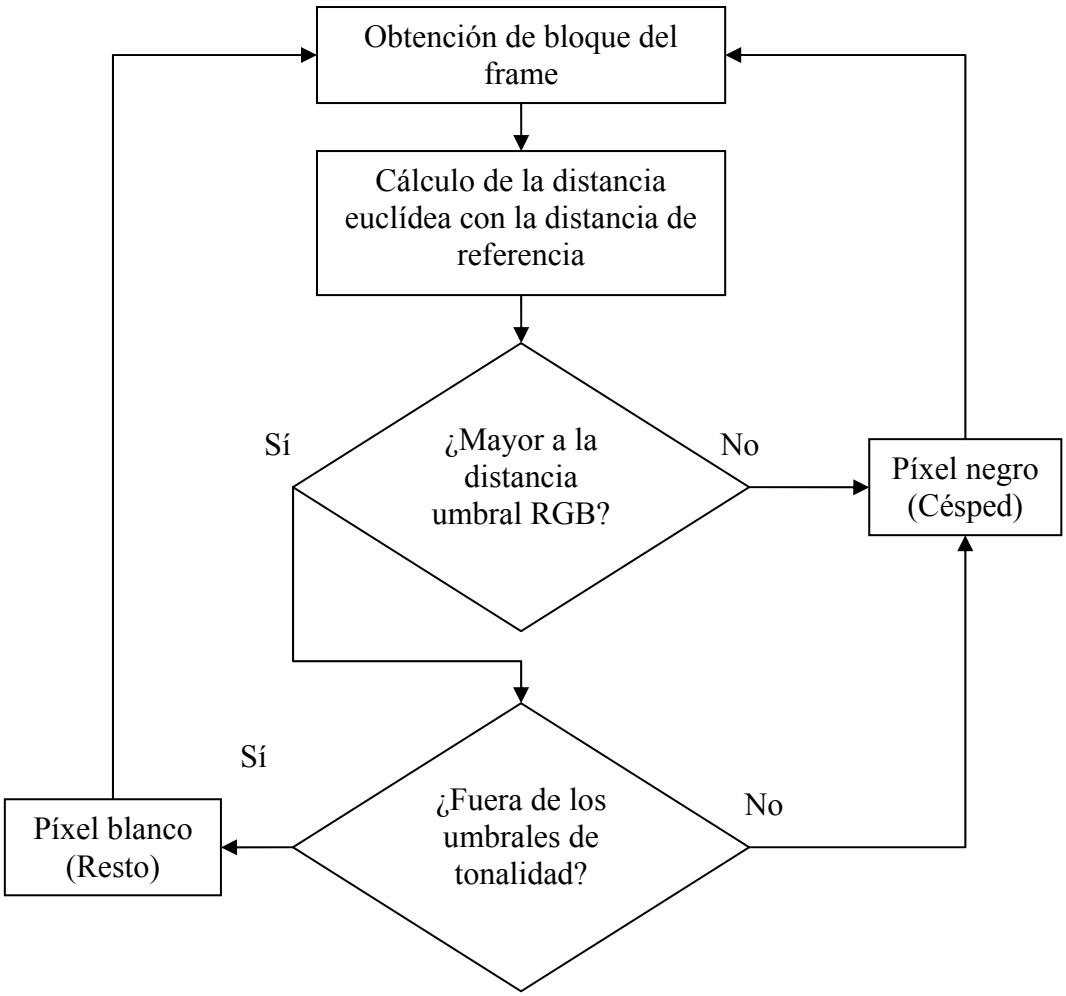


Imagen 23 - Diagrama de bloques segmentación

4.5.4.3 Mala elección del verde de referencia

Como se citó anteriormente, este es otro de los pasos críticos del bloque del cálculo de la segmentación. El utilizar un verde u otro como modelo de referencia para calcular las distancias aparentemente no debería de influir de manera excesiva. Sin embargo, esta elección es básica y tiene una influencia muy grande en la imagen segmentada posterior. Para poder ver mejor este problema, vamos a realizar la segmentación sobre una misma imagen con varios verdes de referencia en los que, a simple vista, podremos observar la gran diferencia entre ellos.

Para el ejemplo vamos a seleccionar la misma imagen que se ha usado anteriormente y vamos a utilizar dos tipos de verdes:

- Verde habitual de referencia para este partido, con representación RGB=[90,139,20].
- Verde puro, con representación RGB=[1,255,1].

Los resultados son los siguientes:

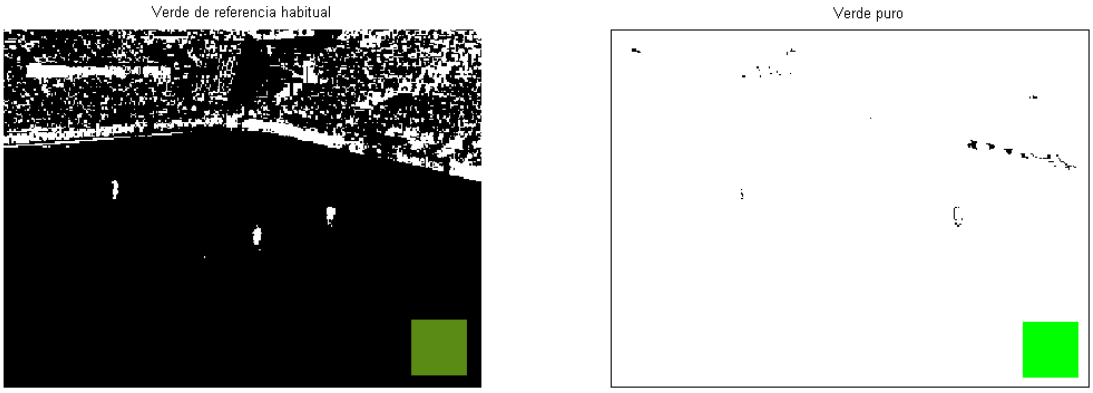


Imagen 24 – Elección de verde puro

Este ejemplo es demasiado drástico, ya que en ningún caso vamos a tener el verde puro como verde de referencia. Si utilizamos otro verde de otro tipo de césped podemos encontrar diferencias pero no tan destacables. Vamos a ver otro ejemplo en el que realizamos la segmentación con otro tipo de verde:

- Verde de otro partido, con representación RGB=[84,119,73]

Como podemos observar en la siguiente imagen, estas diferencias son mínimas pero realmente nos afectan en gran medida como podremos observar en los pasos siguientes del algoritmo.

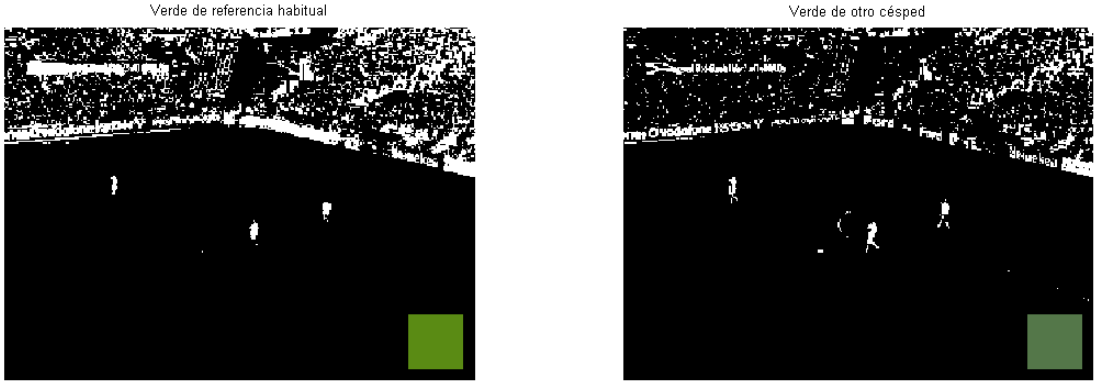


Imagen 25 - Elección de verde de referencia no ideal

Podemos ver en la segunda imagen como al utilizar otro tipo de verde que no es el ideal hay líneas del campo que no elimina. El objetivo principal de esta segmentación es poder obtener los lugares del campo en donde se encuentran los jugadores por lo que cualquier tipo de punto erróneo nos podría llevar a error ya que podría ser identificado (como se verá más adelante) como un jugador.

Por lo tanto, la utilización de un verde de referencia adecuado es crítica para el sistema, ya que una desviación muy pequeña en la representación RGB de este verde podría tener como consecuencia una detección errónea de líneas de campo, lo que nos haría detectar más jugadores de los presentes en la escena. En las dos imágenes anteriores

hemos podido comprobar como afecta el utilizar otro tipo de verde que no es el óptimo. Esto hace que la segmentación dependa demasiado de la elección manual de este color que se hace previa al seguimiento. Por lo tanto, como hemos mencionado anteriormente, un algoritmo que no usara este verde de referencia para la segmentación y que su cálculo fuera totalmente automatizado sería lo óptimo. Como veremos en las líneas futuras, este algoritmo está pensado a grandes rasgos aunque no se desarrolla en este proyecto.

4.6 Detección de jugadores

El siguiente apartado consiste en la detección de los jugadores en la imagen tras haber pasado por el periodo de segmentación. Esta detección será el paso previo al cálculo de movimiento de los jugadores.

4.6.1 ¿Qué entendemos por detección?

Por *detección* entendemos el proceso en el cual obtenemos la posición de cada uno de los jugadores en la imagen. La posición a la que nos referimos es la posición en coordenadas dentro de la propia imagen, no la posición dentro del campo de fútbol. Esta diferenciación es de gran importancia en nuestro proyecto ya que, como comentamos en la motivación, para poder identificar la posición de cada jugador en el propio campo sería necesario otro tipo de estudio y otro tipo de imágenes.

Por lo tanto, la principal tarea que realizaremos en este apartado será el identificar a los jugadores que aparecen en cada frame y asignarles una coordenada de posición. Esta coordenada de posición nos servirá después para poder obtener el movimiento de cada jugador.

4.6.2 ¿Cómo vamos a realizar esta detección?

Para realizar esta detección de los jugadores vamos a utilizar las imágenes obtenidas en el anterior proceso de segmentación. Estas imágenes teóricamente están compuestas por dos colores:

- Negro, en este color tendremos el césped de la imagen.
- Blanco, que será el color de los jugadores.

Si esto funcionara de forma ideal, podríamos obtener imágenes de este tipo:

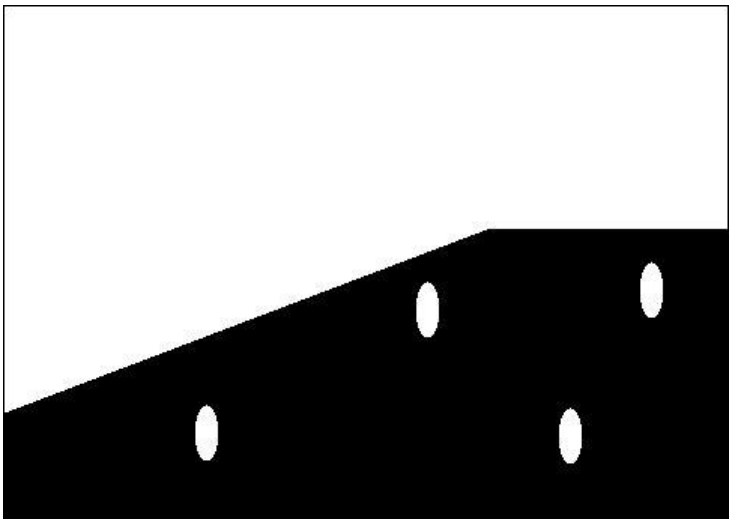


Imagen 26 - Ejemplo ideal

Como hemos podido ver en los ejemplos del algoritmo de segmentación, esta división no es perfecta ya que dentro de los colores del público puede haber una gran variedad por lo que el decisor por distancia puede identificarlo como blanco o como negro. Como veremos en los siguientes apartados, este no será el único problema con el que nos encontremos por lo que tendremos que realizar una serie de acciones previas antes de poder tener la imagen relativamente "ideal" similar a la ofrecida en la imagen.

Una vez obtenida esta imagen "ideal" y mediante el uso de varias rutinas de Matlab de etiquetado de imágenes podremos asignar a cada jugador una coordenada y así poder detectar todos los jugadores disponibles en la imagen. El esquema global sería el siguiente:

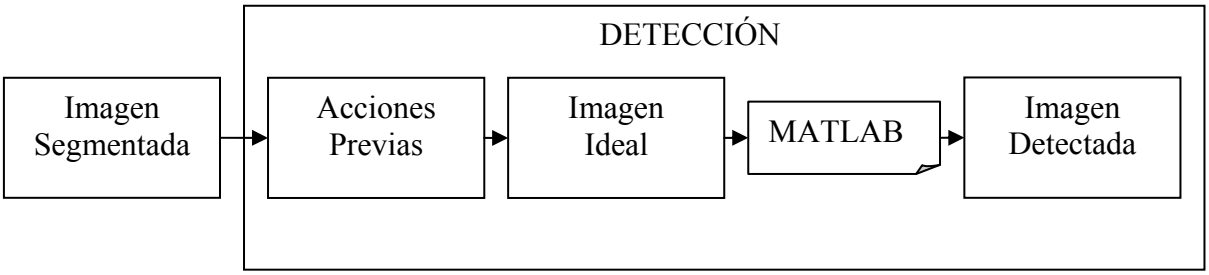


Imagen 27 - Esquema básico de detección

4.6.3 El algoritmo de detección de jugadores

Para poder detallar el algoritmo de detección que hemos desarrollado vamos a apoyarnos en el esquema básico de detección mostrado en el punto anterior. El paso denominado "Acciones Previas" estará compuesto por varias funciones que harán que nuestra imagen segmentada y con errores pase a ser lo más ideal posible y respetando al máximo el parecido con la realidad.

4.6.3.1 Acciones previas

Para poder entender las diferentes acciones previas que vamos a realizar antes de procesar las imágenes con las rutinas de Matlab vamos a realizar una pequeña introducción de cómo funcionan estas rutinas.

Para dar una primera idea del funcionamiento, debemos saber que se van a basar en una idea básica: la identificación de un jugador consiste en la determinación de "manchas" blancas rodeadas de fondo negro. Por manchas entendemos cualquier tipo de dibujo (sin importar la forma) que este unido mediante píxeles de forma conjunta. El algoritmo identifica el número de manchas que existen en el frame y les asigna un número de identificación.

Imagen 28 - Ejemplo simple rutina

En estas imágenes podemos ver unas manchas de formas distintas sobre un fondo negro. El resultado de la rutina nos diría que en el primer ejemplo hay 3 manchas identificadas (lo que nosotros identificaríamos como jugadores) mientras que en el segundo ejemplo también identificaría 3 manchas ya que una de ellas no estaría unida de forma conjunta y sería identificada como dos. Este tipo de identificación va a ser la causa principal de que necesitemos realizar estas acciones previas.

Eliminación del logo de la cadena

Este función es básica ya que el logo de la cadena no suele identificarse como césped por lo que su detección nos llevaría a pensar que en esa zona habría uno o varios jugadores. Este paso depende de donde se sitúe el logo de la cadena teniendo dos opciones:

- Logo situado en la parte superior de la imagen, que estaría situado por delante del público o de parte del estadio y no nos afectaría para la identificación de los jugadores.
- Logo situado en la parte inferior de la imagen, que estaría situado por delante del césped o de los propios jugadores y nos conduciría a errores.

La decisión que se ha tomado para paliar este problema ha sido la colocación de un filtro “paso nada” sobre los píxeles sobre los que se encuentra el logo. Es decir, tras la segmentación, vamos a colocar un rectángulo de las dimensiones adecuadas que se superponga al logo. Este rectángulo es creado de color negro por lo que realmente simula que en lugar del logo habría césped. El resultado sería el siguiente:

Imagen 29 - Eliminación del logo

Como podemos observar en la primera imagen, en la esquina inferior derecha observamos como el algoritmo de segmentación no discrimina el logo mientras que en la segunda se ha superpuesto un rectángulo similar al logo para que este no influya en la detección. Esta decisión es la más sencilla y a la vez la más eficaz pero tiene dos inconvenientes importantes:

1. Cada cadena tiene un logo distinto y en distinta posición.

   Para el ejemplo mostrado, el logo se encuentra en la esquina inferior derecha. Para eliminarlo, hemos usado un rectángulo de coordenadas (x_1, x_2)(y_1, y_2)=(412,443)(557,594). Esto forma un rectángulo de 32x38 píxeles. Pero podría darse el caso de que el logo estuviera en otra coordenada ó que fuera de otro tamaño. Para nuestro proyecto esto no sería un problema en sí ya que como paso previo al algoritmo podríamos indicarle cual es la posición y tamaño de este logo en el caso de que influyera.

2. La suposición del logo como si fuera césped.

   Esta suposición es la más lógica, pero podría darse el caso de que hubiera algún jugador por detrás de este logo. En este caso, el único inconveniente que vemos al suponer esto es que no identificara a dicho jugador pero también sabemos que este jugador no influirá en gran medida en la jugada (al no estar la imagen centrada sobre él) y será identificado cuando salga del rectángulo.

También notar que en la imagen mostrada, aunque las imágenes sean distintas se trata de la misma imagen que al ser capturada desde Matlab por el editor de texto las modifica.

Filtros morfológicos

Una vez eliminado el logo de la cadena el siguiente paso previo es pulir las posibles impurezas que haya cometido el algoritmo de segmentación. Estas impurezas no son errores ya que suelen ocurrir por el buen funcionamiento del algoritmo de segmentación. Sin embargo hay ocasiones en las que no necesitamos un nivel de detalle

óptimo ya que el objetivo principal es intentar obtener una imagen lo más cercana posible a la ideal debido al funcionamiento de las rutinas anteriormente citado.

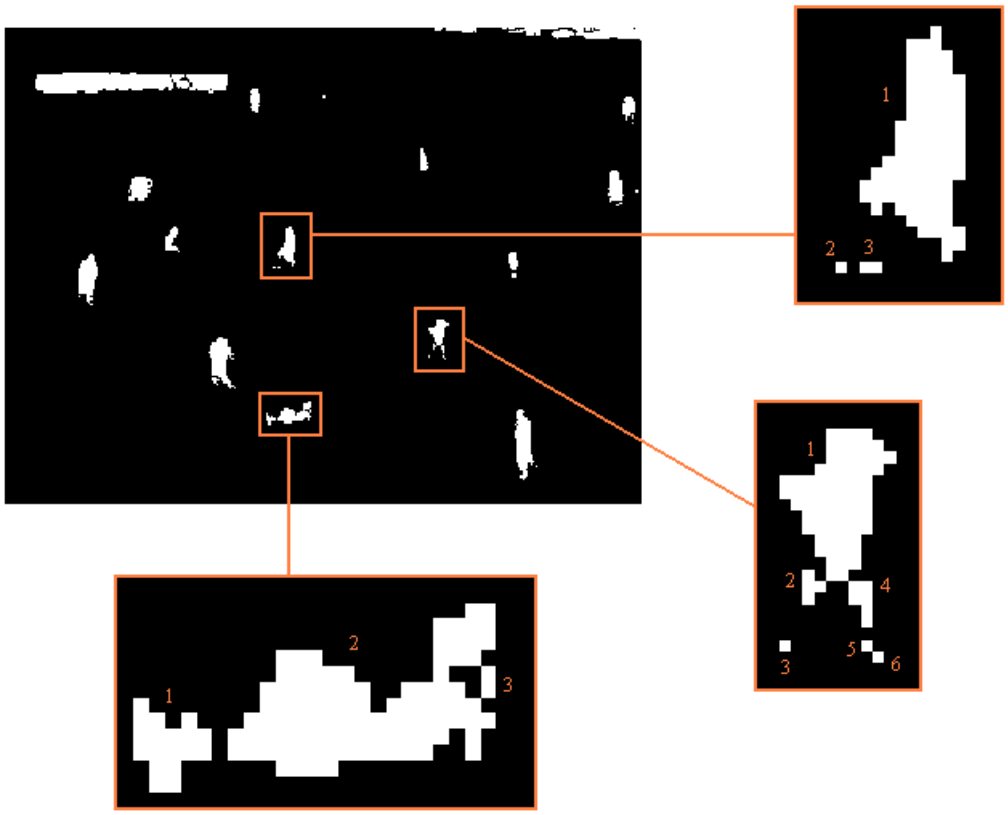


Imagen 30 - Ejemplo rutinas sobre imagen sin acciones previas

Como podemos observar en esta imagen, si directamente procesáramos la salida de la segmentación con la rutina antes citada los resultados no serían satisfactorios ya que identificaríamos muchos más jugadores de los que realmente son. Podemos ver en la imagen como el algoritmo de segmentación funcionaría de manera óptima (ya que podemos intuir incluso la pose de cada jugador) pero no podríamos sacar beneficio de ello ya que el algoritmo identificaría hasta seis jugadores cuando realmente debería identificar uno (el caso peor de los mostrados). Es por esto por lo que el uso de los filtros morfológicos se hace estrictamente necesario.

Una operación morfológica es el resultado de una operación no lineal entre un conjunto de puntos de la imagen original (A) y un conjunto de puntos conocido como elemento estructurante (EE) [27]. El valor en cada píxel en la imagen de salida depende del valor de ese píxel en la imagen de entrada y su relación con la vecindad (definida a través del elemento estructurante).

Estas operaciones morfológicas son llevadas a cabo por los denominados **filtros morfológicos**. Entre los filtros morfológicos más habituales encontramos [28]:

- **Erosión** - Θ

  La fórmula general de la erosión es la siguiente (siendo B el EE):

$$E_B(A) = A \ominus B = \{x \mid B_x \subset A\}$$

Ecuación 6 - Erosión

La erosión reduce el tamaño de la imagen mediante la eliminación de los grupos de píxeles en los que el elemento estructurante no cabe. El elemento estructurante va a tener un píxel que hará de centro (marcado con X) y por el que se medirá si cabe en la imagen o no.

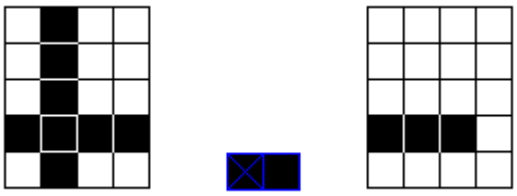


Imagen 31 - Ejemplo erosión

Una aplicación típica de esta operación es la eliminación de detalles irrelevantes.

- **Dilatación -** $\oplus$

La fórmula general es la siguiente:

$$D_B(A) = A \oplus B = \{x \mid (\hat{B})_X \cap A \neq 0\}$$

Ecuación 7 - Dilatación

La dilatación consiste en reemplazar cada uno de los píxeles de la imagen de salida por el elemento estructurante. Al igual que la erosión, tendrá un píxel que hará de centro para fijar el punto de referencia a la hora de reemplazar.

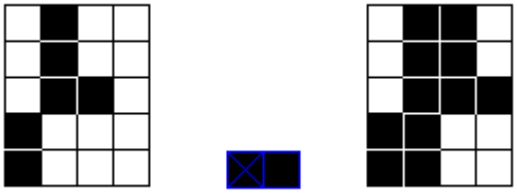


Imagen 32 - Ejemplo dilatación

Una aplicación típica de esta operación es rellenar huecos y agujeros en la imagen.

Añadir llegados a este punto que la erosión y dilatación no son operaciones inversas. Si se dilata y luego se erosiona (o al revés) no se consigue la imagen original. Esto será importante para nosotros ya que utilizaremos ambos filtros.

- **Apertura -** ^{o}

La fórmula general es la siguiente:

$$\gamma_B(A) = A^o B = (A \ominus B) \oplus B$$

Ecuación 8 - Apertura

La apertura es una combinación de la operación de erosión y de dilatación. La salida se obtiene desplazando el elemento estructurante por A y eliminando las zonas por las que no puede pasar.

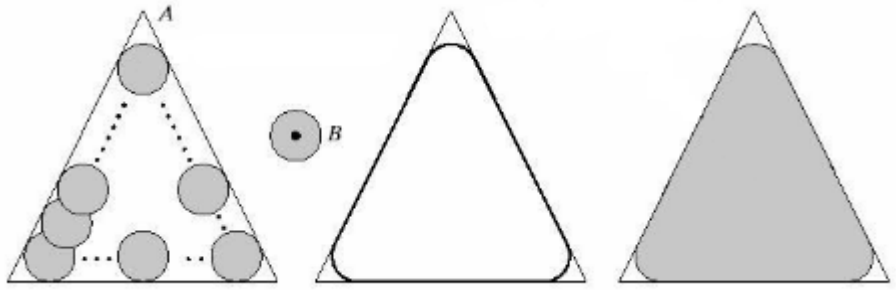


Imagen 33 - Ejemplo apertura[18]

- **Cierre - •**

La fórmula general es la siguiente:

$$\varphi_B(A) = A \bullet B = (A \oplus B) \ominus B$$

Ecuación 9 - Cierre

El cierre es una combinación de la operación de dilatación y de erosión. Al contrario que en la apertura, los desplazamientos de el elemento estructurante van por fuera de la frontera de A.

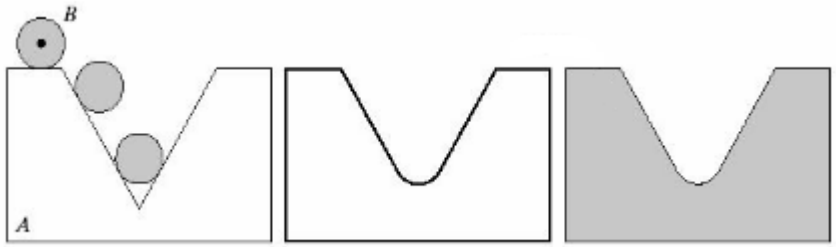


Imagen 34 - Ejemplo cierre[19]

Una vez explicados los 4 filtros morfológicos más conocidos, vamos a exponer cuales han sido los utilizados por nosotros para pulir los pequeños defectos derivados de la segmentación y en que medida afectan a la imagen resultante en cada caso. El efecto que causaría en nuestro algoritmo el utilizar la imagen sin pasar por este filtrado ya ha sido mostrado en la imagen del comienzo del apartado.

Para poder explicar la elección de cada filtro vamos a ir mostrando los distintos resultados que obtenemos para cada caso con una imagen elegida de forma que se pueda ver con claridad los beneficios de su uso.

[18] Imagen obtenida de libro "Digital Image Processing Using Matlab" de R.González, R.Woods y S. Eddins.
[19] Imagen obtenida de libro "Digital Image Processing Using Matlab" de R.González, R.Woods y S. Eddins.

La imagen del jugador elegida es bastante singular ya que el jugador está tumbado sobre el césped tras una caída (coincide con uno de los ejemplos mostrados anteriormente). La imagen original y la resultante tras la segmentación son las siguientes:

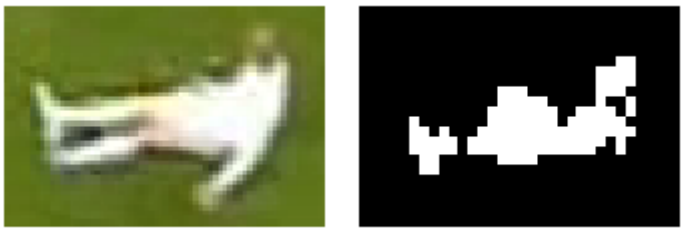


Imagen 35 - Ejemplo filtrado (Paso segmentación)

Como hemos explicado anteriormente, esta imagen daría lugar a error ya que serían identificados tres jugadores en lugar de uno.

La primera idea viendo el funcionamiento de los filtros típicos analizados antes sería la de usar la herramienta de dilatación. De esta forma, las partes que no están unidas completamente podrían llegar a unirse y formar un solo conjunto. Esto sería lo lógico pero, como detallaremos en mayor medida en el siguiente apartado de "Problemas encontrados y decisiones tomadas" el primer filtro que vamos a usar es el de erosión.
A modo de resumen, esto se debe a que existen algunos casos en los que se detectan puntos aleatorios en medio del césped (sobre todo en las líneas blancas del campo) e identificados como jugadores. La erosión elimina estos pequeños puntos y así no cometeremos errores posteriores.

Paso 1. Erosión

El primer filtro a utilizar es el filtro de la erosión. Como los jugadores se pueden distinguir en mayor medida por su posición vertical (altura) que por su posición horizontal (anchura), vamos a elegir como elemento estructurante para este filtro una línea de 5 píxeles (equivale a 2.5 píxeles entre el punto de referencia de la línea y el extremo) a 90º (posición vertical). El resultado es el siguiente:

Imagen 36 - Proceso de erosión

Una vez realizada la erosión para conseguir eliminar los puntos aleatorios el siguiente paso es intentar unir las partes de un mismo jugador para que no realice una mala identificación. Para ello cada una de las partes las tenemos que "agrandar" de forma que se unan y formen el conjunto. Y para ello vamos a usar la herramienta de dilatación.

Paso 2. Dilatación

Como bien hemos mencionado anteriormente, en este paso debemos realizar una compactación de aquellos jugadores que hayan sido identificados por más de un conjunto. En este caso, puede ser porque la segmentación haya generado varios conjuntos a partir de la imagen original o bien porque, tras la erosión inicial, hayan pasado a formar varios conjuntos cuando antes estaba formado por uno solo de ellos. De esta forma, como también se ha mencionado en las capacidades de la dilatación, conseguiremos rellenar huecos y unir conjuntos.

Vamos a usar como elemento estructurante un cuadrado de 7 píxeles de lado donde el punto de referencia será el centro del cuadrado. Se ha elegido dicha figura geométrica ya que la imagen resultante de salida tras pasar por dicho filtro no tendrá esquinas ni elementos que pudieran hacernos creer que hay más de un conjunto.

A partir de la imagen erosionada anterior, vamos a observar la evolución tras haber pasado por el filtro dilatador.

Imagen 37 - Proceso de dilatación

Como podemos observar, el objetivo por el que se ha fijado el filtro de dilatación se cumple ya que obtenemos un solo conjunto de una imagen en la que había 3 de ellos (recordemos que pertenecen a un solo jugador).

Paso 3. Cierre

Como último paso vamos a utilizar el filtro del cierre. En este paso lo único que vamos a intentar obtener son figuras más uniformes y sin cantos muy abruptos. Este cierre lo vamos a realizar con un cuadrado de 9 píxeles de lado.

Imagen 38 - Proceso de cierre

Aparentemente se podría prescindir de este paso ya que en algunos casos no obtenemos resultados aparentes pero hay determinados casos en los que aparecen huecos de gran tamaño que sin este cierre darían lugar a error. Podremos observarlo con mayor detalle en el apartado de problemas encontrados.

Por lo tanto, la acción previa formada por el uso de los filtros morfológicos estaría compuesta por la concatenación de tres de los filtros más comunes: erosión, dilatación y cierre.

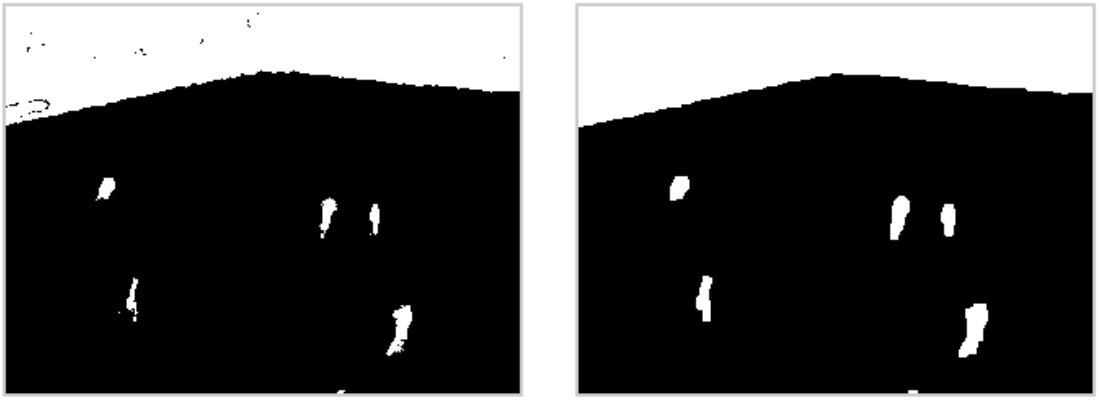


Imagen 39 - Evolución de imagen segmentada a imagen filtrada

En esta imagen podemos ver el resultado del filtrado sobre un frame concreto de un partido de la base de datos. Comprobamos como los puntos perdidos y los huecos desaparecen y como los jugadores se hacen tienen formas más uniformes y más grandes de lo habitual. Como veremos en los apartados posteriores esto no será problema ya que la rutina a usar se va a basar en el cálculo del centroide de cada conjunto.

Paso a binario y eliminación de público erróneo.

En este paso previo vamos a realizar tres tareas:

La primera de ellas va a ser sencillamente el paso de nuestra imagen al sistema binario en el que cada frame va a estar formado por unos o ceros. El color blanco se corresponderá con cero y el color negro se corresponderá con uno. El algoritmo que realiza esto es un decisor simple que a partir del valor de cada píxel (que será negro o blanco) obtiene su correspondiente valor binario.

El segundo paso es un poco más complejo y se realiza debido a los problemas que surgen a la hora de eliminar el público y las gradas. En algunos casos, al realizar la segmentación parte de público y de la grada es identificado como césped. Esto no es erróneo ya que puede haber determinadas zonas donde predomine el color verde y nuestro algoritmo de segmentación no sabe si es césped o es público. De esta forma, se podría identificar determinados conjuntos sobre el público y que el algoritmo entendiera que son jugadores. Este caso aparece en contadas ocasiones pero hace que el algoritmo cometa errores que se podrían paliar de forma relativamente sencilla.

La forma en la que vamos a descartar este público erróneo será descrita en el apartado de "Problemas encontrados y decisiones tomadas" ya que se utiliza la rutina de detección de conjuntos de Matlab y es explicada en los puntos siguientes.

El tercer paso será el invertido al primero, ya que deberemos volver a dejar la imagen tras la eliminación de público erróneo en el formato de color (0-255) para poder utilizar la rutina de Matlab.

El esquema básico de estas acciones previas es el mostrado por el siguiente diagrama:

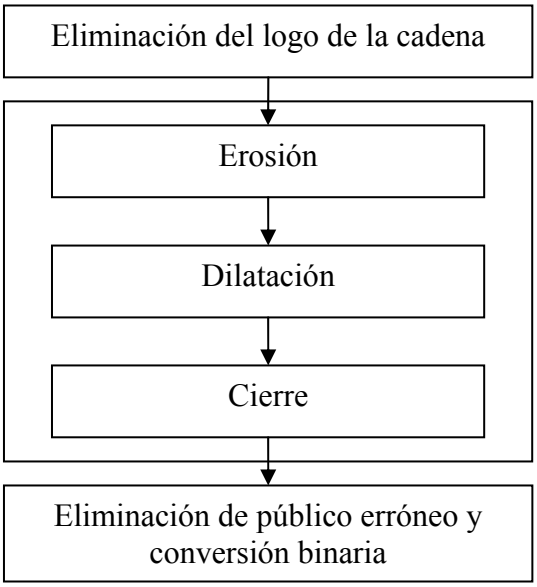


Imagen 40 - Esquema acciones previas

4.6.3.2 Imagen ideal

Tras realizar la segmentación del frame y hacerlo pasar por las tareas previas hemos conseguido el objetivo buscado desde el principio: intentar conseguir una imagen lo más ideal posible para la ejecución de la rutina posterior. Esta imagen ideal debe ser parecerse a la mostrada en el apartado 4.6.2 para que así el procesamiento sea lo mejor posible. Una imagen ideal obtenido de forma real por nuestro sistema sería la siguiente:

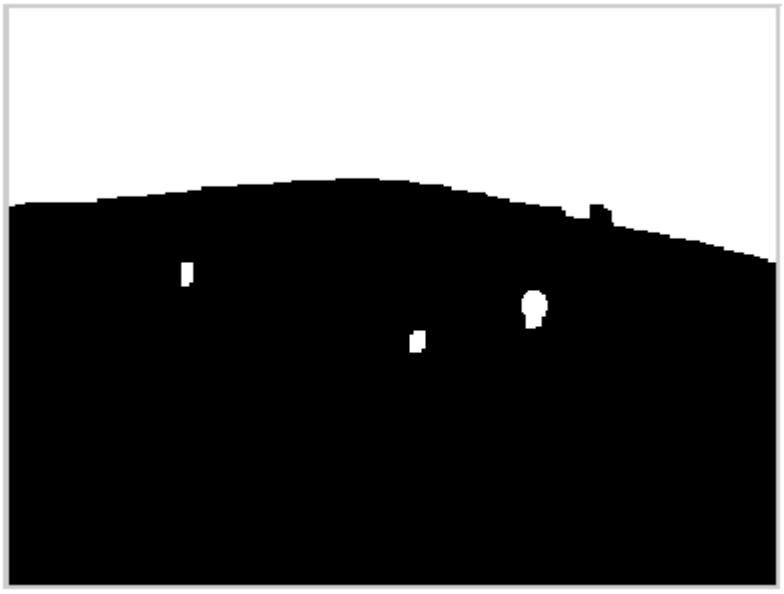


Imagen 41 - Imagen "ideal" obtenida

4.6.3.3 Rutina de identificación de jugadores

Tras la eliminación del público erróneo y la obtención de la imagen “ideal” vamos a ejecutar la rutina clave de Matlab para la identificación de conjuntos. Esta rutina va a estar formada por dos funciones de Matlab: bwlabel y regionprops.

bwlabel

El funcionamiento básico de esta función es el siguiente:

Si introducimos una matriz (en nuestro caso una imagen) formada por distintos valores numéricos (entre cero y cualquier valor) este algoritmo asigna el valor ‘0’ a todos los píxeles que formen el fondo (en nuestro caso el color negro) y va asignando una etiqueta a cada conjunto que no forme parte del fondo de la imagen. Esta etiqueta será un ‘1’ para el primer conjunto, un ‘2’ para el segundo y así sucesivamente.
Como comentamos en la introducción, esta rutina va a servirnos de gran ayuda ya que podremos identificar todos los conjuntos que aparecen en nuestra imagen ideal y etiquetarlos como posibles jugadores dentro de la imagen.

Un esquema del funcionamiento de esta función con la imagen utilizada en la introducción sería el siguiente:

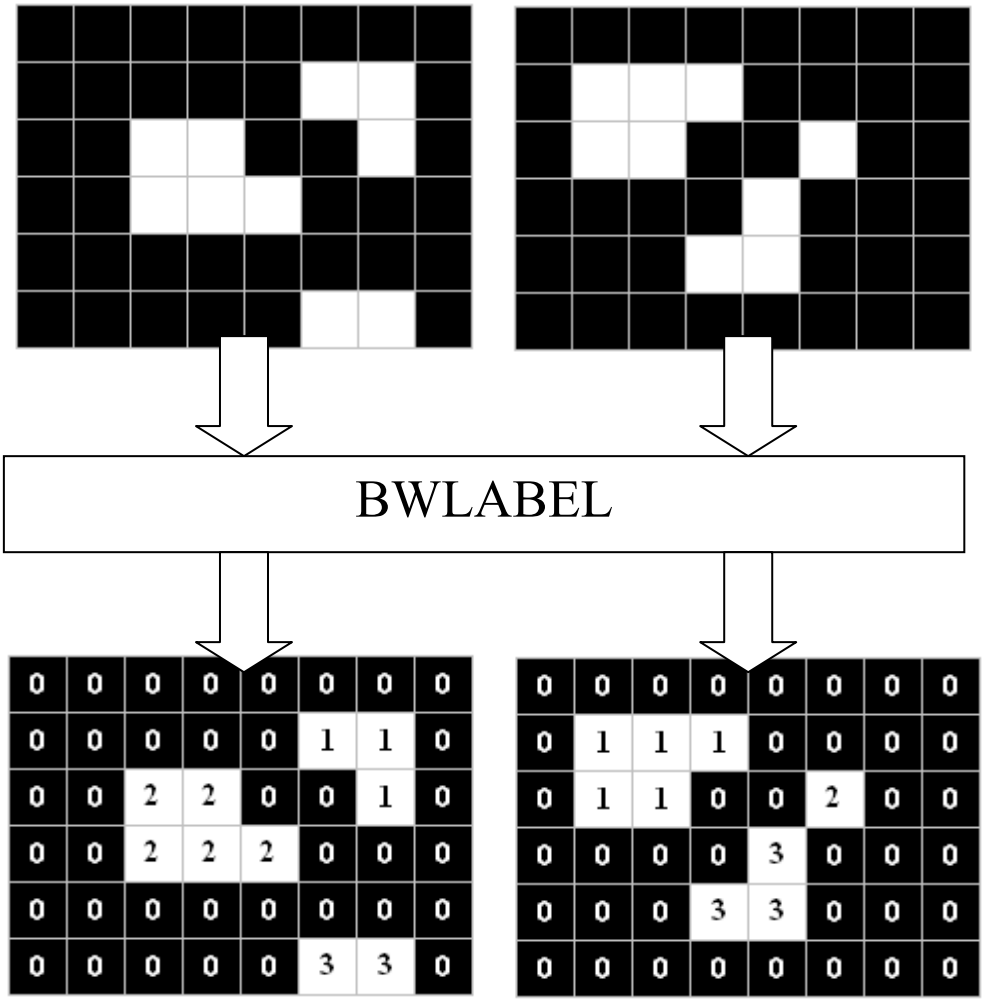


Imagen 42 - Ejemplo función 'bwlabel'

Como podemos observar, esta función nos permite etiquetar los píxeles de cada conjunto. Este ejemplo está hecho a gran escala para entender el funcionamiento. En una imagen real cada conjunto estaría formado por muchos más píxeles.

Una vez que hemos etiquetado los distintos conjuntos por los que va a estar formada la imagen nos planteamos una forma de asignar a cada uno de ellos una coordenada fija de forma que podamos conocer de forma exacta en que posición se encuentra. Esta coordenada nos va a servir para poder calcular el movimiento de cada jugador entre un frame y otro. Para el cálculo de estas coordenadas vamos a hacer uso de la función de matlab *regionprops* unida al cálculo de centroides.

regionprops

La función *regionprops* mide una serie de propiedades para cada conjunto de una imagen binaria. Para nuestro caso, elegimos la propiedad del cálculo del centroide de la imagen ya que obtendremos el centro de masas de cada región. Este cálculo será válido para nuestro algoritmo ya que obtendremos el centro de masas de cada "mancha" que representa a cada jugador. De esta forma, habremos obtenido un solo punto a partir de un conjunto.

Si utilizamos las imágenes anteriores podemos ver en que posiciones ubica MATLAB el centroide para cada caso:

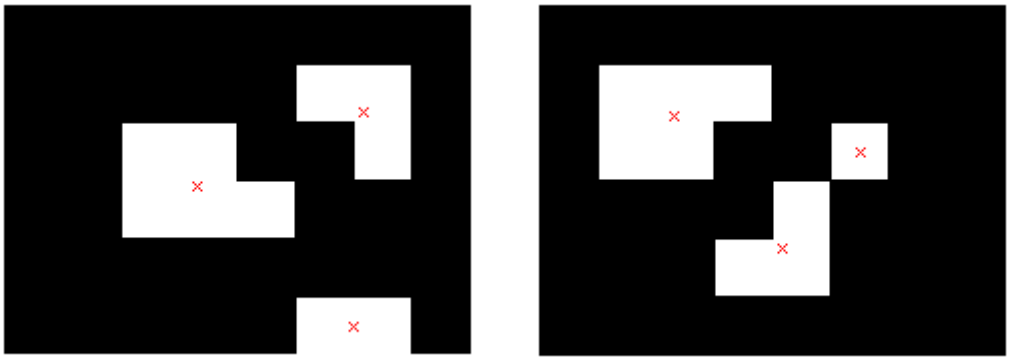


Imagen 43 - Ejemplo centroides

Como podemos ver, en estos casos sencillos el centroide se ubica relativamente cerca del centro de cada uno de las figuras geométricas. Esto es lógico ya que en las figuras geométricas en dos dimensiones el centro de masas coincide con el centro geométrico.

Para poder ver el algoritmo sobre nuestras imágenes vamos a mostrar que coordenadas se obtienen sobre la imagen ideal. Se observa como calcula las coordenadas de los tres jugadores en las posiciones (148.4184, 221.3163), (341.4020, 278.3824) y (438.3209, 250.6493).

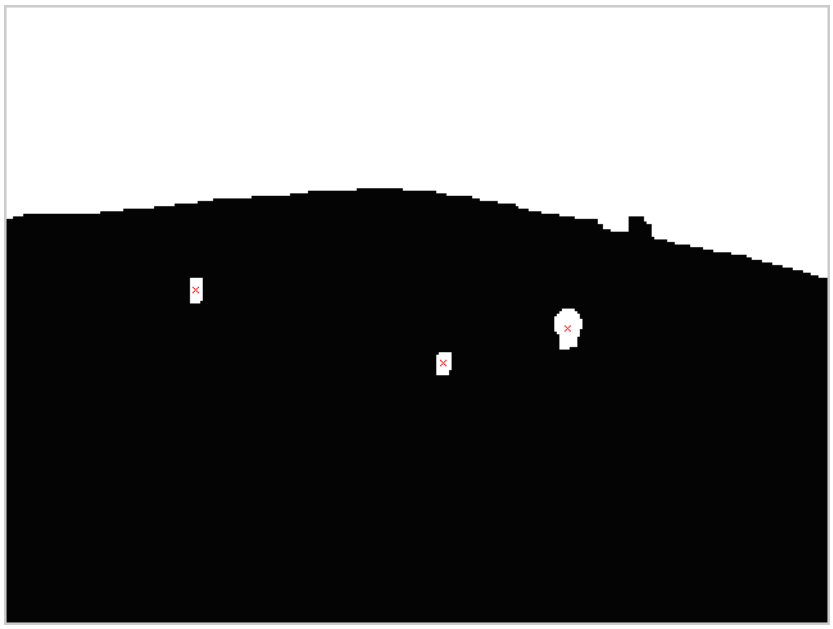


Imagen 44 - Cálculo de centroides sobre imagen ideal

4.6.4 Problemas encontrados y decisiones tomadas

Los problemas encontrados a lo largo de este apartado de detección de jugadores han sido bastantes. Vamos a explicar detalladamente cada uno de ellos y las decisiones que

se han tomado para intentar eliminar estos problemas o reducir su influencia sobre el resultado final.

4.6.4.1 Introducir el filtro de erosión antes del filtro de dilatación

Como se explicó anteriormente, la forma lógica de realizar el filtrado sería contraria a la usada: el primer paso siempre sería una dilatación para unir a todos aquellos jugadores que, a causa de la segmentación, han sido identificados por más de un conjunto.

Sin embargo, tras realizar una serie de pruebas llegamos a la conclusión de que era más interesante de cara al buen funcionamiento del filtrado realizar primero una erosión. Estas pruebas indicaron que en algunos puntos del campo no eran identificados como césped (sobre todo en las líneas blancas aunque aparecen en cualquier zona). Estos puntos eran demasiado blancos y el algoritmo de segmentación no filtraba bien.

Como posibles soluciones se pensaron dos modelos a seguir:

- Modificar el algoritmo de segmentación
- Realizar una erosión anterior a la dilatación

La primera posibilidad se descartó ya que una modificación en el algoritmo de segmentación influiría en la detección del césped y los jugadores y no era viable. La segunda posibilidad dio buenos resultados y se observó también que no influía en el buen funcionamiento del algoritmo posterior por lo que fue elegida como solución final.

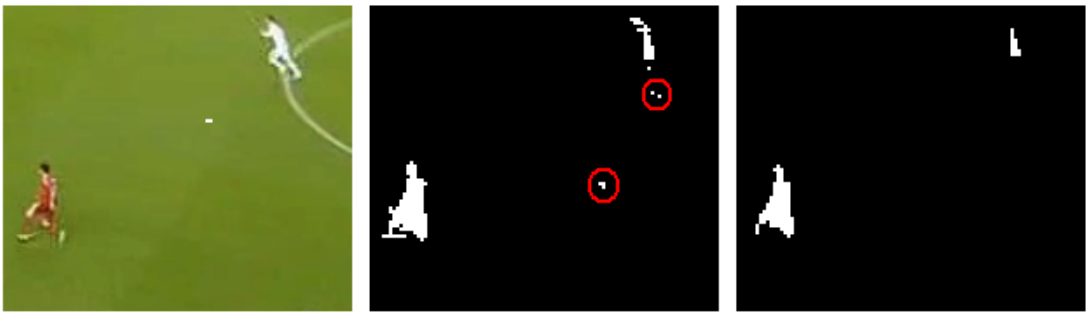


Imagen 45 - Ejemplo uso de erosión previa a dilatación

En este ejemplo podemos ver como aparecen tres puntos que no deberían ser considerados jugadores. En este caso, tras el uso de la erosión, estos puntos desaparecen por lo que no son identificados como jugadores y el proceso de detección adquiere mayor calidad al obtener menos errores.

4.6.4.2 El uso del cierre, ¿es obligatorio?

Podemos ver en el siguiente ejemplo un caso en el que la eliminación del cierre supone un error de identificación:

Imagen 46 - Ejemplo uso del cierre

En el centro de la imagen vemos el jugador original y podemos ver a la izquierda la imagen resultante usando el cierre y a la derecha la misma imagen sin usar el cierre. En este caso vemos como el uso del cierre sí tiene beneficio sobre el algoritmo ya que ayuda a identificar un solo conjunto mientras que sin usarlo identificaríamos dos de ellos.

Este error fue identificado en la fase de pruebas con la posterior introducción del cierre para su solución. Esta solución no afecta en gran medida a los conjuntos que no sufren el problema indicado en la imagen anterior ya que, como se ha comentado anteriormente, el cierre en esos casos hace que la figura sea más uniforme y sin cantos y no afecta (en gran medida) al cálculo del centroide.

4.6.4.3 Eliminación del público erróneo

Como se mencionó en el apartado que trataba este tema, este caso es bastante especial ya que no suele ocurrir con frecuencia aunque sus consecuencias son bastante dañinas para el funcionamiento del algoritmo.

Este fallo se encontró en muchas ocasiones con la segmentación píxel, en la que realizábamos la comparación de todos y cada uno de los píxeles con el píxel de referencia. En estos casos y al ser una comparación tan exhaustiva, muchos de los píxeles del público eran identificados como césped. Podemos ver en la siguiente imagen como encontraba determinadas zonas en las que había conjuntos donde no debería de haberlos:

Imagen 47 - Segmentación de público errónea

Se han marcado en rojo los conjuntos que serían identificados como jugadores y, por lo tanto, errores en la identificación. Una vez conocido el funcionamiento del sistema de

etiquetado de conjuntos de Matlab, la forma en la que se decidió eliminar estos errores es fácil de comprender:

El sistema de etiquetado (bwlabel) identifica a cada conjunto blanco sobre fondo negro. Por lo tanto, el primer paso que vamos a realizar es un volteo (convertir el negro en blanco y viceversa) de la imagen para poder identificar los subconjuntos que debemos eliminar. Si volteamos la imagen, el etiquetado estará compuesto por un conjunto muy numeroso que sería el césped y el resto de puntos que queremos eliminar:

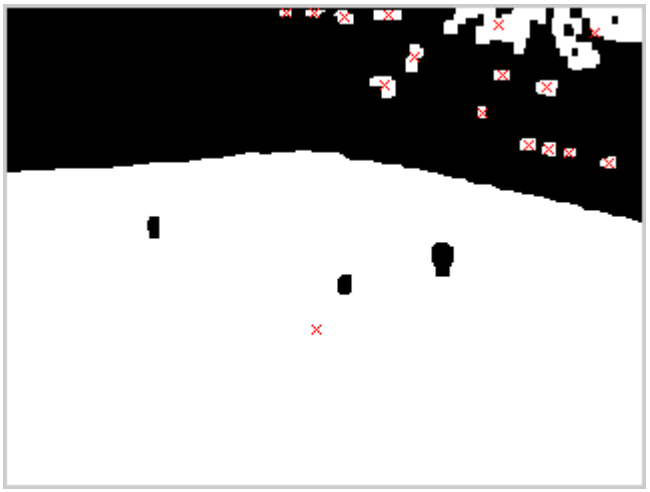


Imagen 48 - Volteo de imagen

El siguiente punto sería obtener el número más repetido de todas las etiquetas designadas por bwlabel. De esta forma, obtendríamos el número de la etiqueta del césped y así podríamos distinguir los números de las etiquetas de los conjuntos que queremos eliminar. Una vez obtenidos estas etiquetas erróneas, bastaría con cambiar el valor de la etiqueta al valor '0' (negro) y volver a voltear la imagen para obtener la imagen inicial en la que los jugadores son blancos y el resto negro. En la imagen mostrada, el centroide que identifica al césped sería el único que no eliminaríamos mientras que los conjuntos del resto de centroides serían eliminados.

Sin embargo, al realizar la mejora de segmentación bloque y calcular las distancias con 4 píxeles en lugar de con uno este error prácticamente desaparecía por lo que su uso no sería necesario. Se ha decidido dejar ya que su carga procesal no es muy alta y podría evitarnos este problema en caso de reproducirse o por ejemplo, si en las vallas del público hubiera algún cartel con un verde similar al del césped.

4.6.4.4 Problema de las bandas

Este problema se identifica tras realizar pruebas aleatorias para comprobar resultados del algoritmo. El problema es sencillo de explicar y bastante complicado de eliminar: consiste en la detección de aquellos jugadores que se encuentran muy cercanos a la banda del campo. Nuestro algoritmo de segmentación lo identifica y lo etiqueta como jugador pero el problema llega con el algoritmo de identificación ya que se basa en la detección de conjuntos sobre fondo negro. Como hemos explicado anteriormente, el público también es detectado como un conjunto por lo que si el jugador esta demasiado pegado a la banda (o incluso por encima "visualmente hablando" del público) será unido a este conjunto y no será identificado.

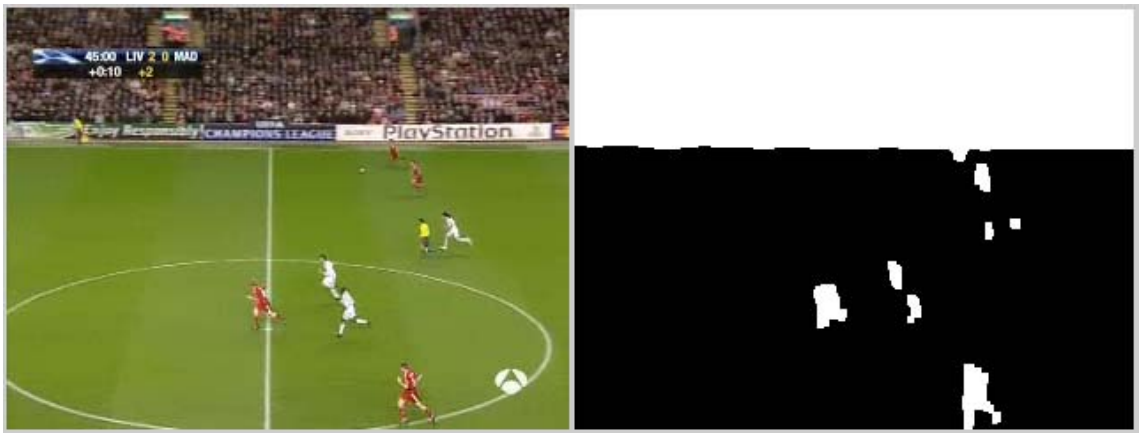


Imagen 49 - Problema bandas

En este ejemplo podemos ver como donde debería identificar un jugador no lo identifica ya que se une al conjunto del público. Al estar los jugadores tan juntos y realizar una dilatación, los píxeles se unen y el algoritmo de identificación no cuenta con ellos. Como se ha mencionado anteriormente, este problema es bastante difícil de atenuar ya que con nuestro algoritmo no somos capaces de identificar cuándo son jugadores y cuándo es público si ambos están unidos.

Una posible solución que se podría implementar sería intentar seguir la línea de campo y usarla como barrera de forma que una parte de la barrera fuera público y la otra fuera césped y jugadores. Esta implementación sería bastante complicada de realizar y estaría fuera del proyecto que nos ocupa.

4.6.4.5 Unión de jugadores

Otro de los problemas encontrados a la hora de realizar pruebas fue el de la unión de los jugadores:

Imagen 50 - Problema de la unión de jugadores

En este caso el problema también es bastante grave ya que si varios jugadores están muy juntos entre ellos, al realizar la dilatación en el algoritmo de filtrado pasarán a ser un solo conjunto con la posterior identificación de un solo centroide en lugar de varios (en el ejemplo debería de haber identificado 4 jugadores). Además, en el ejemplo

propuesto, también coincide con el anterior problema de unión a las bandas por lo que ninguno de esos 4 jugadores sería identificado.

La solución también es bastante compleja y se piensa en una segmentación en la que tuvieran cabida los colores de los uniformes de los jugadores. Si tuviésemos en cuenta esa distinción por color, la detección de varios jugadores que estuvieran cercanos sería más fácil siempre y cuando fueran jugadores de distintos equipos. Sin embargo, en nuestro algoritmo esta distinción no se ha tenido en cuenta por lo que la unión de jugadores se tomará como un fallo aceptable.

Una vez se ha cometido este error, el procesado de los jugadores es normal ya que frame a frame estos jugadores se irán distanciando y el algoritmo los identificará como nuevos jugadores aunque se perderá el seguimiento concreto de cada jugador (lo veremos con más detalle en el cálculo del movimiento).

4.6.4.6 Eliminación de centroide del público

Este problema viene derivado de la detección del público como conjunto, ya que nuestro algoritmo de segmentación solo identifica el césped. Si realizamos la detección tal y como obtenemos la imagen tras la segmentación y filtrado obtendríamos algo como esto:

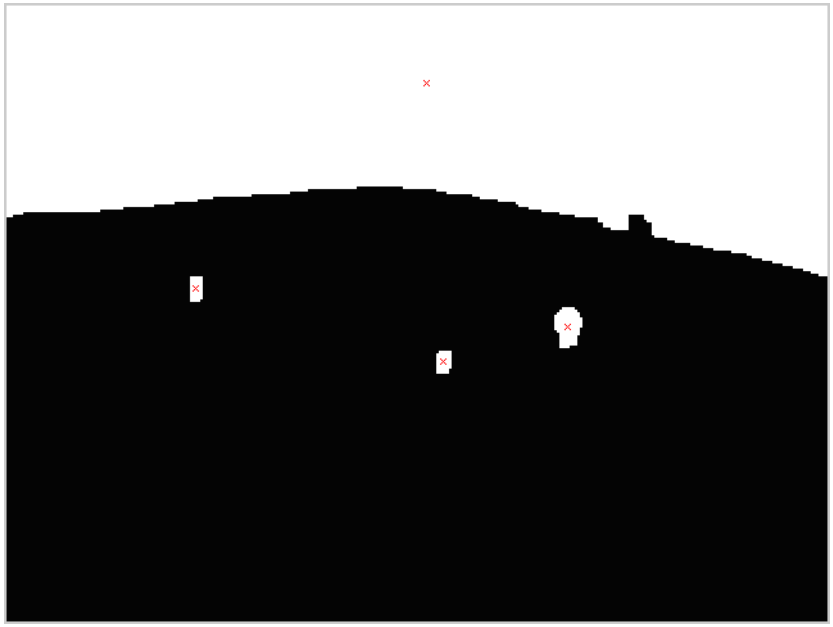


Imagen 51 - Centroide erróneo debido al público

Podemos ver como identifica a un centroide en el público ya que forma un conjunto que acaba en las dos esquinas superiores. La forma de eliminar este conjunto es muy sencilla: tras el etiquetado, la etiqueta más repetida será la de este conjunto, de forma que conocer su valor y eliminarlo será relativamente fácil.

4.7 Cálculo de movimiento

4.7.1 ¿Qué tipo de movimiento vamos a calcular?

Tras los pasos previos de captura de imágenes, segmentación, filtrado y detección llegamos al último paso del sistema: el cálculo del movimiento.

En el apartado anterior llegamos a la conclusión de que a partir de una serie de conjuntos podíamos obtener unos centroides y, a partir de esos centroides, obtener unas coordenadas dentro de cada frame. Estas coordenadas, como se mencionó anteriormente, son coordenadas relativas al frame, no relativas al campo de fútbol.

Si realizamos todo el proceso centrándonos en un jugador en concreto, el algoritmo hasta este punto nos daría como salida una coordenada dentro del frame. Si, pasado un periodo de tiempo, volvemos a ejecutar el algoritmo y nos centramos en el mismo jugador la salida sería otra coordenada distinta a la anterior. La diferencia entre la coordenada del primer frame y la coordenada del segundo sería el movimiento del jugador relativo al frame.

4.7.2 ¿Cómo vamos a realizar este cálculo?

El cálculo de este movimiento es relativamente sencillo aunque el algoritmo de cálculo tendrá que tener muchos puntos a considerar cuando tengamos en cuenta que hay que calcular el movimiento de todos los jugadores del frame y que pueden aparecer nuevos jugadores o desaparecer algunos que ya estaban en el frame.

El primer punto que debemos pensar es el intervalo de captura entre frames. Como primera toma de contacto se tomó 1 segundo de intervalo lo que suponía un intervalo de 25 frames. Al realizar pruebas sobre este intervalo se comprobó que era un intervalo demasiado largo ya que en un segundo el movimiento de jugadores es demasiado grande y se pierde el seguimiento. Por lo tanto, se tomó como intervalo de captura cada 5 frames, lo que equivale a 200ms. Con este intervalo el seguimiento de los jugadores es aceptable.

Por lo tanto, la fórmula inicial de cálculo del movimiento sería la siguiente:

$$\boxed{x(1) = x(0) + mj \Rightarrow mj = x(1) - x(0)}$$

Ecuación 10 - Cálculo inicial de movimiento

En esta ecuación podemos ver que el movimiento de un jugador (*mj*) está formado por la diferencia entre la coordenada en el frame 1 (*x(1)*) menos la coordenada en el frame 0 (*x(0)*).

Como veremos más adelante, esta fórmula se verá modificada para tratar de tener en cuenta e intentar reducir la influencia del movimiento propio de la cámara.

4.7.3 El algoritmo de cálculo de movimiento

Para poder calcular el movimiento de los jugadores es necesario realizar el algoritmo sobre dos frames distintos. Como se ha mencionado en el punto anterior, la distancia entre los frames será de 0.2 segundos. En el primer frame, cada jugador lleva asociada una coordenada y nuestro objetivo será identificar a que coordenada corresponde en el segundo frame. Por lo tanto, el mapeo de los jugadores va a estar formado por entradas de la siguiente tabla:

Frame 1		Frame 2		Representación
Posición X	Posición Y	Posición X	Posicion Y	Color y símbolo

La representación indica el color y símbolo por el que va ser representado cada jugador para poder evaluar el buen funcionamiento del sistema en la fase de resultados (lo veremos más adelante).

Los tres posibles estados de este mapeo son los siguientes:

- Las dos coordenadas rellenas, que indican que ha habido movimiento entre un frame y otro. (Caso 1)
- Solo rellena la coordenada del frame 1, que indica que el jugador estaba en el frame 1 pero no en el frame 2. En la imagen sería que el jugador ha salido del encuadre. (Caso 2)
- Solo rellena la coordenada del frame 2, que indica que el jugador está en el frame 2 pero no estaba en el frame 1. En la imagen sería que el jugador aparece nuevo en el encuadre. (Caso 3)

Para poder rellenar esta tabla de mapeos, vamos a seguir el siguiente procedimiento:

Para cada punto del frame 1:

1. Calculamos el punto del frame 2 más cercano a él y la distancia a la que se encuentran. Este punto del frame 2 será probablemente la posición a la que se haya desplazado después del intervalo de detección (0.2 segundos).

2. Después tenemos un valor umbral de distancia con el que vamos a comparar. Si la distancia al punto más cercano es menor que la umbral, entonces asociaremos el punto que hemos obtenido del frame 2 con el que estamos comparando del frame 1. (Caso 1)

3. Si aparecen puntos del frame 1 que no podemos asociarlo a ningún punto del frame 2 bien porque su distancia es superior a la umbral o bien porque el punto al que queríamos asociarlo ya está ocupado, debemos añadir la coordenada del frame 1 solamente. (Caso 2)

Una vez hemos hecho esta asociación es necesario introducir los puntos del frame 2 que no han sido asociados. Estos puntos serán nuevos jugadores y, por lo tanto, solo se rellenará la coordenada del frame 2. (Caso 3)

Después, lo único que quedará será rellenar la última columna del mapeo. Para la primera iteración se rellenará de forma automática pero para las sucesivas iteraciones deberemos tener en cuenta los valores anteriores para poder realizar un mejor seguimiento. Esto es, por ejemplo, si en el paso del frame 1 al frame 2 hemos asociado dos coordenadas con una cruz de color azul, para la siguiente iteración (paso del frame 2 al frame 3) esta asociación deberá tener el mismo símbolo y color.

El esquema del algoritmo es el siguiente:

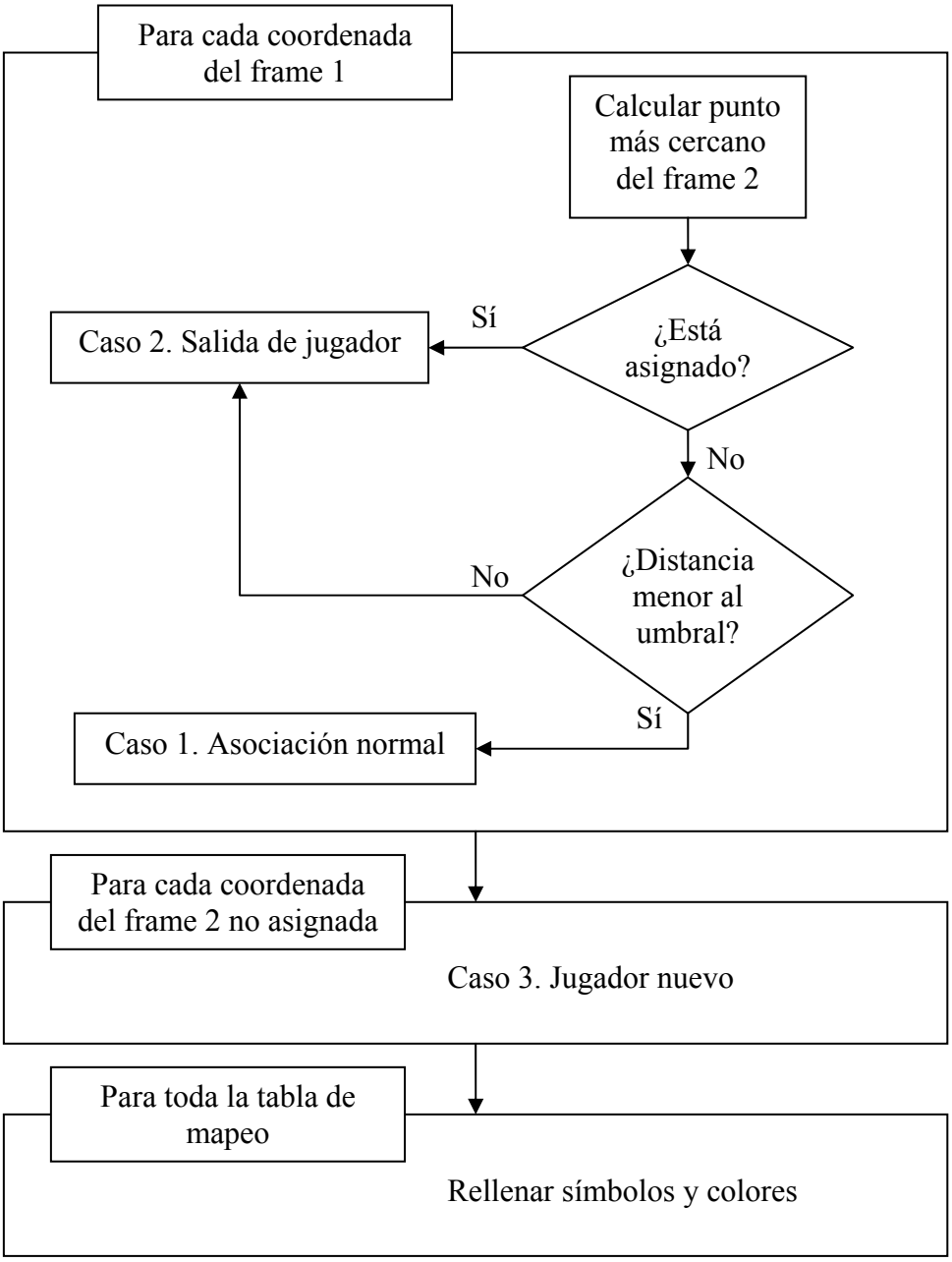


Imagen 52 - Esquema cálculo del movimiento.

Vamos a ver ahora la salida de este algoritmo de forma visual para poder hacernos una idea de cómo representa cada jugador y como calcula el movimiento de cada uno de ellos. Esta prueba vamos a realizarla sobre dos frames, de forma que el intervalo entre ambas imágenes será de 0.2 segundos. En el apartado de "Resultados experimentales" podremos ver cómo funciona el algoritmo sobre un intervalo de tiempo más amplio.

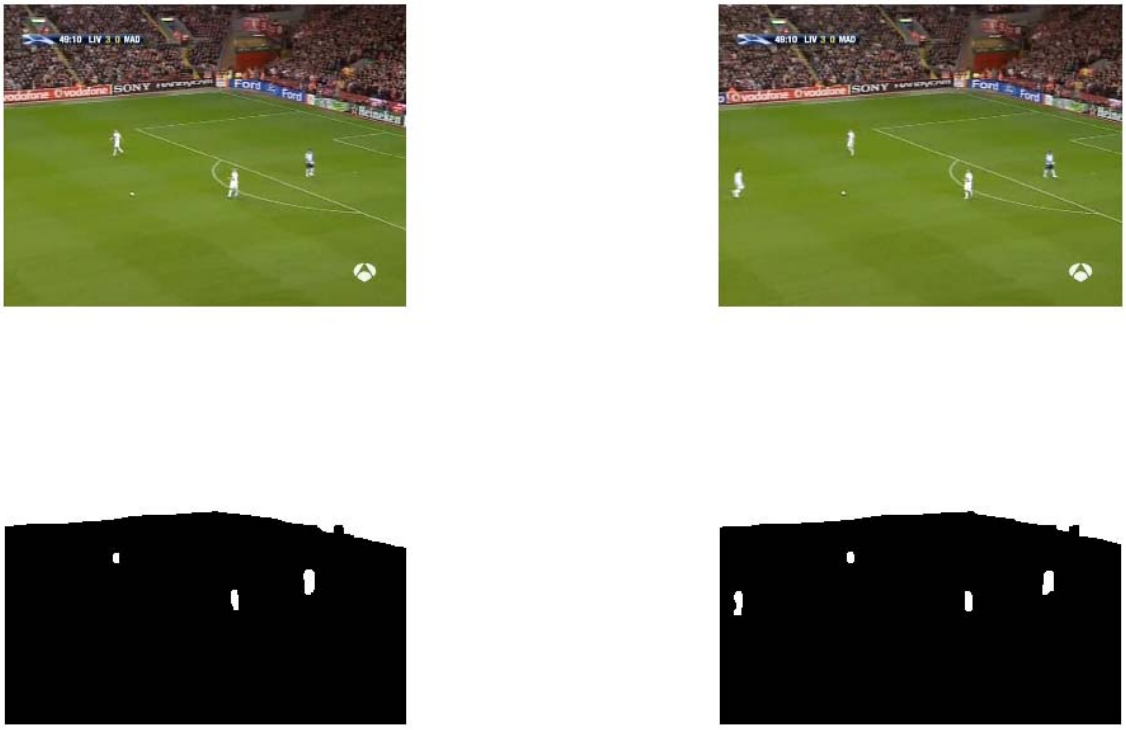


Imagen 53 - Ejemplo de 2 frames

Podemos ver en este ejemplo los dos frames:
A la izquierda podemos ver el frame uno original en la esquina superior izquierda y debajo queda la imagen antes de pasar por la detección de jugadores y el cálculo del movimiento. En la columna derecha podemos ver lo mismo pero para el frame dos. En el frame uno podemos ver que hay dos jugadores y el portero. En el frame dos podemos ver que están los mismos jugadores y aparece un nuevo jugador a recibir el balón por el margen izquierdo. La asignación de los jugadores va a estar representada por los siguientes símbolos:

- Portero, que será representado por una cruz morada.
- Jugador cercano al borde del área, representado por una cruz azul.
- Jugador restante, representado por una cruz roja.
- Jugador nuevo, representado por una cruz amarilla.

Además, el algoritmo realiza una línea de unión entre las dos coordenadas de cada jugador para poder ver con más claridad cual ha sido el movimiento exacto y hacia que dirección se ha realizado [29].

La salida de nuestro algoritmo de cálculo de movimiento sería la siguiente

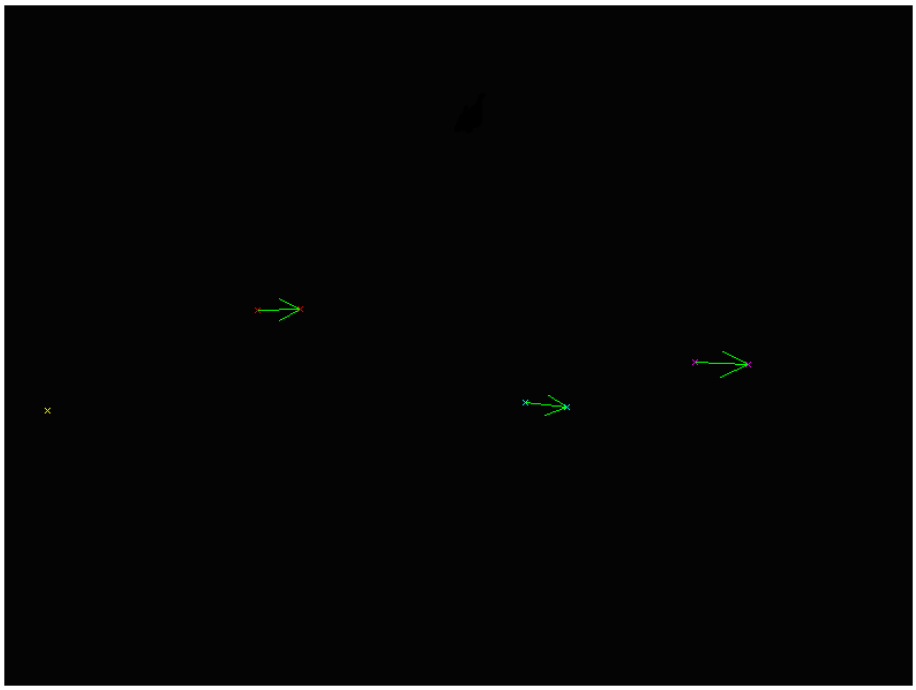


Imagen 54 - Ejemplo cálculo de movimiento

Como se ha comentado antes, podemos ver el movimiento de los tres jugadores y la aparición del cuarto jugador en el segundo frame. También se ha realizado el algoritmo para que nos muestre la dirección del movimiento de los jugadores. Aparentemente este movimiento es erróneo, ya que las flechas indican que los jugadores se han movido hacia la derecha mientras que en las imágenes originales podemos ver que el movimiento debería ser hacia la izquierda. Ninguna de las dos afirmaciones es falsa, los jugadores se han desplazado hacia la izquierda pero a la vez la cámara se ha desplazado hacia la derecha. El movimiento de la cámara es mayor al movimiento de los propios jugadores por lo que nuestro detector (que recordemos funciona con las coordenadas del frame y no del campo de fútbol) indica que el movimiento de los jugadores ha sido hacia la derecha.

Esta diferencia en el cálculo es lo que nos lleva a plantear dos tipos de cálculo de movimiento:

- Movimiento absoluto, que sería el que hemos calculado hasta el momento en el que el algoritmo adquiere tanto el movimiento de los jugadores como el de la cámara.

- Movimiento relativo, en el que se tiene en cuenta el movimiento de la cámara y se aproxima el movimiento de cada jugador aplicando una corrección al movimiento dado por la cámara.

  Como veremos en el siguiente apartado, este segundo cálculo es más laborioso ya que debemos estimar el movimiento de la cámara en cada instante y realizar la corrección sobre el movimiento absoluto.

4.7.4 Problemas encontrados y decisiones tomadas

En este apartado vamos a ver los diferentes problemas que nos encontramos en el desarrollo y prueba del algoritmo y que decisiones se han tomado al respecto para cada uno de ellos. Algunos de los problemas vienen derivados de problemas en los algoritmos anteriores (segmentación, filtrado y detección) pero se han introducido también en este apartado para ver las consecuencias que tiene el cometer estos errores.

4.7.4.1 Unión de jugadores

Este problema viene derivado del problema en el algoritmo de detección anteriormente citado. Recordemos que si varios jugadores estaban muy cercanos, este algoritmo no distinguía entre uno u otro y asignaba un solo conjunto a ambos. La posible solución a este problema ya se explicó en la identificación de los jugadores y consistía en la detección de los jugadores por color, de forma que jugadores de distintos equipos pudieran ser identificados aunque estuvieran juntos. Esta solución sería válida para este caso ya que si encontráramos dos jugadores del mismo equipo muy juntos no habría manera de identificarlos. En la siguiente imagen vamos a ver la detección de varios jugadores que se han unido al realizar la segmentación.

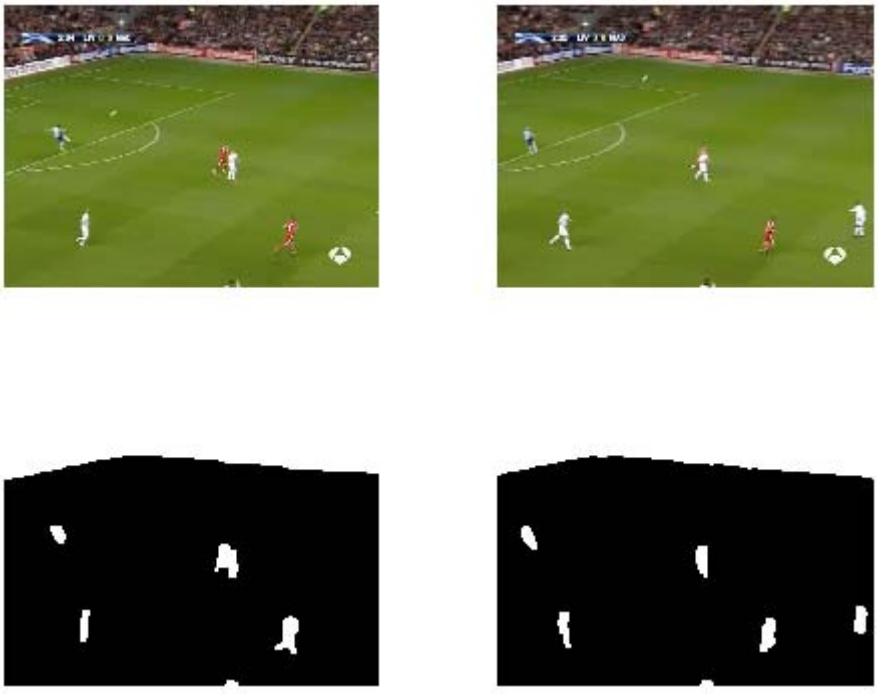


Imagen 55 - Ejemplo de unión de jugadores

En el centro de la imagen vemos como dos jugadores (uno de cada equipo) aparecen juntos por lo que el algoritmo de segmentación no logra identificar a ambos. Por lo tanto, a la hora de realizar el seguimiento se ve representado por un solo jugador en lugar de por dos.

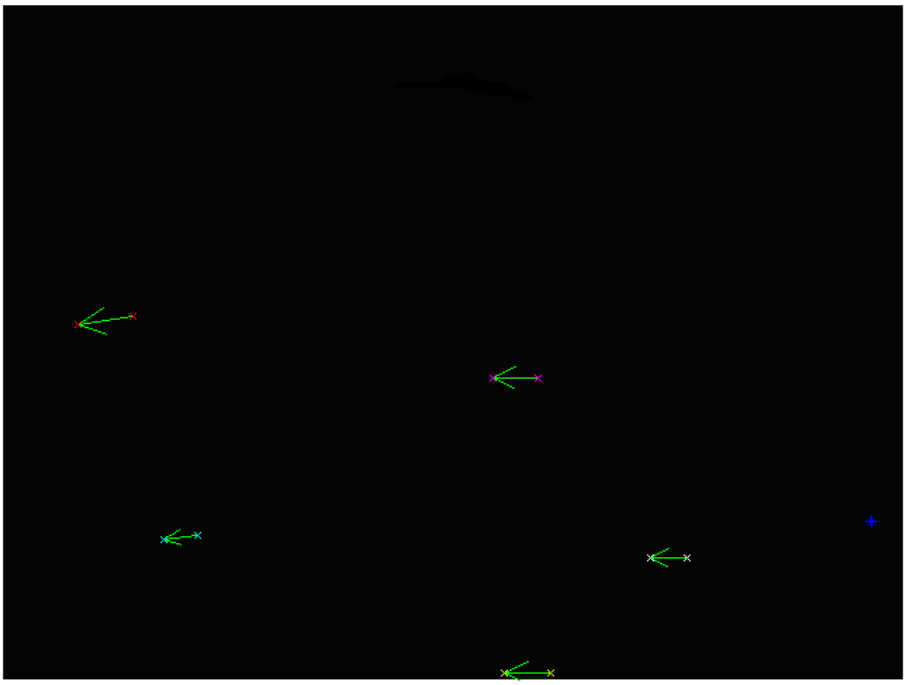


Imagen 56 - Seguimiento con error de unión

La unión de ambos jugadores es representada por el punto morado. El vector de movimiento detectado es correcto, pero lo correcto sería haber detectado dos movimientos en lugar de uno. Como se ha mencionado antes, la única forma de haber detectado ambos movimientos sería realizando la distinción por color.

En nuestro algoritmo, cuando los jugadores se separan es detectado como un jugador nuevo que entra en la escena, asignándole por tanto un nuevo símbolo y realizando el seguimiento posterior de forma habitual.

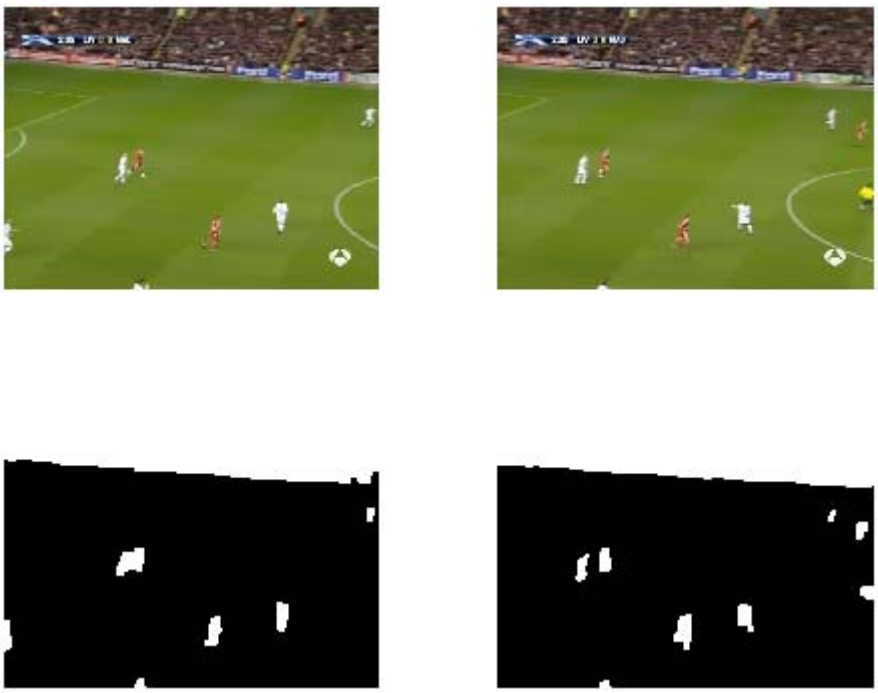


Imagen 57 - Ejemplo de 'desunión' de jugadores

Podemos ver como en el paso entre el primer frame y el segundo el algoritmo de segmentación ya ha diferenciado a dos conjuntos por lo que el posterior procesado identificará ambos jugadores.

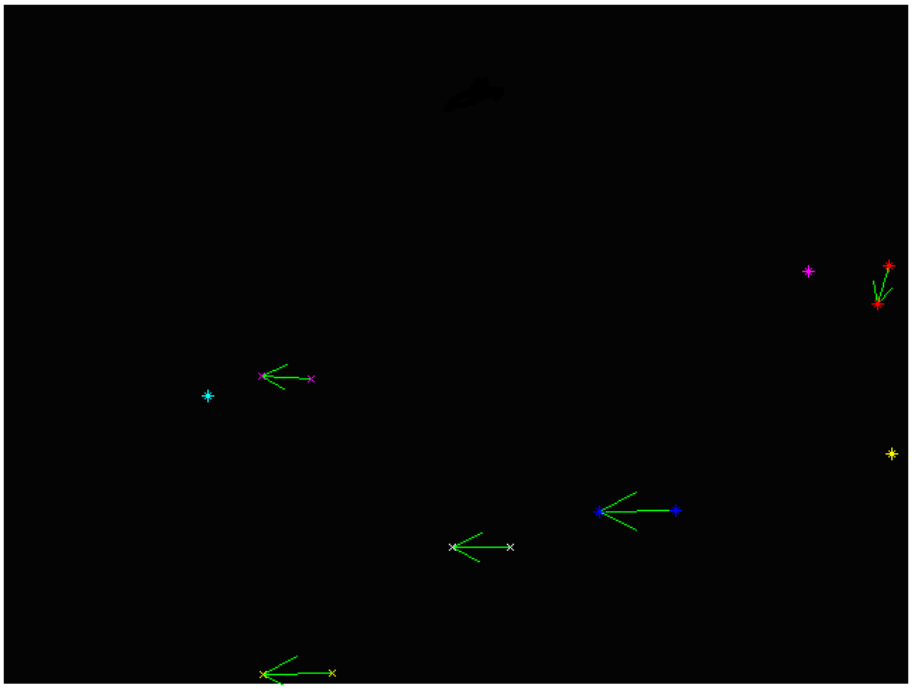


Imagen 58 - Detección de nuevo jugador en 'desunión'

En la detección podemos ver como el símbolo morado ha realizado el seguimiento y ha aparecido un nuevo símbolo azul claro que representa al jugador que se ha desvinculado de la unión. Por defecto, el jugador que aparece nuevo será el que más lejano esté al punto sobre el que se estaba realizando el seguimiento conjunto. A partir de entonces, este nuevo punto tendrá el seguimiento habitual como el resto.

4.7.4.2 ¿Qué ocurre si un jugador sale de la imagen y vuelve a entrar?

Este es otro de los problemas derivados del tipo de imágenes usadas y que, aparentemente, no tienen una fácil solución. Como se mencionó en la introducción, nuestro algoritmo funciona con las imágenes obtenidas de la propia televisión. Esto hace que solo tengamos esta imagen para procesar los movimientos y esta imagen siempre va a ser una imagen parcial del campo, nunca total.

El problema que tratamos en este apartado consiste en la salida de un jugador y su posterior entrada en la imagen bien por el movimiento de la cámara o bien por su propio movimiento. Si estamos siguiendo el movimiento de un jugador y este desaparece de la imagen, en su posterior entrada en la imagen pasados unos segundos somos incapaces de asignarle el mismo símbolo y color, por lo que le identificamos como un jugador nuevo y el seguimiento se pierde.

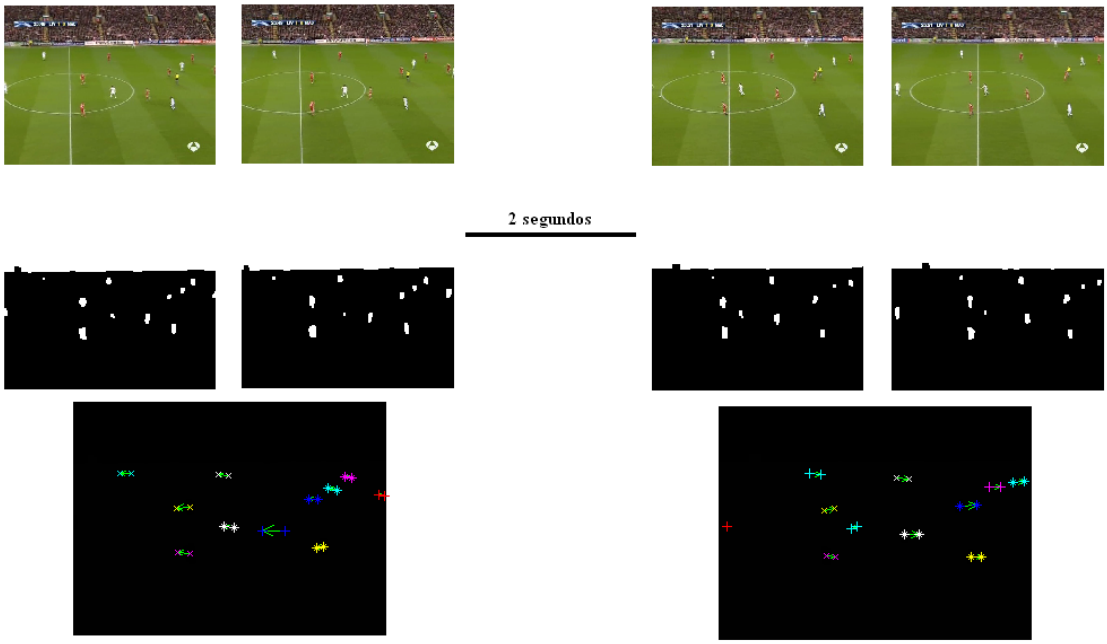


Imagen 59 - Ejemplo salida y entrada de jugador

En el ejemplo podemos ver lo explicado en el párrafo anterior:

En los dos primeros frames el jugador de la izquierda sale de la imagen debido al movimiento de la cámara hacia la derecha. Este jugador era identificado por una cruz de color rojo. Pasados dos segundos, la imagen vuelve hacia ese jugador ya que el balón vuelve a estar dirigido hacia esa zona. En esa identificación, el jugador aparece representado por el símbolo '+' de color rojo y esto indica que se trata de un jugador nuevo y no el que había salido dos segundos antes.

Para este problema no encontramos solución aparente ya que no es un problema que se solucione con un mejor desarrollo, es un problema que viene derivado del tipo de vídeos que estamos tratando en este proyecto. Para poder realizar este proceso sin cometer errores sería necesario obtener una visión total del campo y este no es el caso tratado en este proyecto. Además, si tuviéramos una visión completa este tipo de error no se contemplaría, ya que los 22 jugadores sobre el campo estarían visibles. Por lo tanto, esta complicación no es un problema en sí, es un fallo que aceptamos que vendría derivado del tipo de implementación que estamos realizando. Aún así, si lo identificáramos como el mismo jugador no nos serviría de ayuda ya que perderíamos el seguimiento igualmente al desconocer el movimiento del jugador mientras ha estado fuera de la imagen.

4.7.4.3 Vector de movimiento de la cámara

Este problema es el más importante de este apartado de cálculo. Como se ha mencionado en el algoritmo, al realizar el seguimiento de los jugadores sobre el propio frame podemos calcular dos tipos de movimientos:

- Movimiento absoluto, que es el que hemos explicado en el algoritmo y que calcula el movimiento de cada jugador según la fórmula:

$$x(1) = x(0) + mj \Rightarrow mj = x(1) - x(0)$$

En ella, solo tenemos en cuenta la posición actual del jugador y la posición anterior. Sin embargo, tras realizar pruebas con esta fórmula vimos que el algoritmo era un poco inexacto y se debe al tipo de imágenes que procesamos. Al no encontrarse la cámara fija, podemos ver que el propio movimiento de la cámara se refleja en el movimiento de los jugadores. En la mayoría de los casos influye en la mal interpretación de los movimientos ya que, como podemos ver en la siguiente imagen, podría detectar el movimiento hacia un lado en lugar de hacia el otro.

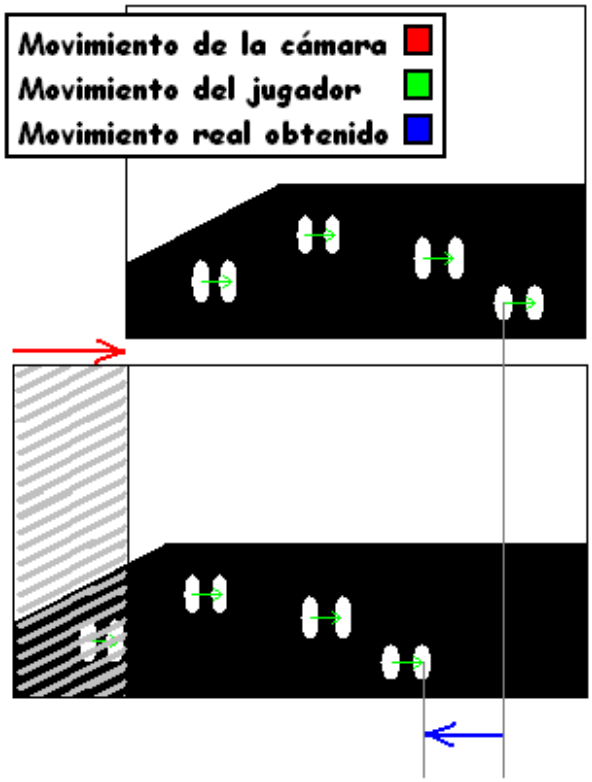


Imagen 60 - Error movimiento absoluto

En este sencillo ejemplo vemos como el movimiento de la cámara puede hacer que detectemos un movimiento del jugador hacia detrás y bastante grande cuando en realidad el movimiento ha sido hacia la derecha y mucho menor.

Si nos fijamos en los vectores de colores mostrados en el ejemplo, podemos ver que el movimiento obtenido del jugador es la diferencia entre el movimiento de la cámara (mc) y el movimiento real del jugador (mrj). Si a esta fórmula le aplicamos la utilizada anteriormente para el cálculo del movimiento podemos llegar a:

$$mj = x(1) - x(0) = mrj - mc \Rightarrow mrj = x(1) - x(0) \pm mc$$

Ecuación 11 - Cálculo de movimiento teniendo en cuenta la cámara

Hemos utilizado el signo ± ya que el movimiento de la cámara puede ser hacia derecha e izquierda. Independientemente de esto, lo que queda claro es que al calcular el movimiento real del jugador hay que tener en cuenta el movimiento de la cámara. Este movimiento de la cámara no se puede calcular de forma

inmediata por lo que calcularemos un valor de compromiso aceptable que podamos usar en este movimiento relativo y que será explicado en el siguiente punto.

- Movimiento relativo, en el que tendremos en cuenta de alguna forma el movimiento propio de la cámara.

  Como acabamos de explicar en el punto anterior debemos tener en cuenta el movimiento de la cámara para poder calcular un movimiento más exacto de los jugadores. Está fórmula nos indica que el movimiento real de un jugador va a componerse de la diferencia entre la posición absoluta en el frame 1 y la posición absoluta en el frame 0 unido a la suma (o diferencia) del movimiento de la cámara.

  Llegados a este punto nos encontramos con el mayor problema de esta fórmula que es decidir que valor tiene el movimiento de la cámara. Este movimiento depende en exclusiva del movimiento de la cámara entre el frame actual y el anterior, de forma que para poder corregir el movimiento de cada jugador tengamos en cuenta el movimiento de la cámara en ese instante.

  Como primera idea se piensa en obtener un punto fijo y fácilmente identificable de la imagen de forma que podamos calcular el movimiento de ese punto de un frame a otro. Como primer punto de partida se piensa en el corner, que sería fácilmente identificable, pero esta zona del campo no aparece en todos los casos. Otro tipo de punto podría ser un cartel pero el cartel se podría repetir en otra zona del campo y es bastante más difícil de identificar que el punto anterior.

  Debido a la dificultad de realizar este procedimiento se piensa en un valor aproximado que nos sea válido. Este valor es el valor medio de todos los movimientos de los jugadores en el instante anterior. El movimiento de la cámara es uniforme en cada frame de forma que les afectará a todos los jugadores por igual. Si obtenemos el valor medio de todos estos movimientos podremos calcular aproximadamente un valor medio del movimiento de la cámara. Es decir, el movimiento real de un jugador en el instante 2 será la diferencia de coordenadas entre los frames en ese instante más (o menos) el movimiento de la cámara en el instante anterior.

$$\boxed{mrj_2 = x_2(1) - x_2(0) \pm mc_1}$$

Ecuación 12 - Ecuación final de movimiento

  Este valor medio del movimiento de todos los jugadores solo será calculado sobre aquellos jugadores que tengan posiciones en ambos frames, los jugadores que salen o entran en la escena no serán tenidos en cuenta.

4.7.4.4 Jugador más cercano que su propio movimiento

Como se ha explicado anteriormente, la detección del movimiento de cada jugador viene determinada por la distancia a la que se encuentren los puntos más cercanos. Esta

es, a priori, la mejor forma de realizar este seguimiento dado el intervalo de tiempo ofrecido ya que los jugadores no tendrán movimientos muy grandes en este periodo.

Debido a esta metodología, nos encontramos con un problema en el seguimiento de los jugadores: ¿qué ocurre si aparece un nuevo jugador que en el siguiente frame está más cerca que su propio movimiento? Es decir, estamos realizando un seguimiento sobre un determinado jugador y en el frame siguiente otro jugador se pone más cerca de este que el propio movimiento del jugador que estamos siguiendo.

Este caso podemos verlo en unas imágenes ya mostradas. Si observamos la imagen de ejemplo de la 'desunión', en la esquina superior derecha podemos ver en el frame 1 aparece un jugador blanco y en el frame 2 el mismo jugador blanco y un jugador rojo. Si observamos la detección obtenida estamos cometiendo un error ya que el punto rojo debería haber asignado el movimiento al punto morado y no es así ya que el otro punto rojo (que equivale al nuevo jugador) está más cerca que el verdadero movimiento (que sería el punto morado).

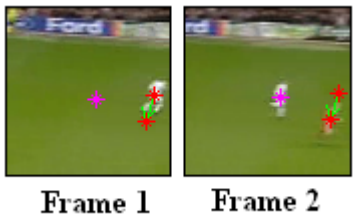
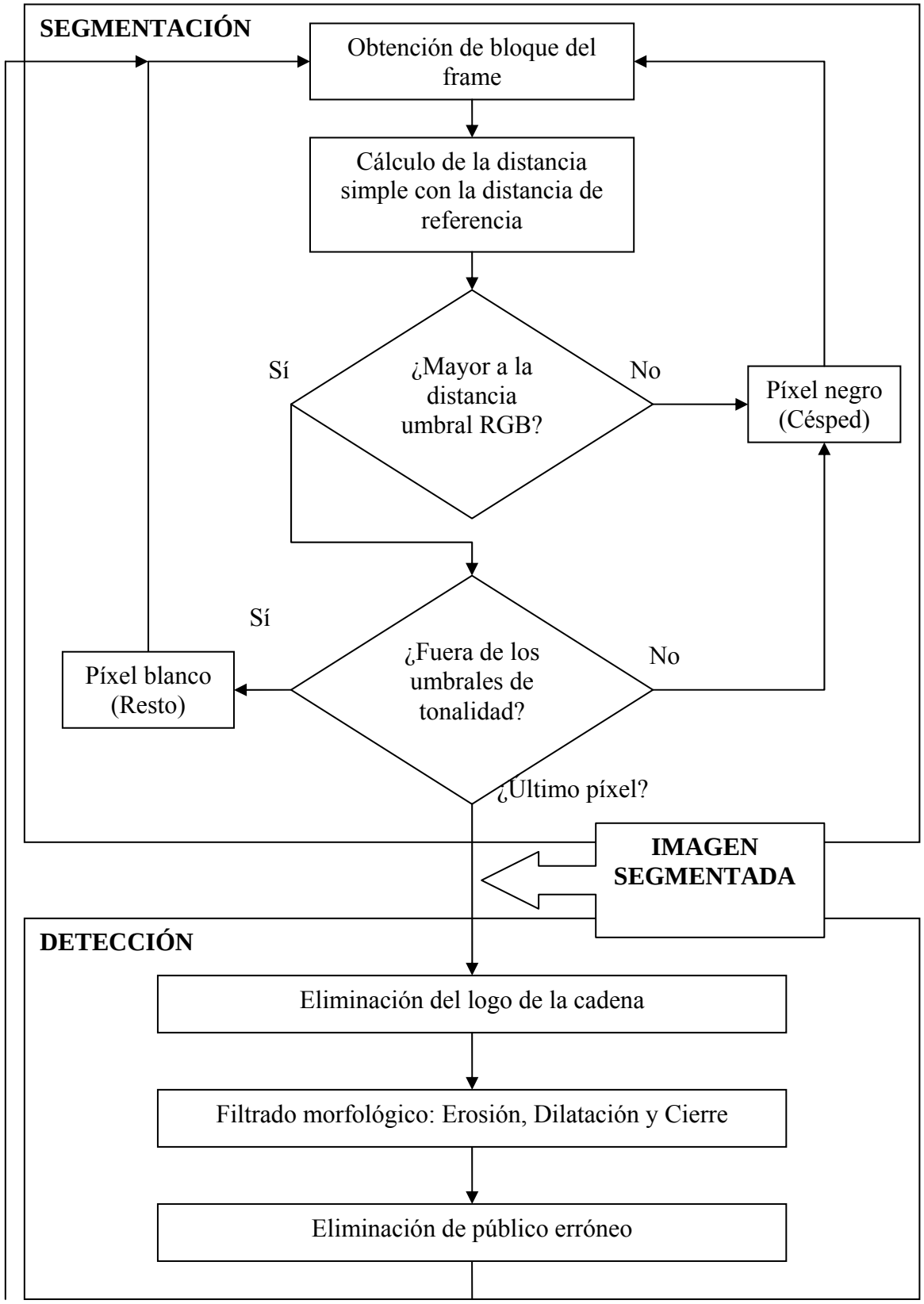


Imagen 61 - Error cometido por el uso de distancia mínima

Este problema es bastante difícil de eliminar porque mediante nuestro método desconocemos el movimiento de cada jugador, de forma que los casos como el anteriormente citado nos darán lugar a error. Una forma de paliar este problema sería por la detección por colores de los equipos que hemos comentado para eliminar otros problemas. En este caso del ejemplo, esta variación sí sería válida pero en un hipotético caso en el que los jugadores fueran del mismo equipo seguiríamos teniendo este error.

4.8 Esquema final del algoritmo

El esquema final del algoritmo sería el siguiente:



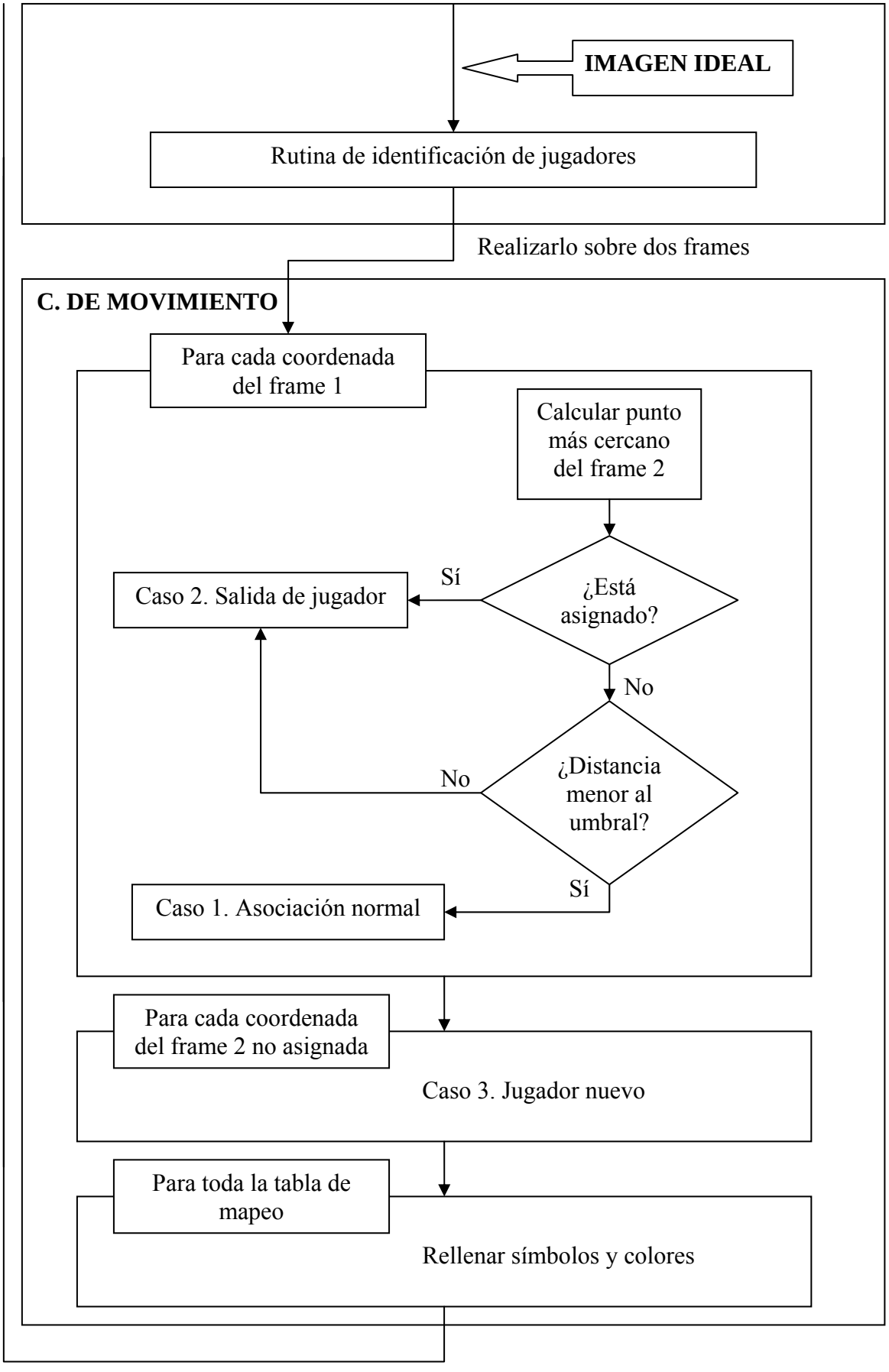


Imagen 62 - Esquema global del algoritmo

4.9 Conclusiones

Durante este tema nos hemos centrado en la explicación del algoritmo de cálculo de movimiento que se ha desarrollado en este proyecto. Como se ha podido comprobar, el algoritmo consta básicamente de 3 apartados bien diferenciados:

- Segmentación, donde realizamos la diferenciación de las distintas partes de la imagen.
- Detección, al conseguir situar cada uno de los jugadores en una coordenada de cada frame.
- Cálculo de movimiento, mediante las coordenadas anteriores y la utilización de dos frames consecutivos.

Estos tres apartados, como hemos comentado, se encuentran bien diferenciados aunque son dependientes entre ellos. Por ejemplo, un error en el algoritmo de segmentación tendría como consecuencia un error en la detección y por lo tanto, un error en el cálculo de movimiento. Esta dependencia la podremos comprobar en el siguiente apartado de resultados experimentales.

Para la realización del algoritmo de **segmentación** se ha necesitado un tipo de filtrado adicional al esperado. En la mayoría de los estudios un filtrado por color es lo más utilizado para realizar esta segmentación. Sin embargo, debido a las necesidades encontradas durante las fases de pruebas nos vemos obligados a realizar un filtrado adicional por tonalidad. También ha sido necesario un cambio en el tipo de algoritmo ya que una segmentación píxel a píxel necesitaba mucho más tiempo que la segmentación bloque sin apenas ganancia de calidad apreciable. Estos han sido los cambios más notables respecto al algoritmo inicial. Decir también en este apartado que el tiempo empleado sobre ambos algoritmos mostrado es el obtenido con el ordenador portátil utilizado. El procesador de este ordenador es bastante antiguo y por lo tanto bastante lento comparado con los procesadores actuales. Si se utilizara un ordenador actual con toda seguridad estos tiempos se reducirían drásticamente y por lo tanto nuestro algoritmo mejoraría en calidad temporal.

En la apartado de la **detección** de los distintos jugadores encontrados ha sido necesario el apoyo de la rutina de matlab de detección de bloques sobre fondo negro ya que nos servía de gran ayuda con la técnica utilizada. También se han utilizado diferentes filtros morfológicos para la mejora de la segmentación y eliminación de posibles errores de esta. El resto de modificaciones realizadas sobre el algoritmo inicial se han logrado en la fase de pruebas al lograr ver los fallos puntuales de cada tipo de partido. Por estas modificaciones nos referimos a la eliminación del logo (dependiente de cada retransmisión), el uso de cierre como filtro morfológico o la eliminación del público erróneo. Los errores que se cometen en este apartado son de solución compleja y fuera de nuestro proyecto por lo que, obviando este tipo de errores, el funcionamiento del algoritmo es prácticamente ideal.

Para la tercera parte, el algoritmo de **cálculo de movimiento** ha sido más complejo ya que la realización del algoritmo principal ha sido propia y existen numerosos casos a tener en cuenta. Además, se han llegado a contabilizar unos 17 jugadores en un mismo

frame lo que hace que el volumen de datos manejados sea muy alto. La resolución de este apartado se ha basado en la utilización de un mapeo de coordenadas de los distintos jugadores entre frames. Este mapeo se va a centrar en la obtención de las coordenadas más cercanas entre frames siendo esta, a priori, la solución más factible. Sin embargo, la detección del movimiento de los jugadores va a tener peculiaridades como la unión de jugadores, la unión a las bandas o la asignación incorrecta de coordenadas que habrá que tener en cuenta. Además, debido al tipo de sistema utilizado basado en la detección de movimiento sobre retransmisiones de televisión, se ha de tener en cuenta el movimiento propio de la cámara. Esto hace que el movimiento de los jugadores se detecte en función de este otro movimiento por lo que también ha de ser tenido en cuenta.

Durante la **fase de pruebas** se ha llegado a la conclusión de que, aparentemente, el sistema funciona de forma bastante aceptable. Se han detectado problemas con los que no se había contado y se han buscado soluciones en la medida de lo posible. Se han realizado pequeñas pruebas sobre algunos fragmentos de algunos partidos sin llevar al sistema al límite de funcionamiento y los resultados han sido positivos.

Durante el siguiente tema, el de **resultados experimentales**, se va a usar el sistema sobre varios fragmentos de varios partidos y llevando al límite (en algunos casos) al algoritmo. Este límite suele estar en el número de jugadores que aparecen en cada frame ya que hace que el mapeo de los jugadores se vuelva más inestable debido a la gran cantidad de coordenadas y a la cercanía entre ellas. También se realizarán pruebas sobre las distintas partes del algoritmo para saber qué partes funcionan mejor y cuáles peor y dónde se podría mejorar. La parte más complicada de esta fase va a estar en la forma en que se determina si los resultados son positivos o son negativos ya que debería ser algo objetivo pero, debido a la alta dificultad en dar unos resultados concretos de forma automática, esta paso va a necesitar una comprobación personal.

5. RESULTADOS EXPERIMENTALES

5.1 Introducción

En este apartado vamos a realizar pruebas del algoritmo sobre una serie de fragmentos de los vídeos indicados en la primera parte del proyecto. Estas pruebas a realizar van a consistir en determinar si la identificación de los jugadores es válida, si el seguimiento se hace de forma correcta sobre cada jugador, si la inclusión o desaparición de los jugadores en escena es correcta así como los diferentes casos problemáticos que hemos visto en los apartados anteriores.

Para comprobar el buen funcionamiento y robustez del sistema vamos a realizar las pruebas siguiendo este patrón:

- 5 vídeos distintos
- 2 fragmentos distintos por vídeo
- 2 segundos de duración por fragmento

Es decir, de cada vídeo vamos a elegir dos fragmentos aleatorios del partido (de dos segundos cada uno) en los que se pueda realizar el seguimiento y vamos a detallar los casos en los que funciona de manera correcta, los casos en los que falla y las causas de estos fallos. Después de cada vídeo tendremos detallado en forma de tabla los resultados obtenidos.

Las distintas pruebas a las que van a ser sometidos son las siguientes:

1.- Prueba de segmentación, en la que comprobaremos si la segmentación de los jugadores en el césped es correcta.

2.- Prueba de detección, en la que demostraremos que la detección de todos los jugadores en la escena es adecuada y conforme a lo esperado.

3.- Prueba de seguimiento, en la que probaremos si el seguimiento de cada jugador ha sido correcto y en el caso de que haya fallado se intentará buscar las causas. Tendremos especial cuidado en comprobar que los símbolos y colores son los adecuados para cada jugador.

4.- Tipos de errores detectados, en los que veremos para cada fragmento que errores se han detectado.

Como se ha mencionado anteriormente, el video utilizado en la mayoría de los casos para la fase de entrenamiento y pruebas ha sido el Liverpool – Real Madrid. Se eligió este partido debido a la alta calidad de la imagen y la buena respuesta a la segmentación. En las conclusiones finales podremos ver si la calidad del seguimiento se puede apreciar en función de que el video utilizado sea el de entrenamiento o de test.

Para la fase de entrenamiento y prueba del algoritmo se han utilizado fragmentos del partido aleatorios y en los que nos enfrentáramos a los problemas habituales encontrados como pueden ser la unión de jugadores, la unión a las bandas o la pérdida de segmentación. Los fragmentos usados en la fase de resultados de este partido han

sido los mismos que los usados para mostrar un algún tipo de error concreto en los capítulos anteriores. En este caso, un frame del fragmento uno se usó para mostrar la aparición de un nuevo jugador (imagen 53) y un frame del fragmento dos se usó para mostrar un fallo en la asignación de movimiento (imagen 61) pero en ningún caso han sido probados durante todo el intervalo de tiempo. Para el resto de partidos se han usado partes aleatorias en las que hubiera variedad de movimientos y casos de error para poder evaluar mejor la respuesta del algoritmo.

Como sabemos, el algoritmo de segmentación lleva asociado una serie de ajustes previos para que su funcionamiento sea óptimo. Estos ajustes son:

- Umbral de distancia de color RGB
- Umbral de distancia de tonalidad
- Verde inicial de referencia.

La elección del verde de referencia se hace de forma manual eligiendo cualquier punto del verde del césped de cada partido. Se ha generado un código adicional que muestra una imagen del partido y puedes elegir con un puntero cualquiera de los puntos (de forma aleatoria). Ese píxel elegido será el verde de referencia usado para la comparación de distancias.

Tanto el umbral de distancia RGB como el umbral de tonalidad (hue) son en todos los casos muy parecidos. Se ha decidido calibrarlos en todos los casos de forma manual para poder obtener la mejor segmentación posible en cada caso y así poder centrar los resultados en el seguimiento de los jugadores. Esta calibración se ha realizado modificando ligeramente los umbrales en todos los casos para llegar a los mejores resultados de segmentación. Al comienzo de cada prueba se comentará cuales han sido los umbrales usados en cada caso.

Una vez explicados los pasos que vamos a seguir para la fase de resultados y las calibraciones previas realizadas antes de cada prueba vamos a intentar explicar, en la medida de lo posible, como se han evaluado los distintos casos. Este es, principalmente, el mayor problema encontrado en esta fase de resultados ya que esta fase debe ser lo más objetiva posible pero es bastante complicado conseguirlo. No se ha encontrado ninguna forma posible de automatizar la fase de resultados y obtener un porcentaje de acierto de forma objetiva. La forma en la que comprobamos los resultados obtenidos es totalmente manual, supervisando los movimientos obtenidos entre cada frame y comprobando con los frames reales que ese movimiento es el correcto. Respecto de la segmentación y detección de los jugadores, puede ser un poco más objetiva pero necesita también una supervisión para comprobar que los jugadores segmentados y detectados son correctos.

La forma en la que se presentan los resultados creemos que ha sido la más conveniente para este caso. Tenemos dos tablas de resultados:

- **Tabla de resultados**, en la que indicamos:
    - Frames sobre los que se realiza la prueba
    - Número de jugadores en cada frame
    - Seguimiento correcto y erróneo en cada caso

- Salidas y entradas de jugadores
- Porcentaje de acierto

- **Tabla de notas**, en la que indicamos:

  - Fallo y/o recuperación de una segmentación errónea.
  - Fallo y/o recuperación de un error de doble conjunto.
  - Fallo y/o recuperación de un error por unión de jugadores.
  - Unión de algún jugador a banda.

En todos los casos indicados decidir si un seguimiento es correcto o erróneo o decidir si ha habido un fallo o recuperación de algún tipo de error es propia, en ningún caso, el algoritmo decide si es correcto o erróneo.

Se ha elegido esta metodología de presentación de resultados debido a que no se ha encontrado ningún tipo de estudio similar en el que se pudiera obtener el software ni la metodología usada por ellos para calcular los resultados de una forma objetiva y de esta forma poder comparar resultados. Sin embargo, a la hora de las conclusiones volveremos a abordar este tema y se intentará comparar en la medida de lo posible nuestro algoritmo con los ya realizados hasta la fecha.

5.2 Casos de prueba

Los vídeos sobre los que vamos a realizar las pruebas son los siguientes:

Caso 1 – Partido Liverpool – Real Madrid
Caso 2 – Partido España-Rusia
Caso 3 – Partido Almería - Valencia
Caso 4 – Partido Atlético de Madrid - Villareal
Caso 5 – Partido Barcelona – Lyon

5.2.1 Partido Liverpool - Real Madrid

Este partido es el partido que hemos utilizado de base a la hora de realizar todos los ejemplos en el proyecto. En un equipo tenemos jugadores de color rojo y en el otro equipo los tenemos de color blanco. En principio no deberíamos tener problemas en la segmentación e identificación. El umbral usado para el cálculo de la distancia es 0.2 y el umbral HUE es 0.12

5.2.1.1 Fragmento 1 – Del frame 87610 al 87660

1.- Prueba de segmentación

La segmentación es correcta para la mayoría de los casos. Al comenzar el fragmento en una esquina, tenemos una valla publicitaria de color verde que influye algo en la segmentación pero no afecta a los resultados. Encontramos un caso en el que un jugador no es detectado por la segmentación en varias ocasiones:

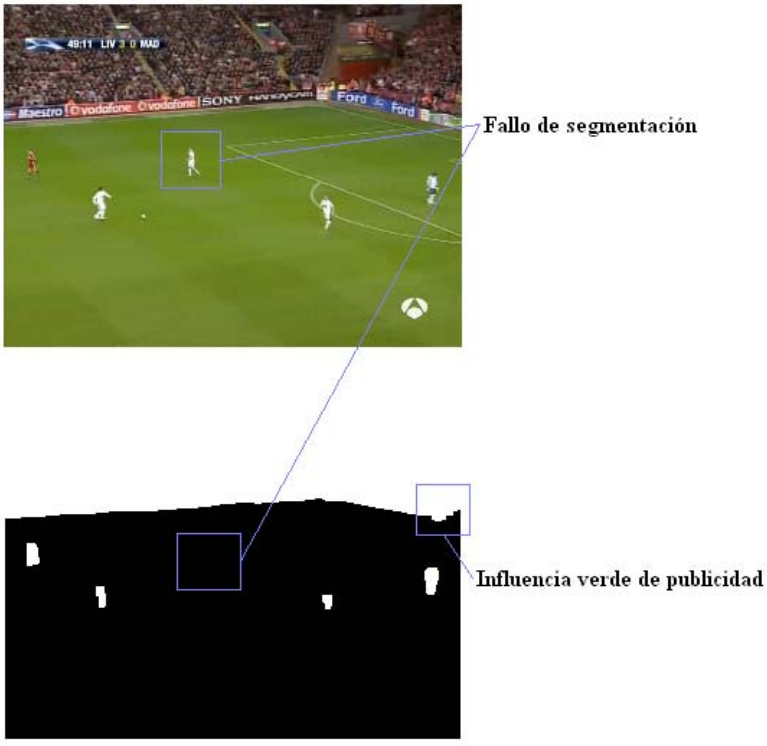


Imagen 63 - Frames 87615-87620 del Liverpool - Real Madrid

2.- Prueba de detección

La prueba de detección de jugadores ha sido correcta en todos los casos. Además el funcionamiento ha sido óptimo ya que se han detectado jugadores en casos extremos como puede ser en el borde de la escena. Solo ha fallado como consecuencia del fallo cometido en la segmentación.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
87610	87615	3	4	3	0	1	0	100,0
87615	87620	4	5	4	0	1	0	100,0
87620	87625	5	5	4	1	0	0	80,0
87625	87630	5	5	5	0	0	0	100,0
87630	87635	5	5	5	0	1	1	100,0
87635	87640	5	5	5	0	0	0	100,0
87640	87645	5	5	3	2	0	0	60,0
87645	87650	5	6	3	2	1	0	66,7
87650	87655	6	6	5	1	0	0	83,3
87655	87660	6	6	5	1	0	0	83,3
							MEDIA	**87,3**

Tabla 1 - Porcentajes fragmento 1 del Liverpool – R.Madrid

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
-	-	-	-	-	-	-
-	-	-	-	-	-	-
SÍ (1)	-	-	-	-	-	-
-	SÍ (1)	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
SÍ (1)	-	SÍ (1)	-	-	-	-
-	NO (1)	-	SÍ (1)	SÍ (1)	-	-
-	SÍ (1)	-	-	SÍ (1)	-	-
-	-	-	-	SÍ (1)	-	-

Tabla 2 - Notas de errores del fragmento 1 del Liverpool – R.Madrid

El porcentaje medio de aciertos en este fragmento es de 87.33% que es un porcentaje bastante bueno. La mayor parte de los errores de seguimiento se han producido por errores conocidos y aceptados por lo que este porcentaje se acerca a un valor óptimo de seguimiento.

4.- Errores detectados

En este fragmento nos hemos encontrado con tres tipos de errores. El primero de ellos es un error de segmentación en el que por alguna causa, no se identifica a un determinado jugador en la segmentación y hace que el seguimiento se paralice hasta que volvemos a retomarlo. En este caso el error de segmentación se produce dos veces en

los dos segundos que dura el fragmento y en ambos casos sobre el mismo jugador. En las dos ocasiones coincide que el jugador se encuentra en una posición en la que hay una menor proporción del color del traje que en la proporción normal, es decir, vemos "menos" color que el habitual. Es por esto por lo que la segmentación falla y hace que no se detecte ningún conjunto sobre el césped. El segundo de ellos se trata de un error de detección de dos conjuntos sobre un mismo jugador y que se explicó anteriormente en el uso del cierre como filtro morfológico. Y como tercer error nos encontramos con la unión de varios jugadores y detección como un solo conjunto de la que también hemos hablado con anterioridad. En la tabla de seguimiento podremos comprobar como para los dos primeros errores el algoritmo es inteligente y recupera el seguimiento al siguiente frame (aunque lo identifique como un jugador distinto al que era) por lo que el funcionamiento es más que aceptable. Para el tercer tipo de error el fragmento no llega a la separación de ambos jugadores por lo que su evaluación no es posible.

5.2.1.2 Fragmento 2 – Del frame 11000 al 11050

1.- Prueba de segmentación

En este caso la segmentación ha sido óptima salvo por un fallo en el frame 11045 de obtención de más de un conjunto en un jugador.

Imagen 64 - Frame 11045 del Liverpool - Real Madrid

Sin embargo, en el siguiente frame la segmentación es buena y se produce una recuperación de este error.

2.- Prueba de detección

Al igual que en el fragmento anterior, la detección de jugadores ha sido óptima. Se han detectado todos los jugadores salvo en los casos de unión de varios de ellos.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
11000	11005	6	6	6	0	0	0	100,0
11005	11010	6	6	6	0	0	0	100,0
11010	11015	6	6	6	0	0	0	100,0
11015	11020	6	6	5	1	0	0	83,3
11020	11025	6	7	5	1	1	0	85,7
11025	11030	7	7	6	1	0	0	85,7
11030	11035	7	6	5	1	0	1	85,7
11035	11040	6	7	5	1	1	0	85,7
11040	11045	7	8	4	1	2	1	87,5
11045	11050	8	8	6	2	0	0	75,0

MEDIA	88,9

Tabla 3 - Porcentajes fragmento 2 del Liverpool – R.Madrid

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	SÍ (1)	-	-
-	-	-	-	SÍ (1)	-	-
-	-	-	-	SÍ (1)	-	-
-	-	-	-	SÍ (1)	-	-
-	-	-	-	SÍ (1)	-	-
-	-	SÍ (1)	-	-	SÍ (1)	-
-	-	-	SÍ (1)	-	-	-

Tabla 4 - Notas de errores del fragmento 2 del Liverpool – R.Madrid

El seguimiento de este apartado ha sido satisfactorio, llegando incluso a recuperarse de la unión de dos jugadores durante 6 de los 8 intervalos. El porcentaje medio de acierto es del 88.86% que es un porcentaje muy alto.

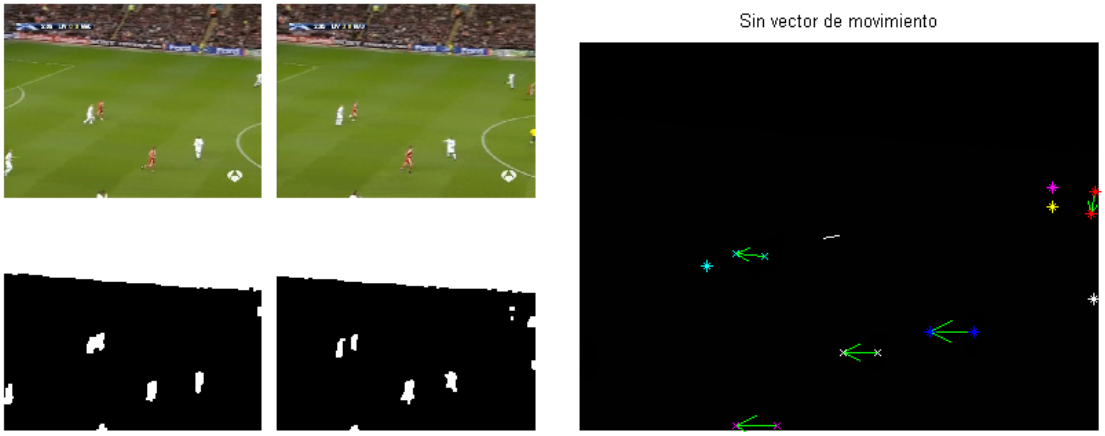


Imagen 65 - Desunión jugadores

En la imagen podemos ver como se produce la desunión de los jugadores en el intervalo de frames y como en la detección el seguimiento de uno de ellos continúa (cruz azul clara) y aparece un nuevo jugador que comenzará el seguimiento en ese punto (estrella azul clara).

4.- Errores detectados

A pesar del buen funcionamiento de los puntos anteriores, podemos ver otro tipo de problema del que también hemos hablado y que consiste en la detección errónea debido a que hay un jugador más cercano que su propio movimiento. Como apreciamos a la derecha de la imagen, el punto rojo superior corresponde al frame 1 mientras que el rojo inferior y los otros dos corresponden al frame 2 (nuevo jugador y fallo por detección doble). Al utilizar un algoritmo que usa la distancia mínima entre puntos, la asignación que realiza es la obtenida pero no es válida, ya que debería asignar este punto rojo superior al punto morado. La salida debería ser esta:

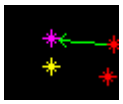


Imagen 66 - Seguimiento correcto ideal

Sin embargo, esta asignación no es posible por lo comentado en el apartado en el que se explicaba este tipo de fallo. Varios frames después este seguimiento se recupera ya que la regla de "el movimiento del jugador es el más cercano" vuelve a cumplirse.

5.2.2 Partido España - Rusia

Este partido tiene la característica de que los colores de las camisetas también son rojas contra blancas por lo que la segmentación no debería de darnos demasiados problemas. En comparación con el caso anterior, este vídeo tiene un césped más vivo de forma que podremos probar el algoritmo con otro tipo de verde. El umbral de cálculo de distancia en este caso es de 0.15 y el umbral HUE de 0.3

5.2.2.1 Fragmento 1 – Del frame 50000 al 50050

Este fragmento consiste en un movimiento normal de los jugadores en el centro del campo. Como algo peculiar de este fragmento vamos a ver como se produce una unión a la banda de uno o dos jugadores de forma constante.

1.- Prueba de segmentación

A pesar de tener que realizar la segmentación sobre un césped de mayor claridad el algoritmo ha respondido de forma satisfactoria al aumentarle el umbral HUE con respecto al caso anterior. Sin embargo, hemos tenido un fallo de segmentación que ha tardado bastante en recuperar. Este fallo se ha debido a que en una zona determinada de la línea nos encontramos con una mancha que no es identificada como césped:

Imagen 67 - Fallo en segmentación de líneas de campo

Se puede comprobar a simple vista como hay una pequeña mancha sobre la línea blanca. Este fallo unido a la posterior dilatación hace que cometamos un error de segmentación. Como esta línea permanece durante toda la imagen hace que la recuperación de este fallo sea más lenta. Como otra posible solución se piensa en

ampliar el umbral del HUE pero podría afectar al resto de jugadores por lo que se descarta.

2.- Prueba de detección

La detección ha sido muy buena en la mayoría de los casos. Nos encontramos con el problema de la unión de los jugadores con la banda pero, como ya comentamos en su momento, este problema es insalvable.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
50000	50005	12	12	10	2	0	0	83,3
50005	50010	12	12	10	2	0	0	83,3
50010	50015	12	11	9	2	0	1	83,3
50015	50020	11	10	7	3	0	1	72,7
50020	50025	10	10	7	3	0	0	70,0
50025	50030	10	10	8	2	0	0	80,0
50030	50035	10	10	8	2	0	0	80,0
50035	50040	10	9	7	2	0	1	80,0
50040	50045	9	10	7	2	1	0	80,0
50045	50050	10	10	8	2	0	0	80,0
							MEDIA	**79,3**

Tabla 5 - Porcentajes fragmento 1 del España – Rusia

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
-	-	SÍ (1)	-	SÍ (1)	-	-
-	-	-	NO (1)	-	NO (1)	-
-	-	-	NO (1)	-	SÍ (1)	-
SÍ (1)	-	-	NO (1)	SÍ (1)	-	SÍ (1)
-	NO (1)	-	NO (1)	-	NO (1)	-
	NO (1)	-	SÍ (1)	-	SÍ (1)	
	NO (1)	SÍ (2)	-	-	-	
	NO (1)	SÍ (1)	SÍ (1)	-	-	
-	SÍ (1)	SÍ (2)	-	-	-	-
SÍ (1)	-	SÍ (1)	SÍ (1)	-	-	SÍ (1)

Tabla 6 - Notas de errores del fragmento 1 del España – Rusia

A pesar de encontrarnos con bastantes fallos previos (como puede ser que dos jugadores estén prácticamente todo el fragmento pegados a la banda) el seguimiento ha sido bastante bueno. Hemos tenido varios fallos de los que hemos recuperado sin problema (incluida una 'desunión' de dos jugadores) y los porcentajes de aciertos han sido muy buenos siendo el más bajo del 70% y el medio de 79.3%. No son tan buenos como en el partido anterior pero recordemos que este fue prácticamente el vídeo de entrenamiento del algoritmo.

4.- Errores detectados

Como error más destacable de este fragmento obtenemos la unión de los jugadores a la banda. Cuando describimos el error obtenido por este fallo, comentamos que era un error aceptado por nosotros y que la solución se planteaba bastante compleja. Además, en este caso, está unión a la banda se ha visto complementada por una unión de jugadores por lo que el fallo ha sido en cadena.

Imagen 68 - Error de unión a banda con unión de jugadores

En la imagen observamos como se ha producido una unión entre un jugador de un equipo con otro jugador del equipo contrario. Además, está unión se ha segmentado junto con las vallas y el público por lo que su detección es imposible.

Mencionar también el tema del árbitro. Nuestro algoritmo no diferencia por colores por lo que el árbitro, independientemente del color que vista, será identificado como un jugador más. En este caso, la vestimenta del árbitro no es la mejor de cara al buen funcionamiento ya que al ser el amarillo un color 'similar' (en cuestión de distancia) al verde realiza una segmentación doble por ello.

5.2.2.2 Fragmento 2 – Del frame 99475 al 99525

Este fragmento se caracteriza porque es una vista de la cámara oblicua de forma que los jugadores se encuentran en una posición que no es la habitual y en la que la cantidad de color que hay sobre el césped para realizar la segmentación es menor.

1.- Prueba de segmentación

La segmentación en este caso ha sido aceptable ya que hemos cometido dos fallos de segmentación de causa desconocida. En estos dos fallos el jugador no segmentado ha sido el mismo por lo que podemos pensar que el problema se encuentra en la posición del jugador y no en el algoritmo.

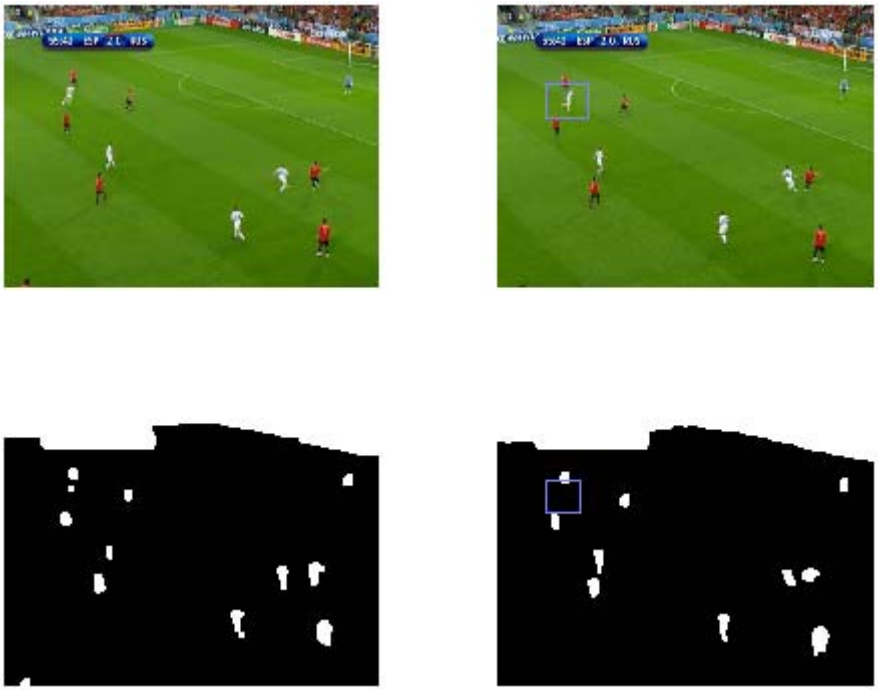


Imagen 69 – Error de segmentación en frames 99490 a 99495 del España – Rusia

Además, podemos comprobar como el marcador del resultado también es detectado como jugador debido a su alta luminosidad pero, en este caso, nos aprovechamos de la unión de este marcador con el público por lo que no se detecta.

2.- Prueba de detección

En este fragmento la detección ha sido muy buena, ya que no hemos cometido ningún fallo por detección errónea.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
99475	99480	12	12	12	0	0	0	100,0
99480	99485	12	12	12	0	0	0	100,0
99485	99490	12	12	12	0	0	0	100,0
99490	99495	12	11	9	2	0	1	83,3
99495	99500	11	11	10	1	0	0	90,9
99500	99505	11	11	9	2	0	0	81,8
99505	99510	11	11	9	2	0	0	81,8
99510	99515	11	11	10	1	0	0	90,9
99515	99520	11	11	11	0	0	0	100,0
99520	99525	11	11	10	1	0	0	90,9
							MEDIA	**92,0**

Tabla 7 - Porcentajes fragmento 2 del España – Rusia

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda

-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
SÍ (1)	-	-	-	SÍ (1)	-	-
-	SÍ (1)	SÍ (1)	-	SÍ (1)	SÍ (1)	-
SÍ (1)	-	-	SÍ (1)	-	NO (1)	-
-	NO (1)	-	-	-	NO (1)	-
-	SÍ (1)	-	-	SÍ (1)	SÍ (1)	-
-	-	-	-	-	SÍ (1)	-
-	-	SÍ (1)	-	SÍ (1)	-	-

Tabla 8 - Notas de errores del fragmento 2 del España – Rusia

Los resultados en este fragmento son francamente buenos ya que hemos obtenido un porcentaje de acierto de 92%. Este es, hasta ahora, el mejor resultado de los obtenidos. Además, este resultado nos dice que el algoritmo funciona bien para los casos en los que la escena se encuentra muy poblada de jugadores ya que en la mayoría de los frames había entre 11 y 12 jugadores que viene a ser la mitad de los jugadores que nos podemos encontrar en un partido de fútbol.

4.- Errores detectados

Como hemos comentado en el seguimiento, en este caso hemos cometido pocos errores. Cabe destacar que durante los 10 frames se han producido cuatro uniones de jugadores distintas (una de ellas para el mismo par de jugadores).

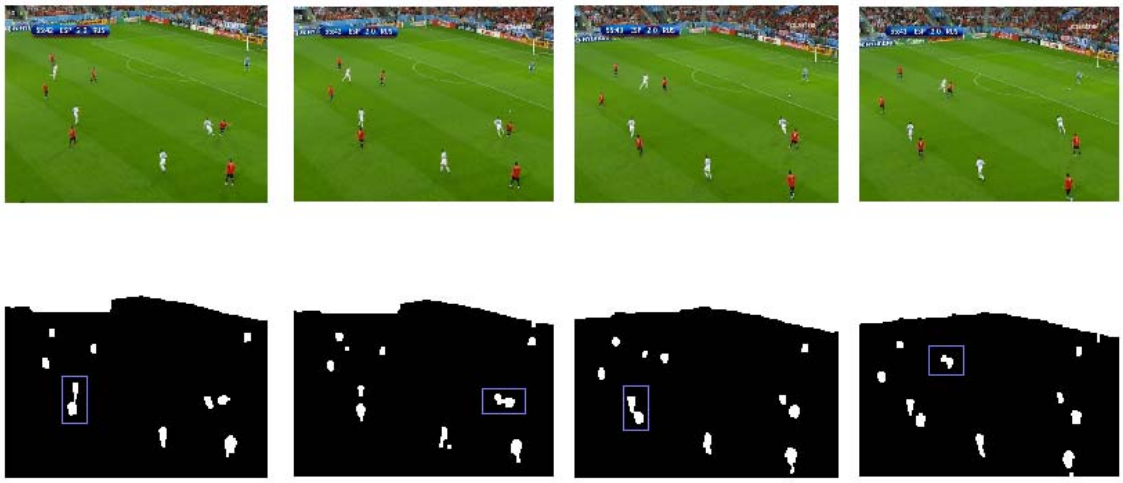


Imagen 70 - Distintos tipos de unión del España – Rusia

5.2.3 Partido Almería - Valencia

En este partido, antes de realizar ninguna prueba decir que nos esperamos resultados peores a los obtenidos en los dos casos anteriores ya que la calidad del vídeo en este caso es algo inferior. El umbral de cálculo de distancia en este caso es de 0.15 y el umbral HUE de 0.25

5.2.3.1 Fragmento 1 – Del frame 123075 al 123125

1.- Prueba de segmentación

En este fragmento hemos tenido bastantes fallos de segmentación por lo comentado anteriormente: la calidad del vídeo no es óptima y hace que el borde de los jugadores sea demasiado 'borroso' de forma que la detección de los jugadores es más complicada. Notamos esto en que la forma de los jugadores es más tosca comparada con los otros dos fragmentos en los que la forma detectada era similar a la posición del jugador. Esto hace que cometamos muchos fallos de detección doble de un jugador (cometemos 7 fallos en 10 frames).

2.- Prueba de detección

La detección en este caso es aceptable, aunque viene limitada por los errores que cometemos en la segmentación. También cometemos excesivos fallos de detección debido a la jugada observada en el fragmento: es una jugada rápida y el movimiento de la cámara es muy fluido lo que hace que cometamos bastantes errores de detección de jugadores más cercanos que el propio movimiento.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
10	10	8	2	0	0	10	10	80,0
10	11	7	2	1	0	10	11	80,0
11	10	6	3	1	2	11	10	75,0
10	9	9	0	0	1	10	9	100,0
9	10	7	3	1	0	9	10	72,7
10	10	5	5	2	2	10	10	64,3
10	8	5	3	0	2	10	8	70,0
8	8	7	1	0	0	8	8	88,9
8	9	7	1	1	0	8	9	88,9
9	9	8	1	0	0	9	9	88,9
							MEDIA	**80,9**

Tabla 9 - Porcentajes fragmento 1 del Almería - Valencia

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
-	-	SÍ (1)	-	SÍ (1)	-	SÍ (1)
SÍ (1)	-	SÍ (1)	SÍ (1)	-	NO (1)	SÍ (1)
-	NO (1)	SÍ (1)	SÍ (1)	-	SI (1)	-
	SÍ (1)	-	SÍ (1)	-	-	-
-	-	SÍ (2)	-	SÍ (1)	-	-
SÍ (2)	-	SÍ (1)	SÍ (2)	-	NO (1)	-
-	SÍ (2)	-	SÍ (1)	-	SÍ (1)	-
SÍ (1)	-	-	-	-	-	-
-	-	SÍ (1)	-	SÍ (1)	-	-

-	-	SÍ (1)	-	SÍ (1)	-	-

Tabla 10 - Notas de errores del fragmento 1 del Almería - Valencia

En este fragmento no hemos conseguido unos resultados muy aceptables. Como comentamos la principio, la calidad del video es inferior pero esto no debería influir de manera tan significativa. Hemos conseguido un porcentaje de acierto medio del 80.9% que, en comparación con los resultados anteriores, no es muy bueno.

Intentaremos ver como funciona el algoritmo sobre el otro fragmento de este video para comprobar que el tipo de jugada utilizada (con un alto movimiento de los jugadores) influye de manera negativa en nuestro algoritmo.

4.- Errores detectados

Los errores cometidos durante el fragmento no distan de los errores cometidos en los casos anteriores aunque en mayor proporción. Sin embargo, se detecta también un error que cometemos por el tipo de vista utilizada por la cámara: al utilizar una vista tan lejana, al acercarse al fondo se puede ver parte de la grada inferior y con la que nosotros no hemos contado en el entrenamiento del algoritmo:

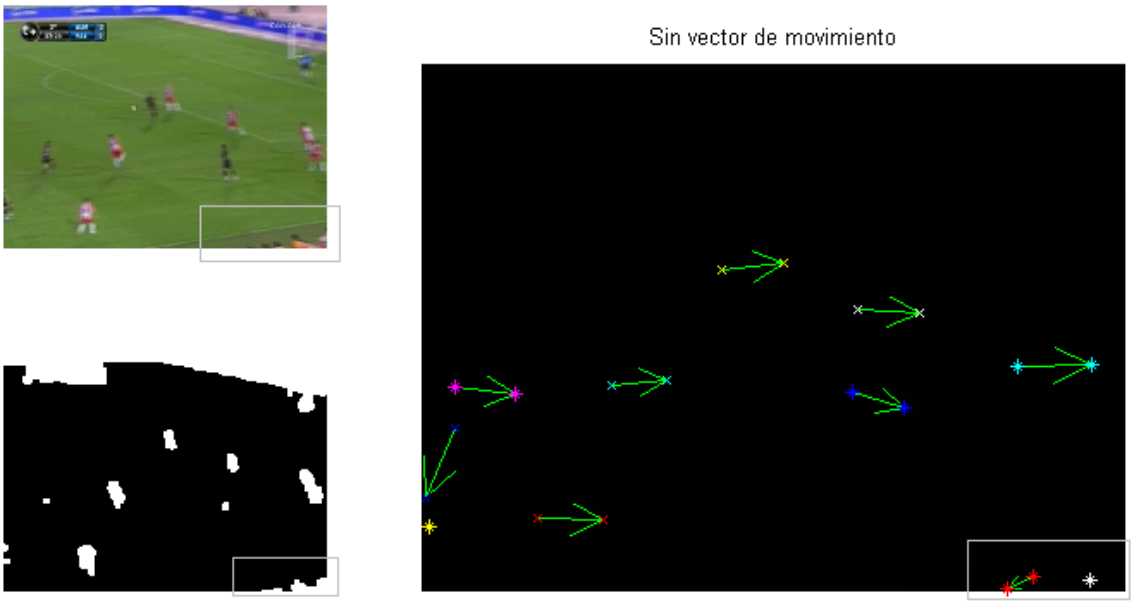


Imagen 71 - Error grada inferior del Almería – Valencia

En esta imagen podemos ver además como aparte de detectarlo, obtiene dos conjuntos lo que hace que detecte dos nuevos jugadores: uno que viene del frame anterior y tiene movimiento (punto rojo) y otro nuevo (punto blanco).

También podemos comprobar en la imagen del frame real como la calidad de la imagen es inferior, siendo bastante más borrosa que los otros dos casos anteriores.

5.2.3.2 Fragmento 2 – Del frame 134150 al 134200

1.- Prueba de segmentación

La segmentación ha sido bastante mejor que la anterior ya que los jugadores estaban bastante separados y el movimiento de la cámara no ha sido excesivamente rápido. Sin embargo, en el último frame vemos que se produce un despeje que hace mover la cámara hacia la derecha muy rápidamente. El frame resultante es muy borroso, lo que resulta en muchos fallos de segmentación:

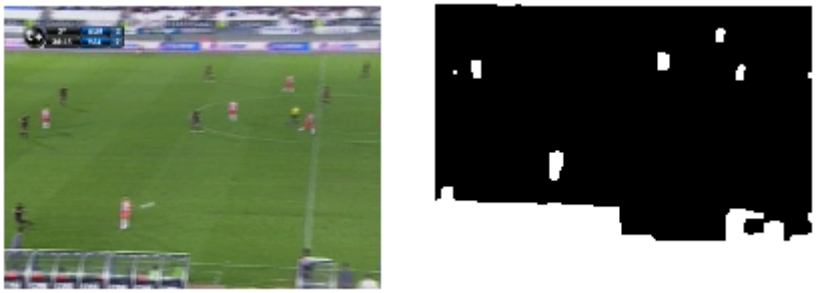


Imagen 72 - Error segmentación por rápido movimiento de cámara del Almería – Valencia

Podemos comprobar como de los 13 jugadores que debería detectar, solo detectamos 7 conjuntos siendo uno de ellos además no válido (el conjunto de la derecha).

2.- Prueba de detección

La detección de los conjuntos ha sido correcta en la mayoría de los casos. Como en el fragmento anterior, hemos cometido el error de detectar la parte inferior de la imagen (en este caso los banquillos) por lo que detecta más conjuntos de los que en realidad son.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
12	11	10	1	0	1	12	11	91,7
11	11	10	1	0	0	11	11	90,9
11	11	10	1	0	0	11	11	90,9
11	11	9	2	0	0	11	11	81,8
11	12	8	3	1	0	11	12	75,0
12	11	10	1	0	1	12	11	91,7
11	11	9	2	0	0	11	11	81,8
11	12	9	2	1	0	11	12	83,3
12	12	9	3	0	0	12	12	75,0
12	13	5	7	0	0	12	13	41,7
							MEDIA	**80,4**

Tabla 11 - Porcentajes fragmento 2 del Almería - Valencia

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
SÍ (1)	-	SÍ (1)	SÍ (1)	SÍ (1)	SÍ (1)	-
-	NO (1)	SÍ (1)	-	-	-	-

-	NO (1)	-	SÍ (1)	-	-	-
-	NO (1)	-	-	SÍ (1)	-	-
SÍ (1)	NO (1)	-	-	-	NO (1)	-
-	NO (1) - SÍ (1)	-	-	-	SÍ (1)	-
-	NO (1)	-	-	SÍ (1)	-	-
-	NO (1)	-	-	-	NO (1)	-
SÍ (1)	NO (1)	-	-	-	NO (1)	-
SÍ (4)	NO (2)	-	-	-	NO (1)	-

Tabla 12 - Notas de errores del fragmento 2 del Almería - Valencia

El seguimiento obtenido en este fragmento ha sido similar al anterior, obteniendo un porcentaje de acierto del 80.4% por lo que podemos deducir que tanto la calidad de la imagen como el movimiento rápido de la cámara influyen de manera negativa en nuestro algoritmo. Creemos que influye en mayor medida la calidad de la imagen ya que en los fragmentos anteriores también teníamos movimientos rápidos de cámara y los resultados no se degradaban de manera tan significativa.

Comentar también que el último intervalo influye mucho en el resultado final ya que, en ese intervalos en concreto, obtenemos un porcentaje de acierto del 41.7%. Este porcentaje es bajísimo comparado con el resto de porcentajes obtenidos. Si realizásemos la media con los nueve primeros intervalos obtendríamos una calidad del 84.7% más cercano a los casos anteriores.

4.- Errores detectados

Al igual que en el otro fragmento de este mismo partido, hemos cometido el error de detectar como conjuntos la parte inferior de la imagen que en este caso se trata de los banquillos de los equipos.

5.2.4 Partido Atlético de Madrid - Villareal

Los colores de cada equipo son amarillo y rojiblanco por lo que no deberíamos tener problemas en la segmentación. La calidad de la imagen en este caso es mejor que en el caso anterior por lo que la detección debería ser buena también. El umbral de cálculo de distancia en este caso es de 0.2 y el umbral HUE de 0.3

5.2.4.1 Fragmento 1 – Del frame 98150 al 98200

1.- Prueba de segmentación

La segmentación no es la esperada ya que hemos cometido más fallos de los esperados, sobre todo en las líneas blancas del campo. Como podemos comprobar en la siguiente imagen, hemos obtenido conjuntos en zonas donde debería haber funcionado la segmentación:

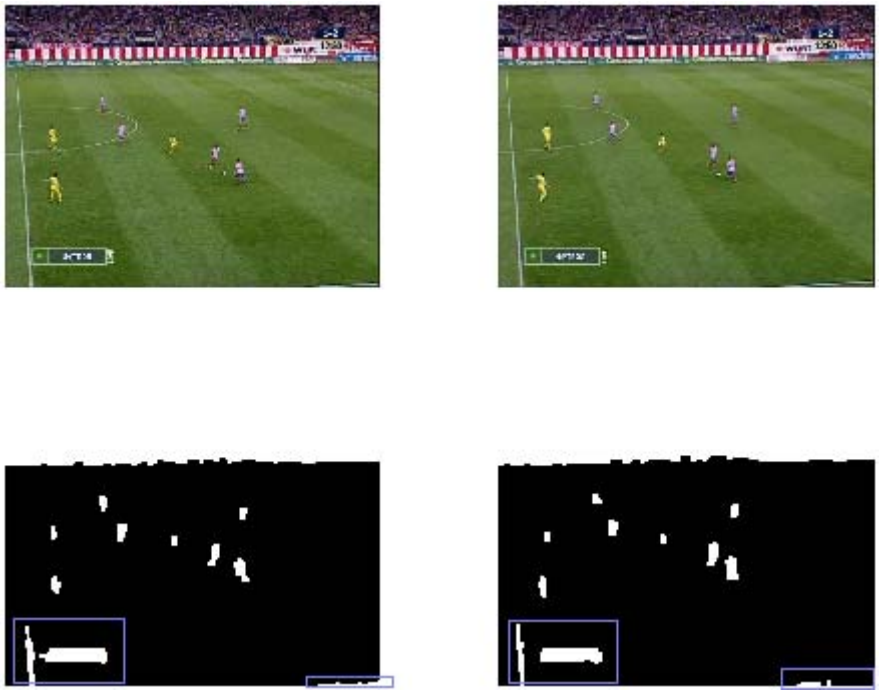


Imagen 73 - Error de segmentación en Atlético de Madrid - Villareal

El error que cometemos en estos casos es la segmentación y posterior detección de más conjuntos de los necesarios pero no implica una perdida de calidad en el seguimiento de los jugadores salvo en los casos en los que esos conjuntos se encuentren cercanos a algún jugador. Se ha querido mostrar en este caso como afectaría el dejar el logo de la cadena al seguimiento de los jugadores. En este caso la afectación es mínima, ya que el logo se encuentra muy lejano a los jugadores.

2.- Prueba de detección

En este caso, la detección ha sido muy buena ya que se detectan todos los conjuntos identificados en la segmentación (incluidos los conjuntos erróneos indicados en el punto anterior). Posteriormente, podremos ver como estas detecciones de errores de segmentación tendrán como consecuencias un seguimiento erróneo en algunos casos.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
8	8	8	0	0	0	8	8	100,0
8	8	7	1	0	0	8	8	87,5
8	8	7	1	0	0	8	8	87,5
8	8	6	2	0	0	8	8	75,0
8	8	6	2	0	0	8	8	75,0
8	8	7	1	0	0	8	8	87,5
8	8	8	0	0	0	8	8	100,0
8	8	8	0	0	0	8	8	100,0
8	8	7	1	0	0	8	8	87,5
8	8	8	0	0	0	8	8	100,0

MEDIA	90,0

Tabla 13 - Porcentajes fragmento 1 del Atlético de Madrid - Villareal

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
-	-	-	-	-	-	-
-	-	-	-	SÍ (1)	-	-
-	-	-	-	-	NO (1)	-
SÍ (1)	-	-	-	-	NO (1)	-
-	NO (1)	-	-	-	NO (1)	-
-	SÍ (1)	-	-	-	NO (1)	-
-	-	-	-	-	SÍ (1)	-
-	-	SÍ (1)	-	-	-	-
-	-	-	SÍ (1)	-	-	-
-	-	-	-	-	-	-

Tabla 14 - Notas de errores del fragmento 1 del Atlético de Madrid - Villareal

En este fragmento hemos obtenido un porcentaje de acierto del 90%. Este resultado es muy bueno, aunque debemos tener en cuenta que para este cálculo no hemos tenido en cuenta el seguimiento realizado sobre conjuntos que no son jugadores indicados en la imagen anterior. Por el contrario, sí que hemos tenido en cuenta el caso en el que algún jugador fallase por estos conjuntos. También hemos cometido algún error de seguimiento en algunas recuperaciones, como por ejemplo al recuperarse de un doble conjunto la asignación no ha sido la adecuada.

Imagen 74 - Error de seguimiento provocado por recuperación de doble conjunto

4.- Errores detectados

Como hemos mostrado en los puntos anteriores, los errores más destacados en este caso son varios:

- Detección de conjuntos erróneos, siendo líneas del campo y logo de cadena las partes de la imagen detectadas que no deberían haberlo sido.
- Una unión de jugadores, que se reproduce durante 5 intervalos pero del que conseguimos recuperarnos y continuar con el seguimiento.
- El fallo por doble conjunto, que es el mostrado anteriormente.

5.2.4.2 Fragmento 2 – Del frame 139300 al 139250

1.- Prueba de segmentación

En esta caso hemos cometido los mismos errores de segmentación que obteníamos en el fragmento anterior. Estos errores consisten en la detección de las líneas de campo como conjuntos. Se supone que modificando los umbrales de color y luminosidad debería poder eliminar estas detecciones pero se ha probado con numerosos valores no consiguiendo mejorar los resultados. Para el cálculo del seguimiento no serán tenidos en cuenta estos errores salvo que afecten a la detección de algún jugador. En la siguiente imagen vamos a ver un ejemplo de esto que comentamos:

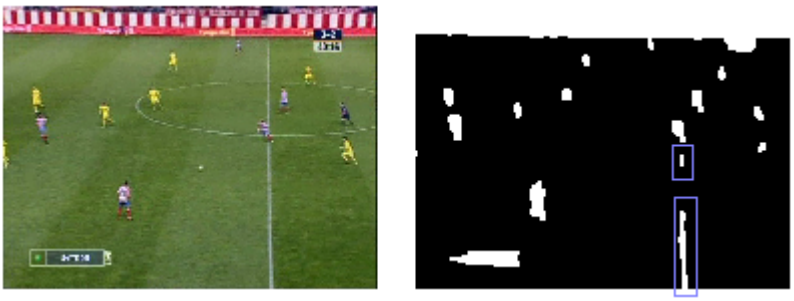


Imagen 75 - Detección de líneas erróneas del Atletico de Madrid - Villareal

2.- Prueba de detección

La detección de los jugadores ha sido correcta. Se han detectado los conjuntos de forma adecuada en todos los casos salvo uno de ellos que debido a su posición (pegado a la banda superior) ha sido incluido en ese conjunto y, por tanto, obviado en el seguimiento.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
13	13	12	1	0	0	13	13	92,3
13	12	12	1	0	1	13	12	92,9
12	12	11	1	0	0	12	12	91,7
12	12	10	2	0	0	12	12	83,3
12	12	11	1	0	0	12	12	91,7
12	9	8	2	0	2	12	9	83,3
9	9	9	0	0	0	9	9	100,0
9	9	8	1	0	0	9	9	88,9
9	9	8	1	0	0	9	9	88,9
9	9	9	0	0	0	9	9	100,0
							MEDIA	**91,3**

Tabla 15 - Porcentajes fragmento 2 del Atlético de Madrid - Villareal

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda

-	-	-	-	-	-	SÍ (1)
-	-	-	-	-	-	SÍ (1)
-	-	-	-	-	-	SÍ (1)
SÍ (1)	-	-	-	-	-	SÍ (1)
-	SÍ (1)	-	-	-	-	SÍ (1)
SÍ (1)	-	-	-	-	-	-
-	SÍ (1)	-	-	-	-	-
-	-	SÍ (1)	-	SÍ (1)	-	-
-	-	-	SÍ (1)	-	NO (1)	-
-	-	-	-	-	NO (1)	-

Tabla 16 - Notas de errores del fragmento 2 del Atlético de Madrid - Villareal

Como se ha mencionado anteriormente, los errores detectados por los problemas de segmentación de las líneas no se han incluido en los cálculos si no influyen en la detección de los jugadores. Es por esto por lo que conseguimos, aparentemente, unos resultados tan buenos en la tabla, con un porcentaje de acierto del 91,3% siendo el segundo mejor porcentaje hasta el momento. Si incluyéramos los errores detectados causados por las líneas blancas del campo, este porcentaje se reduciría en gran medida, pero la idea de la calidad del seguimiento se perdería ya que estos errores se producen por una mala segmentación y no por un mal seguimiento.

4.- Errores detectados

Como error más destacable en este fragmento tenemos la segmentación errónea de las líneas comentada antes. Del resto de errores típicos tenemos una detección de un doble conjunto y una unión de un jugador a la banda. De estos dos problemas conseguimos recuperarnos sin problema con el paso de los frames.

5.2.5 Partido Barcelona - Lyon

Este partido se caracteriza por tener tomas de cámara muy lejanas de forma que los jugadores son más pequeños con respecto al campo. En principio no debería de suponernos un problema pero, como veremos a continuación, los resultados no son tan satisfactorios como en los fragmentos anteriores. El umbral de cálculo de distancia en este caso es de 0.15 y el umbral HUE de 0.3

5.2.5.1 Fragmento 1 – Del frame 123175 al 123225

1.- Prueba de segmentación

Gran parte de la culpa de que los resultados obtenidos no sean aceptables es la segmentación. En este fragmento cometemos muchos más errores de segmentación que en el resto y es debido a la distancia que hay entre la cámara y los jugadores: a mayor distancia, los jugadores son más pequeños y la segmentación por color es peor.

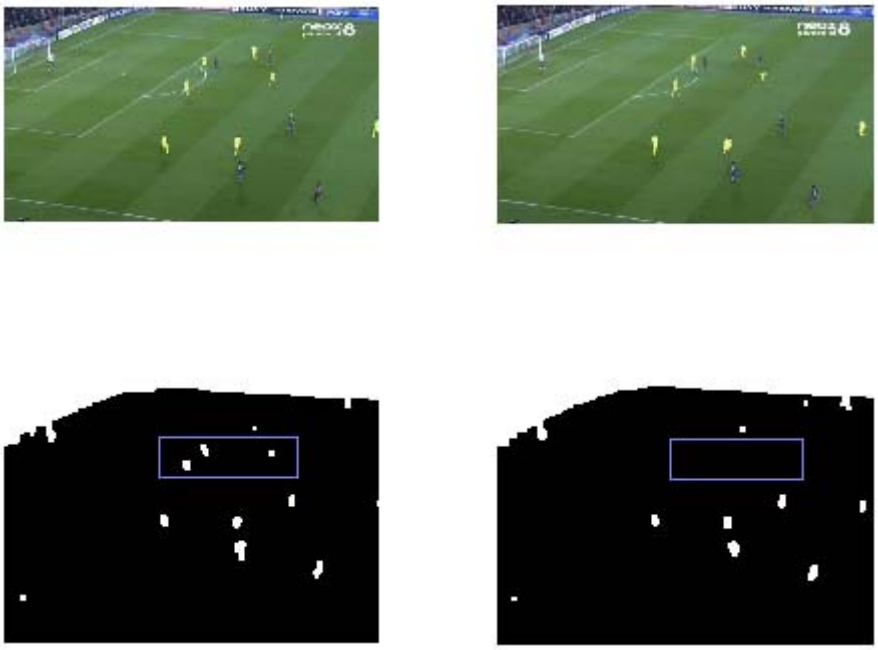


Imagen 76 - Errores de segmentación en intervalo 3 del Barcelona-Lyon

En la imagen podemos ver como tras el paso de 0.2 segundos el algoritmo de segmentación falla y no reconoce a tres jugadores que en el frame anterior sí lo hacía. Además podemos ver también que determinados jugadores no eran reconocidos en ninguno de los dos frames ya que en la parte cercana al marco del primer frame identifica a 4 jugadores mientras que debería identificar a 6. El número de fallos de segmentación que cometemos en este intervalo es muy alto, llegando a cometer 9 fallos en los 2 segundos que dura el intervalo.

2.- Prueba de detección

La detección en general ha sido buena, ya que la detección de los jugadores como conjuntos es aceptable aunque a consecuencia de la mala segmentación obtendremos una mala detección y un mal seguimiento.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
123175	123180	12	12	7	5	0	0	58,3
123180	123185	12	13	9	3	1	0	76,9
123185	123190	13	13	7	6	0	0	53,8
123190	123195	13	13	9	4	0	0	69,2
123195	123200	13	15	9	4	2	0	73,3
123200	123205	15	15	13	2	0	0	86,6
123205	123210	15	15	13	2	0	0	86,6
123210	123215	15	14	11	3	0	1	80
123215	123220	14	14	12	2	0	0	85,7
123220	123225	14	14	12	2	0	0	85,7

MEDIA	75,6

Tabla 17 - Porcentajes fragmento 1 del Barcelona - O.Lyon

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
SÍ (4)	SÍ (2)	-	-	-	-	SÍ (1)
-	NO (2)	-	-		-	SÍ (1)
SÍ (3)	NO (2)	-	-	-	-	SÍ (1)
-	SÍ (2) - NO(3)	-	-	-	-	SÍ (1)
SÍ (1)	SÍ (1) - NO (2)	-	-	-	-	SÍ (1)
-	SÍ (2)	-	-	SÍ (1)	-	SÍ (1)
-	-	-	-	SÍ (1)	-	SÍ (1)
SÍ (1)	-	-	-	SÍ (1)	SÍ (1)	SÍ (1)
-	SÍ (1)	-	-	-	NO (1)	SÍ (1)
-	-	-	-	-	NO (1)	SÍ (1)

Tabla 18 - Notas de errores del fragmento 1 del Barcelona - O. Lyon

Los fallos en la segmentación hacen que durante los primeros intervalos cometamos muchos fallos, como por ejemplo en el intervalo 3 mostrado en el que tenemos un porcentaje de acierto del 53%. Sin embargo, conseguimos que se recupere automáticamente de estos fallos iniciales teniendo un porcentaje de acierto de los 5 intervalos últimos (de 6 al 10) entre el 80% y el 86%. Por lo tanto, los resultados son los peores hasta el momento pero como parte positiva logramos recuperarnos en la mayoría de los casos hasta obtener un porcentaje de acierto cercano al habitual. Los fallos iniciales unidos a la recuperación posterior hace que tengamos un porcentaje de acierto medio del 75.6%, el más bajo hasta el momento.

4.- Errores detectados

Los errores más repetidos en este fragmento han sido por segmentación. En la imagen de las pruebas de segmentación podemos ver un ejemplo y en esta otra imagen podemos ver un error de seguimiento derivado de un error de segmentación:

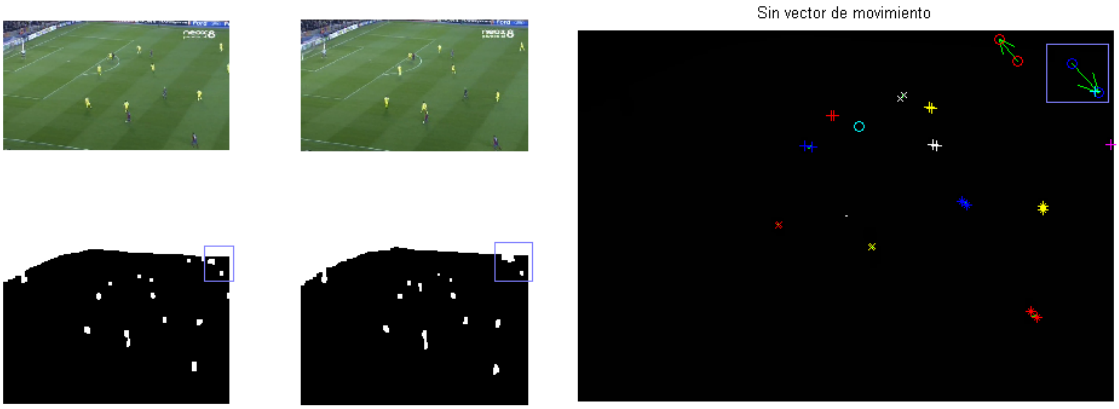


Imagen 77 - Error de detección derivado de un error de segmentación del Barcelona-Lyon

Podemos comprobar como el logo de la cadena es detectado pero no incluido en el público por lo que al hacer el seguimiento la asignación falla y se asigna al jugador.

5.2.5.2 Fragmento 2 – Del frame 34300 al 34350

1.- Prueba de segmentación

Al igual que en el fragmento anterior, hemos cometido unos fallos de segmentación iniciales pero de los que hemos recuperado al siguiente frame. Además, el número de fallos de segmentación en este caso es mucho menor, llegando en algunos casos a tener 15 jugadores en la imagen y el algoritmo de segmentación funcionar de forma perfecta (segmentado todos los conjuntos posibles). También hemos de añadir que en este caso la cámara se encontraba a una distancia menor por lo que la segmentación obtiene resultados mejores.

2.- Prueba de detección

La detección también ha sido óptima en este fragmento. Como hemos venido mencionando en todas las pruebas, la detección depende en gran medida de la segmentación. Al tener buenos resultados en la segmentación, hace que la detección de conjuntos sea también buena.

3.- Prueba de seguimiento

Los resultados obtenidos en este caso han sido los siguientes:

Frames		Jugadores en frame 1	Jugadores en frame 2	Seguimiento		No seguimiento		Porcentaje(%)
Frame 1	Frame 2			Correcto	Erróneo	Entrada jugador	Salida jugador	
34300	34305	15	15	12	3	0	0	80
34305	34310	15	14	11	3	0	1	80
34310	34315	14	15	14	0	1	0	100
34315	34320	15	15	15	0	0	0	100
34320	34325	15	15	15	0	0	0	100
34325	34330	15	15	14	1	0	0	93,3
34330	34335	15	16	13	2	1	0	87,5
34335	34340	16	17	13	3	1	0	82,4
34340	34345	17	17	14	3	0	0	82,4
34345	34350	17	17	16	1	0	0	94,1
							MEDIA	**90**

Tabla 19 - Porcentajes fragmento 2 del Barcelona - O.Lyon

Notas						
Fallo segmentación	Recuperación de fallo segmentación	Fallo por doble conjunto	Recuperación doble conjunto	Unión de jugadores	Desunión de jugadores	Unión a banda
SÍ (2)	SÍ (1)	-	-	SÍ (1)	SÍ (1)	-
SÍ (2)	NO (1)	-	-	-	-	-
-	SÍ (3)	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	SÍ (1)	-	-
-	-	-	-	SÍ (2)	-	-
SÍ (1)	-	-	-	-	NO (2)	-
-	NO (1)	-	-	-	NO (2)	-
-	SÍ (1)	-	-	-	SÍ (1) - NO (1)	-

Tabla 20 - Notas de errores del fragmento 2 del Barcelona - O. Lyon

Si medimos los datos medios de los aciertos para poder comparar resultados con los fragmentos anteriores, llegamos a un porcentaje medio de acierto del 89.96%. Este resultado es muy bueno si tenemos en cuenta que en el fragmento anterior de este mismo partido obtuvimos un 75.6%. Pero lo que llama la atención en este caso es que hemos obtenido durante 3 fragmentos seguidos un porcentaje de acierto del 100% con 15 jugadores sobre el campo. Estos resultados son muy buenos ya que el algoritmo funciona de manera perfecta durante un intervalo de tiempo con una gran carga de jugadores (15 de 23).

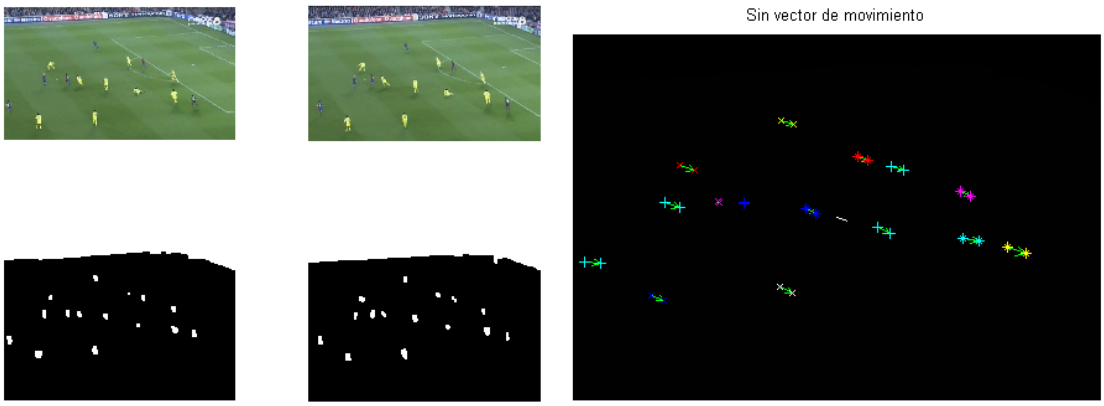


Imagen 78 - Seguimiento sin errores (intervalo 4) del Barcelona - Lyon

4.- Errores detectados

Los errores no han sido muchos y entran dentro de los casos habituales.

5.3 Comparativa de resultados

Los resultados obtenidos se muestran en la siguiente tabla:

Partido	Fragmento 1	Fragmento 2	Media
Liverpool – Real Madrid	87.3%	88.9%	88.1%
España – Rusia	79.3%	92%	85.65%
Almería – Valencia	80.9%	80.4%	80.65%
Atlético de Madrid – Villareal	90%	91.3%	90.65%
Barcelona – O. Lyon	75.6%	90%	82.8%

Tabla 21 - Comparativa resultados

Como hemos podido comprobar a lo largo de las distintas pruebas, los resultados han sido bastante buenos ya que obtenemos unos porcentajes de acierto bastante altos con un mínimo del 80% aproximadamente en la mayoría de los casos.

También debemos tener en cuenta que muchos de los errores cometidos se conocían antes de realizar las pruebas y sabíamos que debíamos tenerlos en cuenta. Estos errores serían la unión de los jugadores a las bandas o la unión de los jugadores cuando se encuentran muy cercanos. Estos errores podrían ser eliminados con más desarrollo del algoritmo pero se sale de las pretensiones de este proyecto.

A modo específico en los resultados podemos resaltar los buenos resultados que hemos obtenido en los dos fragmentos del Atlético de Madrid – Villareal, con un 90.65% de porcentaje de acierto medio. Nos llama la atención ya que ha obtenido mejores resultados incluso que el partido Liverpool – Real Madrid que nos ha servido la mayoría de las veces como vídeo de entrenamiento.

También destacamos los partidos España – Rusia y Barcelona – O.Lyon ya que los resultados de los dos fragmentos son muy dispares por lo que, aunque calculemos el valor medio, no podemos usarlo para darlo como resultado final ya que podría ser erróneo. En los otros tres casos los resultados en ambos fragmentos han sido similares por lo que sí nos podría dar idea del resultado final.

Sin embargo, aún teniendo en cuenta estos dos partidos con un menor porcentaje de acierto, el peor resultado obtenido en todas las pruebas es de 75.6% que nos indica aproximadamente que de cada 4 jugadores detectaríamos 3 que es una proporción aceptable.

5.4 Conclusiones

Como se ha mencionado en la introducción a este tema, la dificultad principal que encontramos en este proyecto a la hora de mostrar unos resultados se basa, sobre todo, en acompañar de la mayor objetividad posible a estos resultados. Como también se ha mencionado, esta tarea es muy difícil de conseguir debido, principalmente, a la dificultad de encontrar un testeo automático que indique la veracidad de los resultados obtenidos. Por lo tanto, creemos que la opción elegida podría ser aceptable teniendo en cuenta el resto de estudios similares hasta la fecha. Esta forma esta dotada de una gran subjetividad pero no se ha encontrado una táctica mejor para dotar a los resultados de un porcentaje de acierto o de fallo, de las recuperaciones ante fallos o de los errores más habituales.

Durante el estudio previo del estado del arte en lo que se refiere a la investigación del seguimiento de jugadores en partidos de fútbol mediante vídeos de televisión no se encontró ningún tipo de prueba uniforme a todos los casos por lo que, además, la comparativa entre estudios se hace mucho más confusa. En la mayoría de los casos, los investigadores introducen sus resultados sin redactar la forma en la que los han obtenido por lo que tampoco pudimos seguir un guión de pruebas similar a alguno de los casos para poder comparar resultados y así obtener la calidad de nuestro algoritmo por medio de la comparativa.

Respecto a los resultados obtenidos, de forma personal consideramos que los resultados han sido bastante buenos. De forma numérica, se ha probado para cuatro casos de test y el de pruebas y los resultados han sido muy similares en todos ellos lo que indica un buen entrenamiento del algoritmo. Como se ha citado en el apartado anterior, en todos los casos hemos obtenido como mínimo un 80% de porcentaje de acierto que, como veremos un poco más adelante, es un porcentaje elevado. Además, en los errores detectados como aceptables en la mayoría de los casos el algoritmo recuperaba el buen funcionamiento de forma automática (caso de desuniones de jugadores o fallos de segmentación) mientras que en los errores irreversibles (unión de jugadores por color o unión a la banda) se ha pensado en posibles soluciones aunque no implementadas por falta de tiempo en este proyecto. Por lo tanto, el número de errores cometidos por causa desconocida es muy bajo y esto hace que los errores habituales estén acotados por lo que se podría mejorar el algoritmo con más dedicación, lo que hace el algoritmo bastante prometedor en un corto-medio plazo de tiempo.

Si intentamos comparar nuestros resultados con los obtenidos por otros investigadores, nos chocamos con la forma de obtener estos resultados: en ninguno de los casos se explica los métodos usados. Por lo tanto, lo único que podemos hacer es intentar comparar nuestros resultados con los resultados obtenidos por otros estudios de la forma más ambigua posible: esto es, mediante la comparativa por porcentajes de acierto o de error. Recordamos que esta comparativa no es del todo aceptable ya que los métodos usados puede que no hayan sido los mismos por lo que podemos estar comparando dos algoritmos distintos. En este caso, se va a tratar de comparar los resultados obtenidos de forma abstracta sin tener en cuenta el algoritmo que lo consigue.

En [16] y [19] los algoritmos usados son similares a los nuestros. El caso de [16] utiliza una segmentación muy similar a la usada por nosotros, no así con el seguimiento de los jugadores. En el estudio [19] se basa en el entrenamiento del algoritmo con modelos de jugadores aunque la segmentación está basada también en el modelo de color y filtros aunque para la detección y seguimiento utiliza entrenamientos con modelos preestablecidos. Sin embargo, en ambos casos los resultados obtenidos son mostrados de forma abstracta con porcentaje de acierto o precisión.

Para el caso [16] obtiene dos tipos de resultados: un resultado para el seguimiento de todos los jugadores y otro para el seguimiento de aquellos que no tienen ningún problema (unión de jugadores, unión a líneas de campo, etc..). Para comparar con nuestro caso, tomaremos la primera opción:

	Recall	Precision
Video 1	72.7%	62.3%
Video 2	60.1%	76.3%
Overall	64.9%	66.9%

Imagen 79 - Tabla de resultados del estado del arte[20]

En este caso, la comparativa es muy favorable para nosotros ya que, tanto para los dos resultados concretos como para el resultado medio nuestro algoritmo obtiene mejores resultados obteniendo, en el peor de los casos, una precisión similar.

Para el estudio realizado en [19] los resultados son muy buenos, mejorando bastante a los resultados obtenidos por nosotros:

GT \ INF	R (%)	TA(%)	TB(%)
R	82.93	17.07	0.00
TA	0.38	98.30	1.32
TB	0.00	2.31	97.69

GT \ INF	R (%)	TA(%)	TB(%)
R	97.12	2.88	0.00
TA	1.23	97.75	1.02
TB	3.18	1.23	95.59

Imagen 80 - Tabla de resultados del estado de arte II[21]

En este caso, se ha obtenido un porcentaje de acierto mínimo del 95,59% muy por encima de nuestro mejor porcentaje de acierto. Sin embargo, hemos de decir que para la obtención de estos resultados en este estudio solo se han utilizado los jugadores detectados y segmentados de forma correcta por lo que esos errores no son tenidos en cuenta. En este caso, la comparativa nos es posible ya que nosotros no hemos realizado de esta forma las pruebas, en [19] se han separado los resultados por segmentación-detección y seguimiento mientras que en nuestro caso se ha unido todo.

[20] Imagen obtenida de [16]
[21] Imagen obtenido de [19]

6. CONCLUSIONES Y LÍNEAS FUTURAS

6.1 Conclusiones

Cuando comenzamos el estudio previo de este proyecto fin de carrera nos dimos cuenta de que la detección y seguimiento de jugadores en partidos de fútbol, a pesar de ser un campo bastante estudiado, no iba a ser una tarea fácil debido, sobre todo, a la alta complejidad de realizar algoritmos de procesado de imágenes.

Durante la **fase de investigación** se encontraron diversos tipos de estudios relacionados con el tema tratado en este proyecto: desde simples algoritmos de segmentación del campo hasta complejos desarrollos de seguimiento de jugadores. Además, pudimos comprobar como estos algoritmos de seguimiento son usados por determinadas empresas de forma comercial para obtener soluciones a las necesidades propuestas por equipos deportivos de alto nivel. También comprobamos, por ejemplo, que los kilómetros recorridos por un jugador se calculan con un algoritmo de seguimiento lo que nos hizo presuponer que estos algoritmos nos los podemos encontrar también de forma cotidiana aunque no nos hayamos parado a pensarlo. Esta fase previa de estudio, nos hizo prever que el campo sobre el que íbamos a investigar no era un campo inexplorado ni novedoso aunque sí nos encontramos con bastantes líneas de trabajo abiertas a mejoras. Lo que sí pudimos identificar de forma más genérica al realizar este estudio previo es que existen tres vías de estudio en este tipo de proyectos:

- Seguimiento con visión total del campo (caso usado por las empresas profesionales)
- Seguimiento por visión de las retransmisiones "en bruto"
- Seguimiento por visión de televisión

La complejidad de cada una de las vías de estudio va en aumento, ya que conseguir un seguimiento aceptable se hace más complicado a medida que los vídeos utilizados se vuelven más genéricos y no tan adaptados a las necesidades concretas.

Por lo tanto, se decide elegir la tercera de las vías de estudio ya que a priori es la vía menos desarrollada y con más posibilidad de mejora. Además, de cara a la creación de la base de datos necesaria para el estudio, la obtención de vídeos de partidos de fútbol retransmitidos es más sencilla que cualquiera de las otras dos vías (para el primer caso es inviable). Por lo tanto, la diferencia principal en nuestro estudio con el resto se basa en la dificultad de obtener un seguimiento sin necesidad de tener un hardware específico o una forma de captura de imagen concreta. Y todo esto obteniendo unos resultados medibles y razonables.

Como **objetivo principal** nos planteamos detectar el movimiento de cada uno de los jugadores usando la captura de la emisión de partidos de la televisión. Debido a la alta dificultad de realizar un sistema online, se descartó esta opción y se optó por desarrollar el sistema partiendo de unos vídeos conocidos.

Como se puede comprobar en el índice de este proyecto, se ha intentado dividir el desarrollo de algoritmo en tres partes bien diferenciadas aunque con mucha relación entre sí. Estas tres partes son:

- *Segmentación* del campo, en la que intentamos transformar cada frame del vídeo original en un frame 'binario' en el que por un lado tengamos el césped y por otro el resto de la imagen.

- *Detección* de los jugadores, intentando asignar a cada jugador detectado una coordenada para poder posicionarlo sobre la imagen y detectar el movimiento.

- *Cálculo de movimiento,* a partir de las posiciones anteriores y mediante un algoritmo de asociación de puntos.

Durante la **fase de segmentación** nos hemos apoyado con bastante frecuencia en los estudios previos ya que, independientemente del tipo de seguimiento escogido, la mayor parte de ellos realizaban la segmentación como primer paso. Incluso hemos encontrado textos en los que únicamente se realizaban algoritmos de segmentación. Como se ha comentado durante el tema, el tipo de segmentación escogida se basa en el color de los píxeles y en la distancia entre la representación RGB de cada píxel. Esto hizo que nos encontráramos con el primer problema: elegir el píxel de referencia para realizar la comparativa. Como solución, se optó por la más sencilla aunque menos deseable ya que deberíamos introducir de forma manual y a modo de configuración previa este píxel de referencia. El otro problema grave con el que nos enfrentamos fue el tiempo. Nuestro algoritmo inicial basado en la comparativa píxel tenía un tiempo de procesamiento muy alto por lo que el tiempo obtenido para segmentar cada frame era bastante más grande del esperado. La solución tomada (realizar la segmentación bloque) fue muy buena ya que el tiempo de procesamiento adquirió unos valores más realistas. Después de encontrar solución a varios problemas más nos encontramos con una segmentación de frames de calidad y pasando a la siguiente fase.

La **fase de detección** de los jugadores se apoyó básicamente en dos aspectos. Por un lado tenemos el descubrimiento de una rutina de Matlab que encajaba a la perfección con el etiquetado necesario de los jugadores. Esta rutina adquiere un valor especial en esta fase ya que implementa básicamente la idea principal necesaria para realizar esta fase de detección. En el otro lado tenemos el uso de filtros morfológicos para mejora de la imagen inicial obtenida. Estos filtros hacen que determinados errores que podríamos cometer en la segmentación queden eliminados llegando a obtener un incremento de calidad notable. Esta fase ha sido la más compleja en el desarrollo ya que se tenía que adecuar en gran medida a los resultados obtenidos durante la fase de entrenamiento y pruebas y eran necesarias modificaciones sobre problemas que nos íbamos encontrando sobre la marcha. En el apartado de problemas encontrados de este capítulo podemos comprobar todos estos problemas concretos.

En la fase de **cálculo de movimiento** hemos tenido menos problemas ya que la fase anterior tenía como salida las coordenadas de cada uno de los jugadores en los frames consecutivos por lo que hubo que implementar simplemente un algoritmo de asignación entre coordenadas correcto. El mayor problema que nos encontramos en esta fase fue distinguir la diferencia entre el movimiento de los jugadores en la imagen y el movimiento real. Esta diferencia provenía del propio movimiento de la cámara de televisión. La solución tomada en este caso fue calcular un 'posible' movimiento de la cámara y utilizarlo como corrector aunque se han dejado ambas implementaciones en función de las necesidades futuras de cada caso.

Una vez realizado el algoritmo y complementado con una fase de pruebas y testeo inicial se continuó por la **fase de obtención de resultados**. Al comienzo del desarrollo de este sistema no pensamos que esta fase nos llevaría tanto esfuerzo ya que no detectamos el problema principal de esta fase en este tipo de sistemas y que consistía en la forma en la que evaluaríamos estos resultados. Como ya se ha mencionado, esta evaluación de resultados es bastante subjetiva mientras que la teoría indica que las pruebas deberían ser lo más objetivas posible. Sin embargo, rastreando los diferentes estudios similares al nuestro ha sido imposible encontrar una implementación razonable de esta fase de pruebas que pudiéramos utilizar o por lo menos comparar. En la mayoría de los casos los investigadores eligen una forma de evaluación propia (en ningún caso totalmente objetiva) y evalúan los resultados a partir de ellas por lo que la fase de comparación en este caso ha sido prácticamente nula. En lo que concierne a nuestro caso, la fase de resultados ha intentado ser lo más objetiva posible dentro de los límites reales. Es complicado evaluar si un seguimiento de un determinado jugador ha sido correcto sin recurrir a la supervisión humana.

Tras realizar todos los pasos anteriores, solo queda la **fase de evaluación de resultados**. En esta fase ha quedado reflejado que estamos bastante satisfechos con los resultados obtenidos ya que en todos los fragmentos usados los resultados de las pruebas han sido muy buenos.

Entre los aspectos positivos del proyecto destacamos el trabajo de la fase de segmentación, en la que hemos tenido que llegar hasta el más bajo nivel de la imagen (color de cada píxel) para obtener un frame adecuado a nuestras necesidades. También destacamos las diferentes decisiones tomadas a todos los problemas encontrados, ya que en la mayoría de los casos se ha llegado a compromisos entre el trabajo necesario y el impacto en los resultados así como el orden seguido en el desarrollo del sistema, ya que hemos seguido un orden adecuado y lineal hasta obtener unos resultados similares a los esperados. También nos gustaría detallar la gran respuesta recibida por parte de Asier Zubillaga (responsable de Amisco España) en la tarea de investigación inicial al contestarnos a determinadas dudas surgidas sobre el funcionamiento de su algoritmo comercial.

Entre los aspectos negativos del proyecto cabría destacar la imposibilidad de desarrollar soluciones más elegantes a determinados tipos de problema por falta de tiempo o por aumento excesivo de la carga del proyecto. Dentro de este apartado englobamos, por ejemplo, la interacción manual inicial o la división por color de los distintos jugadores y árbitro. Estos aspectos negativos serán mostrados (con breves ideas surgidas) en las líneas futuras para ser tenidas en cuenta en estudios posteriores que sigan la línea de trabajo marcada durante todo este proyecto.

6.2 Líneas futuras

6.2.1 Sobre el algoritmo

Como se ha mencionado en las conclusiones al apartado de resultados y en las conclusiones finales del apartado anterior, los resultados obtenidos son muy buenos y bastante prometedores. Son muy buenos ya que la precisión obtenida en la mayoría de los casos supera los resultados esperados y son prometedores porque aún hay bastante margen de mejora del algoritmo. Como líneas futuras se van a comentar tres tipos de casos que nos hemos encontrado y en los que se ha intentado dotar de una solución pero no se han implementado debido al alto coste temporal de desarrollo asociado.

Como primer aspecto a mejorar en nuestro algoritmo obtenemos la carga manual inicial del verde de referencia. En nuestro algoritmo este verde de referencia es introducido de forma manual por lo que se debería intentar automatizar por completo este paso inicial. Esta solución entra dentro de lo aceptable, ya que podría considerarse una configuración previa del algoritmo aunque se pensó otra posible implementación alternativa en la que no fuera necesario esta configuración inicial. La alternativa era comenzar con un verde de referencia fijo (que podría ser el verde puro RGB=[0,255,0]) e ir calculando un verde de referencia en cada frame, que tuviera en cuenta todos los píxeles detectados como césped. Inicialmente el umbral de detección sería mayor ya que habría más distancia entre el verde puro y el de cada píxel aunque este umbral debería irse reduciendo conforme fuéramos consiguiendo un verde de referencia ideal. De modo esquemático sería algo como esto:

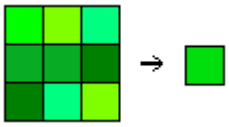
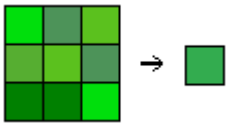
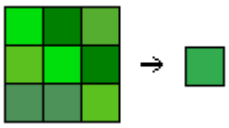
Frame	Frame 1	Frame 2	Frame 3	Frame 4
Verde de referencia	 Verde puro	 Verde inicial	 Similar al verde de referencia	 Verde de referencia deseado
Umbral	↑↑↑	↑↑	↑	↑

Tabla 22 - Esquema obtención verde de referencia

A partir del frame 3, el cuadro de píxeles similares al de referencia sería mucho más amplio ya que cada vez los píxeles serían más similares al de referencia, no se ha puesto por motivos de espacio. Esta opción llevaría asociada un coste de estabilización del verde de referencia por lo que habría que estudiar que sería más costoso: el tiempo de estabilización o la introducción del píxel de forma manual.

En el segundo aspecto mejorable del algoritmo nos encontramos con la separación de los jugadores en la segmentación por colores del uniforme. Como ya se ha comentado, nuestro algoritmo no distingue los colores de los uniformes por lo que tanto el árbitro como los jugadores de ambos equipos no son diferenciados entre ellos. Si pudiéramos diferenciar los distintos equipos, obtendríamos mejoras tanto a nivel de segmentación, ya que podríamos detectar dos jugadores unidos entre sí por el color de su camiseta, como a nivel de seguimiento al poder diferenciar una mala asignación entre jugadores

de distinto equipo. Para realizar esta mejora pensamos en una segmentación extra que indique el valor negro en el césped y los jugadores los muestre tal y como aparecen en la imagen inicial. De esta forma y mediante un procesado de histogramas de color podríamos detectar los colores básicos de ambas camisetas y conocer su representación RGB. Después, sería necesario pasar estos jugadores a la representación habitual (valor blanco) y realizar el etiquetado. Mediante una asociación intermedia se podría saber que valores etiquetados pertenecen a un equipo u a otro. Se ha realizado un desarrollo inicial de este algoritmo que identificaría los dos colores dominantes sobre una escena negra utilizando el campo de tonalidad y mediante el uso de histograma de color que sería el punto de partida de esta mejora propuesta. Este algoritmo detecta los colores predominantes en la escena, elimina el color negro y decide que los dos colores predominantes restantes son los colores de los dos equipos. Este desarrollo es muy básico y tendría multitud de problemas posteriores (color del árbitro, porteros, gama de colores de las camisetas, etc.). Lo único que se ha querido mostrar en esta parte es el punto de partida sobre el que se podría empezar a desarrollar.

Ya como tercer aspecto podemos distinguir el error cometido cuando algún jugador se encuentra cercano a la banda superior del campo. Como se ha podido ver en el apartado de resultados, este caso era bastante habitual por lo que podríamos reducir el número de fallos encontrando una solución a este problema. La búsqueda y desarrollo de esta solución la encontramos bastante compleja por lo que simplemente se ha pensado en una leve pincelada de lo que podría ser la posible solución. Esta pequeña introducción la encontramos en el encuadre del campo: si encontráramos alguna forma de poder delimitar el campo respecto de las gradas podríamos distinguir a aquellos jugadores que se fundieran con esta de forma sencilla. Sin embargo, esta delimitación se antoja muy complicada debido al movimiento constante de la cámara por todo el terreno de juego.

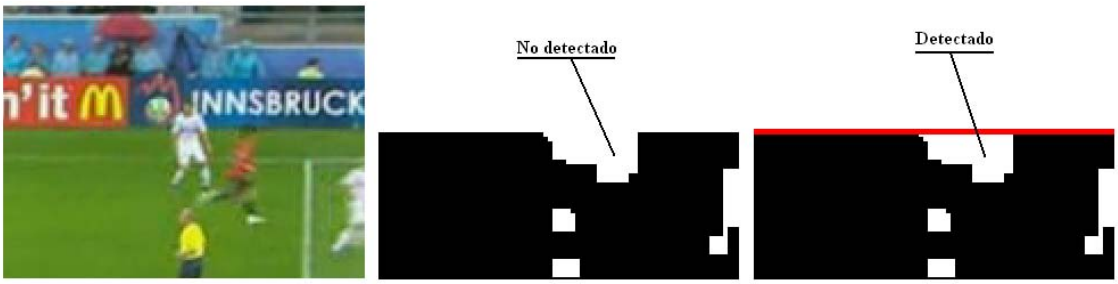


Imagen 81 - Posible encuadre del campo

Como se ha mencionado anteriormente, a priori esta posible implementación de mejora es la más complicada de las tres mostradas.

6.2.2 Sobre el uso del algoritmo

En este subapartado se quiere hacer una pequeña introducción sobre los posibles usos del algoritmo a largo plazo.

Entre los usos más destacados en la rama deportiva observamos la detección de movimiento de los jugadores en el campo y su posterior estudio para las tácticas de los equipos, el análisis parcial o total de un jugador determinado o la detección de eventos de alta importancia en el partido para su posterior procesado y etiquetado. Además cabe

destacar que nuestro algoritmo, al depender de un píxel de referencia podría usarse sobre otro tipo de terrenos de juego deportivos distintos como pueden ser el baloncesto, el rugby o el hockey.

Imagen 82 - Eye tracking[22]

Posteriormente, como ideas propias y llevando el algoritmo a un desarrollo perfecto se podría usar sobre algoritmos de interacción persona-ordenador [30] (eye-tracking, en la imagen), detección de intrusos en casas, rastreo de coches sobre zonas uniformes o movimientos militares sobre zonas desérticas.

[22] Imagen obtenida del Instituto Max Planck de psicolingüística.

7. PRESUPUESTO

7.1 Introducción

En este capítulo final se va a realizar un presupuesto aproximado de los distintos conceptos que han sido necesarios para la realización de este proyecto. De esta forma podremos cuantificar el esfuerzo realizado en su elaboración y obtener también una valoración económica.

Dentro del propio presupuesto vamos a diferenciar dos partes, de forma que la primera sea el coste material de la realización del proyecto y la segunda se refiera al coste humano.

Por coste material podemos entender tanto desarrollo hardware y software como las instalaciones, aparatos y programas necesarios para su elaboración.

Por coste humano vamos a tener en cuenta el coste del salario de todas las personas involucradas en la realización del mismo.

7.2 Coste material

El coste material derivado de la realización de este proyecto es el siguiente:

- Ordenador portátil, valorado en 600€. Conociendo que el coeficiente máximo de amortización de los equipos para el tratamiento de la información es del 25% y el periodo máximo de uso es de 8 años [31] y sabiendo que el ordenador usado tiene una antigüedad de 3 años podemos calcular su valor aproximado en **253€**.

- El lugar de desarrollo del trabajo, que conlleva unos gastos aproximados de 50€/mes. La duración aproximada del proyecto ha sido de 12 meses lo que llevaría a un coste final de **600€**.

- Software Matlab, cuya licencia de estudiante con el paquete de procesado de imágenes tiene un coste de **1100€**.

- Sistema operativo necesario para poder ejecutar el sotware: Windows XP cuyo precio va incluido en el coste del ordenador portátil.

- Coste de Internet, con un precio medio por mes de 30€. Este servicio se ha utilizado para la búsqueda de información y contacto con el tutor. El coste total durante los 12 meses sería de **360€**.

- Otros conceptos, como pueden ser fotocopias, impresión y encuadernación, transporte, etc. con un coste estimado de **300€**.

La suma de estos costes se muestra en la siguiente tabla, en la que podemos ver los distintos conceptos y su coste:

Concepto	Precio
Ordenador portátil	253€
Lugar de trabajo	600€
Software Matlab	1.100€
Windows XP	Incluido en ordenador portátil
Internet	360€
Otros conceptos	300€
Total	**2.613€**

Tabla 23 - Costes materiales

7.3 Coste humano

Para obtener el coste derivado de los salarios derivados del proyecto se ha tenido en cuenta tanto los honorarios del proyectante como del tutor, derivados de la dirección del mismo.

Los honorarios correspondientes a un ingeniero de telecomunicación han sido obtenidos de la página web del colegio oficial de ingenieros de telecomunicación [32] y ascienden a 70€.

Para el cálculo de los honorarios debemos tener en cuenta las siguientes consideraciones:

- El proyecto se ha realizado en 12 meses. Si consideramos que un mes consta de 20 días laborables y se han dedicado 2 horas/día de media llegamos a un resultante de horas de trabajo de 480 horas, lo que derivaría en un coste de 33600€.
- La dirección del proyecto ha sido realizada por el profesor D. Jesús Cid Sueiro que ha empleado aproximadamente unas 100 horas, lo que supone un coste de 7000€.

Usando estos honorarios llegamos a la siguiente tabla:

Concepto	Precio
Honorarios proyectante	33.600€
Honorarios director	7.000€
Total	**40.600€**

Tabla 24 - Coste humano

7.4 Coste total y presupuesto

Para poder realizar el presupuesto vamos a tener en cuenta la suma de los dos tipos de costes anteriores. De esta forma, el coste total asociado al proyecto resultaría en la siguiente tabla:

Concepto	Precio
Coste material	2.613€
Coste humano	40.600€
Total	**43.213€**

Tabla 25 - Costes totales

Si tenemos en cuenta el I.V.A. aplicable en estos casos (16%) llegamos a que el presupuesto final del proyecto asciende a cuarenta y seis mil sesenta y siete con ocho céntimos de euro (50.127,08€).

En Leganés, a 21 de Mayo de 2010
El ingeniero proyectista,

Fdo: Jose Manuel Moreno García.

8. BIBLIOGRAFÍA

[1] *FIFA Big Count 2006: 270 million people active in football. Sitio web oficial de la FIFA.*
[2] INE – Instituto Nacional de Estadistica, tercer trimestre de 2009. DOI=http://www.ine.es/
[3] *La Sexta*© GESTORA DE INVERSIONES AUDIOVISUALES LA SEXTA, S.A., con C.I.F. A-84434935
[4] SUP© *Sport Universal Process* copyright SUP 2004 - Tous droits réservés
[5] *Delta Tre*© Copyright deltatre 2007 All rights reserved
[6] *Amisco en El Día Después*, DOI=http://www.youtube.com/watch?v=grhzeIX8AKQ
[7] *Roja Directa,* Creative Commons © DOI=http://www.rojadirecta.com
[8] *DivX*, © 2009 DivX, Inc. All rights reserved. DivX® is a registered trademark of DivX, Inc.
[9] F. V. Jensen, *An Introduction to Bayesian Networks.* New York: Springer, 1996.
[10] J. Assfalg and M. Bertini, *Semantic annotation of soccer vídeo: automatic highlights identification,* Comput. Vis. Image Undestand, vol. 91, No. 3, 2003.
[11] K. Wan, J.H. Lim, C. Xu and X.Yu, *Real-time camera field-view tracking in soccer vídeo*, in Proc. ICASSP, Hong Kong, 2003, vol. 3, pp.185-188.
[12] C. L. Huang, H. C. Shih and C. Y. Chao, *Semantic analysis of soccer vídeo using dynamic Bayesian network*, IEEE Transactions on multimedia, vol. 8, No 4, 2006.
[13] Janez Pers, Matej Kristan, Matej Perse and Stanislav Kovacic *Analysis of Player Motion in Sports Matches.* University of Ljubljana.
[14] Thomas Mauthner and Horst Bischof *A Robust Multiple Object Tracking for Sport Applications*. Graz University of Technology, Austria.
[15] D.Zhong and S.F.Chang *Video Object Model and Segmentation for Content-Based Video Indexing.* University of Columbia, Ney York, USA.
[16] Okihisa Utsumi, Koichi Miura, Ichiro IDE, Shuichi Sakai and Hidehiko Tanaka, *An object detection method for describing soccer games from vide.* National Institute of Informatics, University of Tokio.
[17] Xinguo Yu, Changshen Xu, Qi Tian and Hon Wai Leong, *A ball tracking framework for broadcast soccer video*. Infocomm Res. Inst., Heng Mui Keng Terrace, Singapore
[18] Xinguo Yu, Changsheng Xu, Hon Wai Leong, Qi Tian, Qing Tang, and Kong Wah Wan. *Trajectory-Based Ball Detection and Tracking with Applications to Semantic Analysis of Broadcast Soccer Video*. Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore
[19] Jia Liu1, Xiaofeng Tong, Wenlong Li, Tao Wang, Yimin Zhang, Hongqi Wang, Bo Yang, Lifeng Sun, Shiqiang Yang. *Automatic Player Detection, Labeling and Tracking in Broadcast Soccer Video.* China.
[20] *Matlab,* © 1994-2009 The MathWorks, Inc.
[21] AVI, formato de archivo contenedor de audio y vídeo. Microsoft, 1992 DOI= http://support.microsoft.com/kb/316992
[22] PAL, Sistema de codificación analógica usado en Europa.
[23] Wordreference, diccionario en línea gratuito. DOI=http://www.wordreference.com/es/
[24] Distancia de Mahalanobis, Wikipedia. DOI=http://es.wikipedia.org/wiki/Distancia_de_Mahalanobis
[25] Pablo Ronclagio B, *Ejemplo práctico de función de segmentación de color utilizando la distancia de Mahalanobis.* DOI= http://www.elo.utfsm.cl/~elo328/PDI14_EjemploMahalanobis.pdf
[26] Modelo de color HSV, Wikipedia. DOI=http://es.wikipedia.org/wiki/Modelo_de_color_HSV
[27] Pelayo García García and Rubén Panizo Poncelas, *Filtros Morfológicos*, DOI= http://zip.rincondelvago.com/00062764
[28] Departamento de Teoría de la Señal, *Operaciones Morfológicas,* Universidad Carlos III de Madrid DOI=http://www.tsc.uc3m.es/imagine/Curso_ProcesadoMorfologico/Contenido/Operaciones/OperacionesMorfologicas.html
[29] Rentian Xiong, *vectarrow.m,* Abril 2005, DOI= www.mathworks.com
[30]Yusef Hassan Montero y Víctor Herrero-Solana, *Eye tracking con interacción personal*, Octubre de 2007. DOI=http://www.nosolousabilidad.com/articulos/eye-tracking.htm
[31] Agencia Tributaria Española. Datos del años 2010. DOI=www.aeat.es
[32] Colegio oficial de ingenieros de telecomunicación. DOI=*www.coit.es/*