Proyecto Fin de Grado
Grado en Ingeniería de las tecnologías de telecomunicación
Intensificación Imagen y Sonido

Sistema de seguimiento de la acción en eventos deportivos

Autor: Fco. Javier Cortés Conde

Tutor: José Ramón Cerquides Bueno

Dep. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Grado
Grado en Ingeniería de las tecnologías de telecomunicación

Sistema de seguimiento de la acción en eventos deportivos

Autor:
Fco. Javier Cortés Conde

Tutor:
José Ramón Cerquides Bueno
Profesor titular

Dep. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Proyecto Fin de Grado: Sistema de seguimiento de la acción en eventos deportivos

Autor: Fco. Javier Cortés Conde

Tutor: José Ramón Cerquides Bueno

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

La realización de este texto ha sido posible gracias al tiempo que dejé de dedicarle a mis seres queridos, que son y siempre serán lo más importante en mi vida.

Javier Cortés Conde

Estudiante de grado en ingeniería de las tecnologías de telecomunicación

Sevilla, 2016

Resumen

Este documento pretende estudiar distintos modos de seguimiento de la acción en eventos deportivos utilizando análisis in situ de imágenes captadas por una cámara. Para esta empresa, se realiza un procesado de la información que proporcionan las imágnes con el objetivo de producir un ángulo de paneo, un valor que será el ángulo de rotación de un servo mecánico. Aunque inicialmente ese es el objetivo principal, este documento tratará sobre el análisis de la imágen de video exclusivamente.

El análisis de las imágenes se ha realizado mediente código Matlab® ya que proporcionaba un modo sencillo de experimentar con distintos algoritmos.

La razón principal que me ha llevado a la realización de este proyecto, aunque mas adelante se mostrará mas detallada, es el fuerte vínculo que me une al mundo del baloncesto, en concreto al arbitraje del mismo, todo un mundo desconocido que, tiene muchísimas horas de trabajo detrás y personas fasciantes.

Abstract

This document aims to study different ways of monitoring the action at sporting events using real-time analysis of the images captured by a camera. This requires a processing of the information containing in the source in order to provide a pan angle, a value that will be the angle of rotation of a mechanical servo. Although initially that was the objective, this paper will focus on the analysis of the video image only.

-English version-

Índice

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y MOTIVACIÓN

No basta tener buen ingenio, lo principal es aplicarlo bien.

- René Descartes -

El objetivo de este proyecto era en un principio poder realizar una grabación de un evento deportivo sin necesidad de un operador de cámara que se encargara de realizar los paneos de la grabación. Tras haber investigdo sobre el tema, veremos el desarrollo en el siguiente apartado, se llega a la conclusión de que no había ningún sistema que utilizase el análisis de la información del video in situ[1] para hacer frente a este reto.

Para esta labor he desarrollado un algoritmo utilizando el lenguaje Matlab® que servirá como solución al problema.

Dentro del proyecto que es crear un producto capaz de sustituir a un operario de cámara en un evento deportivo, este texto se centra en el sistema de seguimiento mediante el análisis de la imagen en tiempo real.

1.1. Motivación Personal

Mi motivación personal es la necesidad de grabar partidos de baloncesto, deporte el cual está muy unido a mi ya que desde hace años soy árbitro de la Federación Andaluza de Baloncesto.

Un árbitro necesita ver los videos de los partidos para analizar los arbitrajes en detalle y así mejorar en la toma de decisiones, colocación e imagen. Tras hablar con compañeros y entrenadores por toda España y comprobar que era una necesidad colectiva, decidí embarcarme en este proyecto cuyo pilar fundamental que hace que se diferencie de la mayoría de soluciones existentes actualmente en el mercado es el uso de la visión artificial para obtener información activa de un partido. Para ello se realizará una breve exposición de los avances que ya hay en el campo de la grabación autónoma de eventos deportivos y en los sistemas de seguimiento activo para ese mismo fin.

[1] En el mismo momento de la grabación

1.2 Alcance

El hecho de focalizar la investigación a un uso en eventos deportivos surge a raíz de una necesidad propia, pero hay infinidad de campos en los que un análisis inteligente de la imagen in situ es muy interesante. Además de la obvia que podría ser adaptar el algoritmo para otros deportes además del baloncesto, esta investigación puede llegar al almacenamiento eficiente de datos provenientes de cámaras de vigilancia, contador de personas, identificación personal…

1.2.1 Otras Aplicaciones

Como ya se ha mencionado, además de aplicaciones como adaptar el algoritmo para otros deportes tales como Fútbol, Fútbol Sala, Balonmano, Natación y en general cualquier deporte que implique un paneo de la cámara por parte de un operador de cámara; se muestan algunas idéas que no tienen tanto que ver con las anteriores.

Como sabemos hay miles de recintos protegidos con cámaras de seguridad cuyos datos producidos deben ser almacenados para ser posteriormente visualizados si es necesario [1] [2]. Controlar la cámara para que almacene unicamente el tiempo que haya movimiento destacable en el encuadre ahorraría millones de GB de almacenamiento al año[2] en todo el mundo.

Recalcando la idea principal, el modo de detección de movimiento que se propone no incluye sensores pasivos (como por ejemplo los encargados de evitar que una puerta de un ascensor se cierre) sino que es el análisis de la imagen recibida el que determinará autónomamente, mediante algoritmos, si hay movimiento relevante en la graciación y por tanto si merece realmente la pena almacenar la información que adquiere la cámara.

Esta idea es una solución eficaz por ejemplo para el caso de una cámara de vigilancia en el exterior: la cámara detecta el movimiento de un árbol por el viento, con el análisis de la imagen ese movimiento del entorno no sería considerado por el algoritmo como suficientemente relevante como para almacenarlo.

Otro ejemplo, que no tiene tanto que ver con el movimiento sino con la interpretación de la imagen, podría darse en la aduana de un aeropuerto en la que varias decenas de pasajeros llegan a una terminal y con una o varias cámaras la policía es capaz de identificar a todos los pasajeros sin necesidad de documentación de identificación física [3]. Esto no solo facilita el trabajo a la guardía de aduanas sino que agiliza muchísimo el proceso de identificación de personas y el tiempo de espera de los pasajeros.

1.3 Estructura

Por lo tanto, este proyecto estará enfocado a: una **solución que permita sustituir un operario de cámara en un evento deportivo** que, en este caso, será el de un partido de Baloncesto.

Este documento lo he dividido en varios aparatados además del actual:

- Estado del arte de la tecnología de seguimiento
- Estructura del algoritmo centrado en conjuntos
- Conclusiones
- Código

[2] Un día de grabación con calidad mínima de 176x120 con una codificación H.264 suponen 920 MB. Ocho mil cámaras en un recinto como el metro de madrid en un día supondrían más de 5000 TB de espacio necesario para hacer frente a un solo día.

2 ESTADO DEL ARTE DE LA TECNOLOGÍA DE SEGUIMIENTO

La mente que se abre a una nueva idea nunca vuelve a su tamaño original.

- Albert Einstein -

Actualmente existen multitud de soluciones muy interesantes para grabar de forma autónoma eventos deportivos, presentaciones[3], entrenamientos y demás situaciones. Un punto común entre todas estas soluciones comerciales es la necesidad de un dispositivo adicional que debe llevar el sujeto que es grabado, es decir, no interpretan la información de la imagen, no les importa lo que graben, siemplemente siguen una etiqueta (tag).

Veremos también un *paper* (Mimicking Human Camera Operators) sobre lo que, en esta ocasión, es un estudio e interpretación de la imagen, el problema es que este *paper* parte de una base no sencilla de conseguir, la cual es que tenemos a todos los jugadores localizados en el campo en todo momento lo cual se consigue con una cámara fija colocada en el techo o en un lugar que abarque todo el campo.

Dicho esto, pasamos a describir varias soluciones comerciales al problema del seguimiento y posteriormente veremos el paper mencionado anteriormente.

Aunque antes de nada se introduce brevemente la tecnología de localización local usada para algunos sistemas de seguimiento, nos referimos a RTLS *Real-time Location Services*.

2.1 Real-Time Location Services

La razón por la que se habla de este sistema es que es dicho sistema el empleado por todos los productos comerciales destinados a la solución de nuestro problema. Debido a ello para entender como funcionan estos productos debemos conocer, aunque sea minimamente en qué consiste, qué es y como funciona la tecnología RTLS.

Los Sistemas de Localización en Tiempo Real (RTLS) son sistemas completamente automáticos (pueden cumplir la condición de sustituir a un operario de cámara) que monitorizan con una determinada frecuencia, la posición de un elemento que puede estar o no en movimiento.

[3] Del estilo de las keynotes de Apple

La tecnología RTLS, son aplicaciones que comenzaron a usarse desde los años 70 para el tracking de misiles y aplicaciones de telemetría [4]. Recientemente, su uso se ha ido extendido hacia los sectores industriales, en logística y para la identificación de animales en una granja.

2.1.1. Funcionamiento y elementos

Para su funcionamiento utilizan tecnología de alta frecuencia que permite localizar los componentes. Como mínimo el sistema tiene 2 elementos:

- Interrogator o reader, que contiene un módulo de radio frecuencia que le permite enviar y/o recibir datos
- Transponder o dispositivo cliente, instalado en el elemento a localizar (mediante una etiqueta o tag)

La localización mediante redes locales puede llevarse a cabo de diferentes maneras que se exponen a continuación:

- ❖ La más sencilla es la basada únicamente en el punto de acceso más cercano al terminal. Este método, utilizado en localización en recintos urbanos, confunde a menudo la planta del edificio. Pues es fácil que la antena más cercana a un usuario ubicado en una determinada planta sea la misma que la correspondiente a un usuario situado en una planta superior, si la posición sobre el piso es similar. Por otra parte, la señal es vulnerable debido a las interferencias, lo que puede afectar, además de a la precisión de la localización, a la seguridad de una hipotética comunicación.

- ❖ Otro método de localización es TDOA (Time Difference Of Arrival) basado en técnicas de triangulación que emplea la diferencia entre los tiempos de llegada de la señal procedente del terminal móvil (tag) a distintos pares de estaciones base, es decir los dispositivos readers, para calcular la posición.

- ❖ Los métodos más vanguardistas de localización se basan en motores de posicionamiento que almacenan la medida de potencia de señal en diferentes puntos del área de cobertura. La técnica, conocida como *WiFi mapping*, arroja resultados más exactos que los métodos de triangulación celular.

[5] [4]

La frecuencia en la que trabaje el servicio es de gran importancia pues a menor frecuencia el sistema tendrá mayor alcance, pero menor precisión. En cambio, a mayor frecuencia se requirá de más dispositivos readers, pero el sistema dispondrá de una mayor precisión.

En una residencia o un psiquiátrico es de gran utilidad monitorizar la posición de los pacientes en todo momento. No solo se limita a la localización de personas, en un hospital además de los pacientes también es necesario saber donde están los equipos (que se utilizan para más de un solo paciente) rápidamente (como por ejemplo un sistema AED[4]).

2.1.2. Uso de RTLS en la industria ganadera

En ganadería se utiliza un tipo específico de RTLS llamado RFID [6] [7]. La diferencia ente estas dos versiones reside en resumen en el alcance. En RFID (Radio Frecuency IDentification) las etiquetas (tags) se leen a medida que pasan por los puntos fijos en un proceso estructurado, mientras que con RTLS las etiquetas se leen de forma automática y continua, independiente del proceso que mueve las etiquetas. Podemos resumirlo de manera que con RTLS no se necesita ninguna intervención o proceso controlado para determinar la localización de las etiquetas.

[4] Automatic External Defibrillator

Es una herramienta útil para el seguimiento de los animales ya que una vez etiquetado, escaneado y registrado, un animal puede ser monitoreado tanto en sus movimientos como en acciones de comederos lo que ayuda a identificar y obtener datos y estadísticas sobre los animales. Las etiquetas se pueden utilizar ya sea en placas de identificación para las orejas como en otros formatos incluso en modo de ingestión del animal y subcutáneo.

Ilustración 1: uso de RTLS en ganadería

El uso de etiquetas RFID con el ganado mejora el seguimiento y la automatización para el agricultor, las empresas de exportación de alimentos, casas de subastas de animales y las instituciones gubernamentales, lo que permite a todas las partes identificar a un animal, demostrar su propiedad y evitar el fraude.

2.1.3. Uso de RTLS en Salud

Desde hace ya unos años el sector hospitalario está realizando una apuesta por la tecnología de localización de elementos en tiempo real (RTLS) aplicada a la gestión de personal médico, pacientes y equipos [8]. Especialmente en países como Estados Unidos y Reino Unido se están realizando despliegues que dependiendo del hospital pueden cubrir la localización del personal médico, pacientes, historiales clínicos, así como equipos y dispositivos de mano.

La tecnología aplicada, incluye sistemas RFID activos y los basados en el estándar 802.11. El posicionamiento puede llevarse a cabo por diversas técnicas, como la triangulación en potencia, medida del vector potencia y técnicas heurísticas[5]. Muchos métodos utilizan multilaterización, midiendo el tiempo de vuelo de la señal, diferencia del tiempo de vuelo de la señal, medida del ángulo de llegada, medida de RSSI y otros [9].

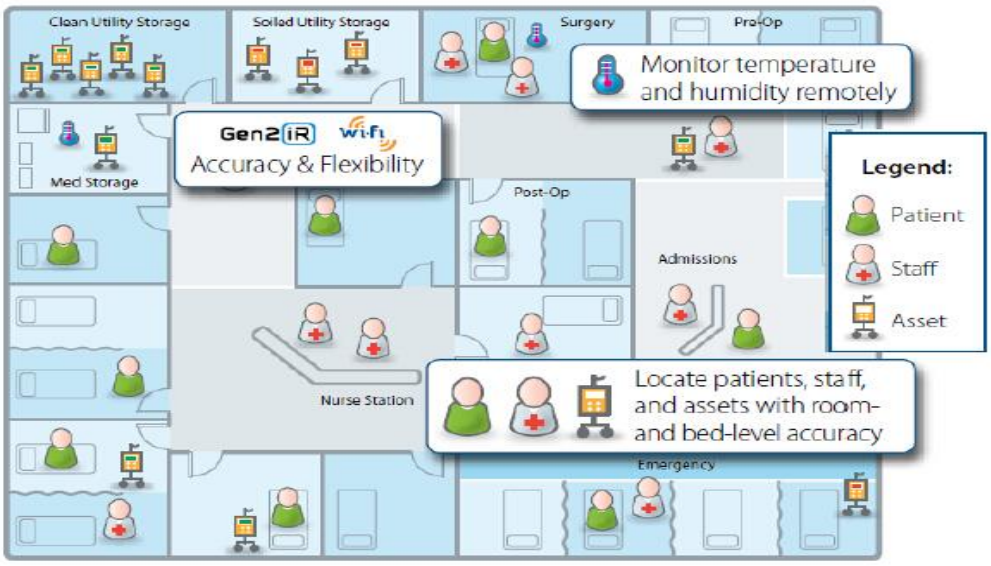

Ilustración 2: uso de RTLS en salud

La introducción de tecnología de banda ultra ancha (UWB, Ultra Wide-Band) puede convertirse en un futuro en la base de desarrollos de sistemas de mejor precisión, bajo coste y menor consumo. Si bien en el mercado americano se está realizando ya algún despliegue experimental, en Europa tendremos que esperar a que el regulador europeo de las comunicaciones resuelva un contencioso que tiene que

[5] Manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

ver con las tecnologías americanas y su viabilidad de uso en el Viejo Continente.

2.2. Soluciones Comerciales

Se muestran a continuación soluciones comerciales al problema del seguimiento y ejemplos de uso de la visión artificial para lograr seguimientos in situ.

A diferencia de los métodos anteriores el uso que se le da a RTLS a continuación, está destinado exclusivamente a seguimiento de objetos para enmarcarlos en un video de forma automática. Era necesaria una introducción sobre en que consistía la tecnología y ahora mostramos las aplicaciones de ésta en el ámbito que nos interesa.

2.1.1 SOLOSHOT3

SOLOSHOT3 es un sistema autónomo que consiste en una cámara de video sobre una base móvil la cual panea, enfoca y hace zoom si es necesario con el objetivo de para mantener encuadrado en todo momento al objeto [10].

¿Cómo consigue esto? Pues el sujeto que quiere ser encuadrado necesita llevar un tag, una especie de pulsera, que se conecta con la base mediante un link al que ellos llaman SS Link. De este modo siempre que se lleve la pulsera el sujeto en cuestión será grabado, presumen de ser capaz de funcionar a una distancia de hasta 600 metros.

El CEO Chris Boyle, aficionado al surf, fundó esta empresa buscando un objetivo claro que era el de mejorar su técnica deportiva viendo en video sus acrobacias sobre la tabla de surf, para ello debía grabar sus sesiones y no siempre podía disponer de un compañero grabándole desde la orilla. Es un ejemplo a modo de anécdota de cómo la necesidad individual ha creado un dispositivo autónomo que sustituye a un operario de cámara o en este caso a un "amigo".

Ilustración 3: dispositivo SoloShot3

De las 3 opciones mencionadas anteriormente en el apartado de la tecnología RTLS, podemos aventurarnos[6] a afirmar que estamos ante la primera opción (punto de acceso mas cercano al terminal) ya que el sistema solo se compone de un tag y un reader. Este dispositivo posee un gran alcance debido a que utiliza una frecuencia baja, por lo tanto, no posee gran precisión, pero al ser una cámara con un encuadre tan amplio no es un gran inconveniente. Si quisiéramos que la cámara encuadrara únicamente al surfista en en primer plano este dispositivo no nos valdría.

[6] Puesto que los fabricantes no proporcionan información de sus métodos de seguimiento o algoritmos.

2.1.2 AIMe

Ilustración 4: AIMe con una cámara de video convencional

Este producto de Jigabot® funciona de modo bastante parecido al ejemplo anterior solo que esta solución nos permite acoplar una cámara o un móvil al sistema de seguimiento, lo cual es muy práctico pues no nos limita a la hora de elegir una cámara u otra, en el caso anterior la cámara es fija y viene de serie con el dispositivo [11].

Además de una cámara de video convencional, nos da la opción de ajustar una cámara de fotos, una cámara gran angular o incluso un Smartphone. Es un sitema resistente tanto al agua como al polvo por lo que nos proponen en su publicidad utilizarlo tanto en tierra como en un barco.

Esta empresa tiene a la venta también una versión "pro" del AIMe, el AIMe Pro. Este es un sistema mucho mas interesante pues utiliza varios tags y varios readers, gracias a esto la localización no solo es más precisa, si no que también da pie a crear algoritmos que incluyan de un modo u otros varios tags a la vez a la hora de producir el encuadre.

2.1.3 Pixio

De los tres sistemas que presento, este es el más conocido. Este dispositivo funciona como la versión pro del anterior y también es posible acoplarle una cámara de terceros. Es un sistema diferente al anterior por la sencilla razón de que cada sistema tiene un algoritmo distinto, no es una razón de peso trivial, ya que ésta es una empresa con mucha mas experiencia en el sector y con elementos de mayor calidad.

Esta marca también dispone de una versión pro, la cual tiene algo nuevo que no habíamos visto hasta el momento, el tag se localiza, adicionalmente, mediante GPS lo cual implica una mayor precisión y además de que solo podrá ser utilizado en exteriores. Esta versión pro incluye un tag de gran tamaño con controles del sistema de seguimiento a distancia, es decir podremos indicarle al dispositivo en el momento que nos está grabando cómo queremos que lo haga [12].

Ilustración 5: Detalle pixio y sus elementos

2.3. Mimicking Human Camera Operators

Este paper realizado por *Jianhui Chen y Peter Carr* nos muestra un algoritmo de seguimiento de la acción en un evento deportivo que resulta ser un partido de baloncesto [13]. Este algoritmo utiliza como datos la posición precisa de todos los jugadores del campo, y a partir de estas posiciones el algoritmo genera un paneo inteligente que nos muestra la acción, es decir, lo interesante en un partido de baloncesto. Lo que logra este algoritmo propuesto es sustituir al operador de cámara añadiendo además un toque humano en la grabación

con el fin de que no parezca un movimiento robótico.

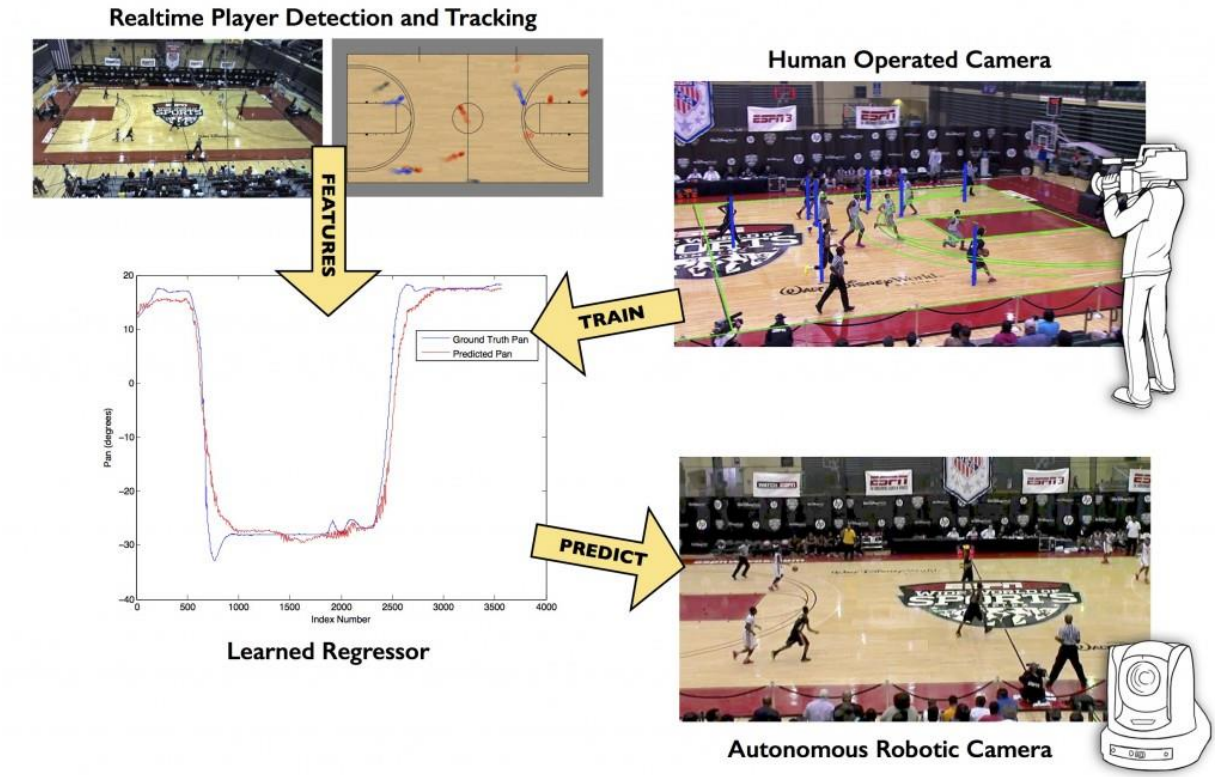

Ilustración 6: Resumen del objetivo de Disney Research

Partir de todos los jugadores localizados es tener la mitad de todo el problema resuelto, si no fuera por el hecho de que para localizar a los jugadores hace falta de algún modo (ya sea con una cámara en el techo del pabellón o en un lateral) una cámara fija que aloje a todos los jugadores ya habríamos llegado a la solución del problema original.

No podemos pedirle a todos lo jugadores que se pongan un brazalete que incluya un tag para que un sistema combinado de RLTS y el algoritmo (que nos presenta el paper) sean la solución al problema, además uno de los objetivos que estamos buscando con esta investigación es un sistema que sea sencillo, útil y versátil. Por ello, una fase de preparación que incluya colocar los readers y los tags por el hecho de sustituir al operario de cámara, aunque interesante, pienso que no saldría rentable a un nivel de competiciones inferiores y no solo rentable refiriéndonos a aspectos económicos sino también a la inversión de tiempo y esfuerzo.

En definitiva, este paper no nos es práctico en cuanto a solución definitiva, aunque introduce algunos conceptos interesantes como la idea de evitar un paneo demasiado robótico o el uso de los centroides para referirse a los jugadores.

Más adelante, en la explicación de los algoritmos, veremos como adaptamos estas ideas por el momento abstractas.

3 SISTEMA DE SEGUIMIENTO DE LA ACCIÓN

No basta saber, se debe también aplicar. No es suficiente querer, se debe también hacer.

- Goethe -

Se presenta un método para llevar a cabo el seguimiento de la acción en los eventos deportivos en el ámbito del baloncesto. Es importante hacer incapie en el tipo de deporte ya que tendrá unas peliculiaridades inherentes que nos ayudarán a crear un algoritmo más inteligente.

Como se mencionó previamente se ha utilizado el lenguaje de programación Matlab® ya que es más cómodo hacer experimentos con él, de este modo habrá una base estudiada para un futuro proyecto[7] destinado a implementar estas idéas en un código de menor nivel que de la opción de trabajar en tiempo real, como por ejemplo el lenguaje de programación *C*.

Se han utilizado 4 vídeos para ilustrar este método, éstos se han adquirido, en su mayoría, en competiciones provinciales de la comunidad autónoma de Andalucía. Se ha pretendido que la grabación abarque todo el campo para que el producto del programa sea un cuadro que delimite la zona de interés, a pesar de ello en muchas ocasiones no se ha podido grabar dicho marco ya que se necesitaría de una cámara de gran angular o situarse en la grada muy lejos del campo, en la mayoria de ocasiones esa opción era irrealizable ya que la distancia a la que habría que colocarse excedería las propias dimensiones del pabellón.

3.1. Videos utilizados

Como ya se ha dicho, se han utilizado 4 videos para ilustrar el algoritmo, paso a describirlos para que quede constancia de cuando y como se obtuvieron:

- Video 1: Es un fragmento de un encuentro de baloncesto en el que la cámara está fija orientada únicamente a la mitad del campo. Fue grabado el 8.12.15 en la localidad sevillana de Coria del Río, el encuentro enfrenta a CD Coria contra CD Villaverde.
- Video 2: Es un fragmento de un encuentro de baloncesto en la que la cámara está fija abarcando gran parte del campo, la peculiaridad de esta grabación es una mala elección de la situación de la cámara ya que no era posible colocarla en otro lugar por limitación de espacio en el pabellón. Debido a este emplazamiento, en la grabación también aparece el público entre la cámara y la pista. Fue grabado el 21.05.16 en la localidad sevillana de Mairena del Alcor, el encuentro enfrenta a CD Los Alcores contra CD Guillena.

[7] Que incluya tanto el algoritmo como los dispositivos mecánicos encargados del paneo a nivel físico.

- Video 3: Es un fragmento de un encuentro de baloncesto en el que la cámara está sujeta a pulso, por lo que tiene algunas vibraciones que complican el funcionamiento del algoritmo. En este caso si es grabado el campo completo puesto que la estructura de este pabellón permitía situar la cámara muy lejos del campo. Fue grabado el 7.7.16 en la localidad de Las Palmas de Gran Canaria, el encuentro enfrenta a dos equipos participantes en el Showcase CIBA 2016 [14].
- Video 4: Es un fragmento de un encuentro de baloncesto en el que la cámara utilizada es una cámara gran angular, gracias a esto se aprecia sobradamente toda la pista, el problema de esta grabación es la ínfima calidad de dicha cámara que hace que se aprecie el partido con cierta dificultad. Fue grabado el 11.8.16 en la localidad onubense de Punta Umbría, el encuentro enfrenta a dos equipos de la liga de verano 2016 organizada por FabHuelva.

3.2. Algoritmo de seguimiento de la acción

Para hacer frente a los distintos problemas que van sucediedo he dividido el código en siete partes con el fin de facilitar la comprensión y realización. A pesar de esta división por partes al final del documento se incluirá el código completo.

Este algoritmo es el resultado final de multitud de pruebas realizadas con los 4 videos, en las que se ha procurado llegar a procesado óptimo dentro de este primer sistema. Se verán la gran mayoría de ellas y por qué se ha decidido en cada caso la opción adoptada.

3.2.1 Inicialización

A modo de introdución del código:

```
clc
clear all
close all
```

Estas funciones de Matlab limpian la ventana de comandos, eliminan todas las variables y cierra todas las ventanas que pudiera haber, respectivamente.

En este apartado se definirán también todas las vaiables globales que utilizará el programa. Habrá también algunas que deberemos modificar para obtener el resultado buscado. Dependiendo del algoritmo que utilicemos[8] estas variables serán distintas. Pueden verse detalladamente en el punto 5 de este documento (Códigos).

3.2.2 Lectura del video

En este apartado, Matlab realiza la lectura del video en cuestión, la función `VideoReader` crea un objeto[9] multimedia exclusivamente de lectura, esto implica que tras esta definición ya sabemos todo lo relativo al video que se ha introducido como por ejemplo la longitud, el numero de cuadros por segundo, el tamaño, el formato de video, la tasa de bits por segundo etc.

[8] Más adelante veremos en que punto los métodos se bifurcan.
[9] En Matlab se pueden crear objetos de datos para especificar valores, rangos de valores, tipos de datos, afinabilidad y otras características de señales, estados y parámetros de bloque.

```
video =

  VideoReader with properties:

   General Properties:
           Name: 'ciba.avi'
           Path: '/Users/javi/Desktop/Trabajo/videos'
       Duration: 69.2000
    CurrentTime: 26.7000
            Tag: ''
       UserData: []

   Video Properties:
          Width: 1920
         Height: 481
      FrameRate: 30
   BitsPerPixel: 24
    VideoFormat: 'RGB24'
```

Ilustración 7: Ejemplo de dato objeto

Más adelante habrá que utilizar la función `read` para acceder a un fotograma concreto del objeto creado anteriormente, esta función es una subfunción de `videoReader` es decir, que no funcionaría si no introducimos antes dicha función.

`videoReader` tiene multitud de funciones por ejemplo `videoReader.FrameRate`, `videoReader.BitsPerPixel` o la que utilizaremos principalmente que es `VideoReader.read`

```
video= VideoReader('ciba1.mp4');
```

Todos los videos tendrán un numero de cuadros por segundo igual a 30, algunos de ellos se han grabado en 25 cuadros por segundo por lo que al pasarlo a 30 cuadros por segundo han sufrido un efecto de aceleración que no nos afecta en absoluto en nuestra meta. Es decir, todos los videos tienen 30 cuadros por segundo.

El bucle for lo utilizaremos para recorrer todos los cuadros del video uno a uno y detectar movimiento como se explica más adelante. Dentro del bucle introduciremos la función de lectura de los cuadros que será `read`. Además, para facilitar el estudio y como el video original era demasiado grande se ha realizado un paso de reducción del cuadro con la función resize al 50% de este modo el cálculo es más rápido y no afecta al resultado final más que en el tamaño del cuadro final, es simplemente una facilidad mas para trabajar con Matlab.

En el caso de implementarse este algoritmo en un código más veloz habría que hacer un estudio de cómo afecta el tamaño del cuadro al tiempo de trabajo del algoritmo. Éste reescalado se realiza con la función `imresize` de Matlab.

```
for a=cuadroInicial:cuadroFinal

I01=imresize(read(video,a),0.5);
I02=imresize(read(video,a-1),0.5);
```

Como puede apreciarse, `I01` y `I02` serán dos cuadros en color de tipo uint8 (se explicará mas adelante). El bucle `for` acabará en el valor proporcionado por `cuadroFinal` con saltos unitarios, es decir, se compará un número determinado de cuadros consecutivos. Para agilizar el proceso y con el fin realizar comprobaciones sin eternizar el tiempo de ejecución no se ha querido que el algoritmo lea la totalidad del video.

Se entiende que `a=2` reperesenta el segundo cuadro del video, si no se desea empezar por el segundo cuadro simplemente habrá que cambiar ese valor al que se busque o bien modificar la estructura creada por `VideoReader` para que esta función empiece a leer el video desde el cuadro que se requiera.

Se utiliza el segundo cuadro y no el primero ya que estamos buscando un algoritmo que trabaje a tiempo real, si empezaramos en el primer cuadro y buscaramos una imagen diferencia a partir de la resta del segundo cuadro

con el primero, estaríamos accediendo a una información que aun no ha ocurrido. En la simulación si puede hacerse ya que el video está definido en el objeto creado.

3.2.3 Transformación de la imagen a imagen diferencia

Para realizar este paso introducimos los tipos de imágenes que vamos a utilizar en código, éstas son:

- Uint8: Es un formato de imagen en el que los píxeles están representados por números del 1 al 255 en función de su intensidad, representando el 1 la intensidad más baja o negro y el 255 la más alta o blanco, por lo tanto, este formato nos dará una imagen en escala de grises. Ahora bien, con este formato se puede dar lugar a imágenes en color siendo *uint8* una combinación de 3 matrices[10] representando 3 colores: rojo, verde y azul. En este código utilizamos ambos sistemas de representación de uint8.
- Double: Es un formato de imagen en el que los píxeles, a diferencia del formato anterior, están representados por valores comprendidos entre 0 y 1, siendo el 0 el negro y 1 el blanco. Como en el caso anterior este formato admite imágenes a color estando la variable *double* compuesta de 3 matrices cada una representando un color (rojo, verde y azul). En este código solo utilizaremos el primer caso del formato double.

Lo primero será convertir la imagen obtenida mediante el procedimiento anterior, que está en formato *uint8* en color, a formato *unit8* en blanco y negro (pasar de 3 matrices a una sola matriz). Posteriormente convertimos esta imagen a formato *double* para poder realizar operaciones más tarde.

Es importante recalcar que con el formato *double*, a diferencia del formato *uint8*, podemos realizar operaciones con los píxeles sin los inconvenientes que uint8 pudiera causar, como por ejemplo el problema de losvalores negativos.

Al utilizar el formato *uint8*, éste convertirá todos los valores negativos en cero. Por ejemplo: al restar una imagen en formato *uint8* por un valor numérico entero (100 por ejemplo) el resultado podría ser una matriz con resultados tanto negativos como positivos, el formato uint8 hace que todos los valores negativos tengan el valor cero. Esta peculiaridad es lo que nos fuerza a trabajar en formato *double* para restar matrices[11] sin encontrarnos errores. Al hacer esta misma operación con el formato double el resultado será una matriz con vaalores negativos y positivos.

El código será el siguiente:

```
I1=rgb2gray(I01);
I1=im2double(I1);

I2=rgb2gray(I02);
I2=im2double(I2);

I3=I1-I2;
```

Tanto `I2` como `I1` representan dos cuadros distintos del video original, éstos están en formato *double* en blanco y negro por lo que I3 (la resta de ambos) será la imagen diferencia y podrá tener valores positivos y negativos conprendidos entre -1 y 1.

Al haber restado la segunda imagen menos la primera, los valores positivos, si los hay, serán las zonas a las que se desplazan los objetos (si hay desplazamiento) y las zonas negativas serán aquellas de las que provienen los

[10] De valores [1-255] cada una
[11] Imágenes

objetos. Este es un dato importante para pasos posteriores.

Podemos ver un ejemplo en la siguente ilustración[12] en la que el color rojo representa hacia donde se mueve el sujeto y el color verde del donde viene.

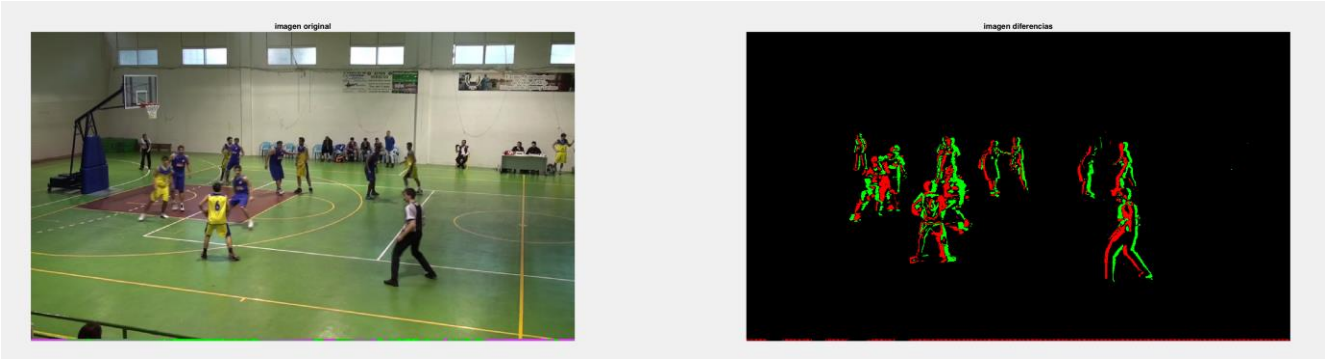

Ilustración 8: Imagen diferencias con de detalle colores

3.2.4 Transformación de la imagen a imagen binaria

Una imagen binaria será aquella cuyos píxeles solo puedan tomar valores de uno y cero.

Habrá que encontrar pues, un umbral que difiera los valores de la imagen diferencia. Este umbral deberá suponer una variación sufientemente notable para considerarla un movimiento. En un principio se ajustó el umbral mediante un número arbitrario encontrado tras un proceso de prueba-error para que el programa funcionara correctamente, pero al utilizar un video diferente, este umbral ya no era útil por lo que finalmente se decidió realizar este paso de forma automática sin importar el video que le introduzcamos.

Para llevar a cabo esta elección, el método elegido es el método de Otsu

3.2.4.1 Método de Otsu

El método de Otsu, llamado así en honor a Nobuyuki Otsu que lo inventó en 1979, utiliza técnicas estadísticas, para resolver el problema. En concreto, se utiliza la varianza, que es una medida de la dispersión de valores. El método de Otsu calcula el valor umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Para ello se calcula el cociente entre ambas variancias y se busca un valor umbral para el que este cociente sea máximo.

En definitiva, realiza, apoyándose en el histograma de la imagen diferencia, una separación de regiones encontrando el punto en el que las dos agrupaciones de pixeles mas parecidos se encuentran.

En esta imagen podemos ver, a modo de ejemplo, el umbral hallado por el método de Otsu en una imagen con formato *uint8*.

[12] El código que realiza esta representación aparece al final de este documento en la sección de códigos.

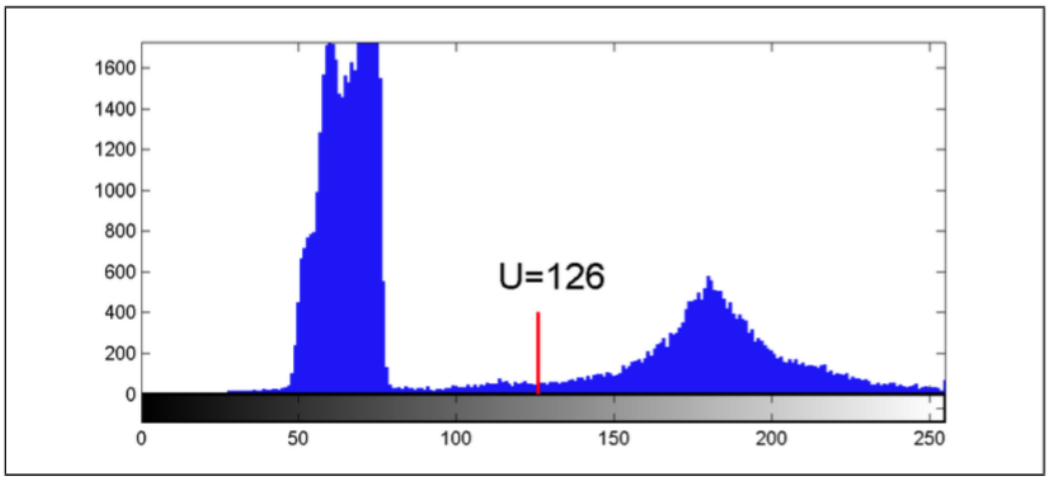

Ilustración 9: Ejemplo poblaciones para el método de Otsu

Se puede apreciar las dos agrupaciones de valores en el histograma y el valor hallado automáticamente por el método de otsu.

El código será el siguiente:

```
umbral=graythresh(I3);
I3(abs(I3)>umbral)=1;
I3(I3~=1)=0;
```

Una vez tenemos el umbral que vamos a utilizar se realiza la conversión a una imagen binaria. Como se aprecia, todos los valores absulutos de I3 por encima del umbral tendrán el valor uno y una vez realizado este paso todos los valores que no sean ya uno, se les asignará el valor 0. De este modo llegamos a la imagen binaria.

Esto se ha implementado de este modo no importándonos exclusivamente hacia donde va un objeto (valores positivos) si no tambien de donde proviene (valores negativos). Esto no es totalmente necesario; para detectar un movimiento podríamos haber despreciado los valores negativos, pero tras varias pruebas se llega a la conclusión de que para evitar que movimientos de la cámara o del público interfirieran en el objetivo es necesario que lo que se movía notablemente tuviera mayor peso en la imagen binaria.

En este ejemplo podemos observar una imagen direfencias teniendo en cuenta solo los valores positivos y otra teniendo en cuenta valores tanto positivos como negativos.

Ilustración 10: Imagen diferencia

Esto se entendrá mejor en el apartado siguiente que trata sobre las operaciones morfológicas.

3.2.5 Operaciones Morfológicas

Este es, de lejos, el paso más importante del algoritmo de seguimiento, pues consiste en realzar en la imagen binaria los movimientos que realmente son importantes y descartar los ruidos o los que suponen una lacra para el buen funcionamiento del sistema. Los objetivos de este paso son eliminar el ruido y realzar los movimientos útiles.

Matlab ofrece gran variedad de operaciones morfológicas sobre imágenes binarias mediante la función `bwmorph`. Es tan versátil que es necesario dedicar unas líneas a estudiar y exponer sus posibilidades:

3.2.5.1 bwmorph

Funciona del siguiente modo:

```
I3=bwmorph(I1,'option',n);
```

Siendo `I1` la imagen binaria, `option` un tipo de operación morfológica y `n` el numero de iteraciones de esa operación (al introducir el valor `inf` en `n` realizará la operación indefinidamente hasta que la imagen no cambie. A continuación, se muestran los diferentes valores que puede tomar `option` y una breve descripción.

- Bothat: una acreción seguida de una erosión.
- Branchpoint: encuentra los puntos de ramificación.
- Bridge: conecta pixeles (con valores uno) separados (por valores cero)
- Clean: elimina pixeles solitarios.
- Diag: rellena las diagonales.
- Dilate: realiza una acreción (una acreción consiste en aumentar la población de unos siguiendo el patrón proporcionado por una plantila) con plantilla 3x3.
- Erode: realiza una erosión (una erosión consiste en disminuir la población de unos siguiendo el patrón proporcionado por una plantila) con plantilla 3x3. Para variar el tamaño de la plantilla hay que usar la función `imerode` y crear una plantilla mediante la función `strel`, más adelante se montrará como se hace.
- Fill: convierte a uno pixeles aislados (rodeado de unos).
- Majority: convierte el píxel a uno si 5 o más píxeles de la ventana 3x3 son uno, a cero en el caso contrario.
- Open: una erosión seguida de una acreción.
- Shrink: reduce conjunto de unos a un solo pixel
- Spur: elimina los espolones de agrupaciones de unos.
- Thicken: agranda agrupaciones de unos haciéndolas mas gruesas
- Thin: caso opuesto al anterior
- Tophat: realiza una erosión seguida de una acreción.

Practicamente las opciones son infinitas debido a la gran cantidad de operaciones morfológicas que incluye Matlab. La conclusión mas importante a la que se llega tras varias pruebas es que no se puede encontrar una solución que satisfaga todos los casos, pues habrá grabaciones con más luz, menos luz, más movimientos de

fondo … etc.

Una posible solución podría ser, establecer una serie de condiciones en cuanto a cómo debe realizarse la grabación del evento deportivo como por ejemplo únicamente en pabellones de parquet en la que la cámara esté a una distancia determinada y con una iluminación específica, el problema que conlleva esta solución es que el sistema perdería la capacidad de adaptarse a cualquier entorno que se le quiere conceder.

La otra solución es un análisis morfológico que no será óptimo para todas las situaciones pero que darán un resultado muy cercano a la realidad. En definitia perderíamos precisión, pero al mismo tiempo se gana versatilidad

Así que, se desarrolla a continuación una combinación de operaciones que nos sirven para la mayoría de los casos y conducen a relativamente pocos errores. Acontinuación se muestran el código con las operaciones elegida y una muestra de los cambios que sufre una única imagen diferencias a lo largo de las transformaciones.

```
se=strel('square',2);


I4=imerode(I3,se);
I4=bwmorph(I4,'clean',inf);

I4=imerode(I4,se);
I4=bwmorph(I4,'clean',inf);

I4=bwmorph(I4,'dilate',1);
I4=bwmorph(I4,'clean',inf);
```

Lo primero ha sido crear una plantilla de 2x2 mediante la función `strel` con el atributo `'square'` que hace que la plantilla sea cuadrada (ya que así se controla mejor las operaciones al no tener una plantilla que con una dimensión tan pequeña tendría picos), luego se procede a aplicar la operación de erosión con *imerode*.

Pueden distinguirse 4 partes:

- Creación de una plantilla
- Erosión más limpieza
- Erosión más limpieza
- Y por último una acreción con una plantilla 3x3.

De este modo conseguimos eliminar casi todo el ruido, es decir el movimiento que no nos interesa, y establecer una imagen binaria más limpia para que el apartado que sigue a este, encargado de la obtención del centroide, funcione correctamente.

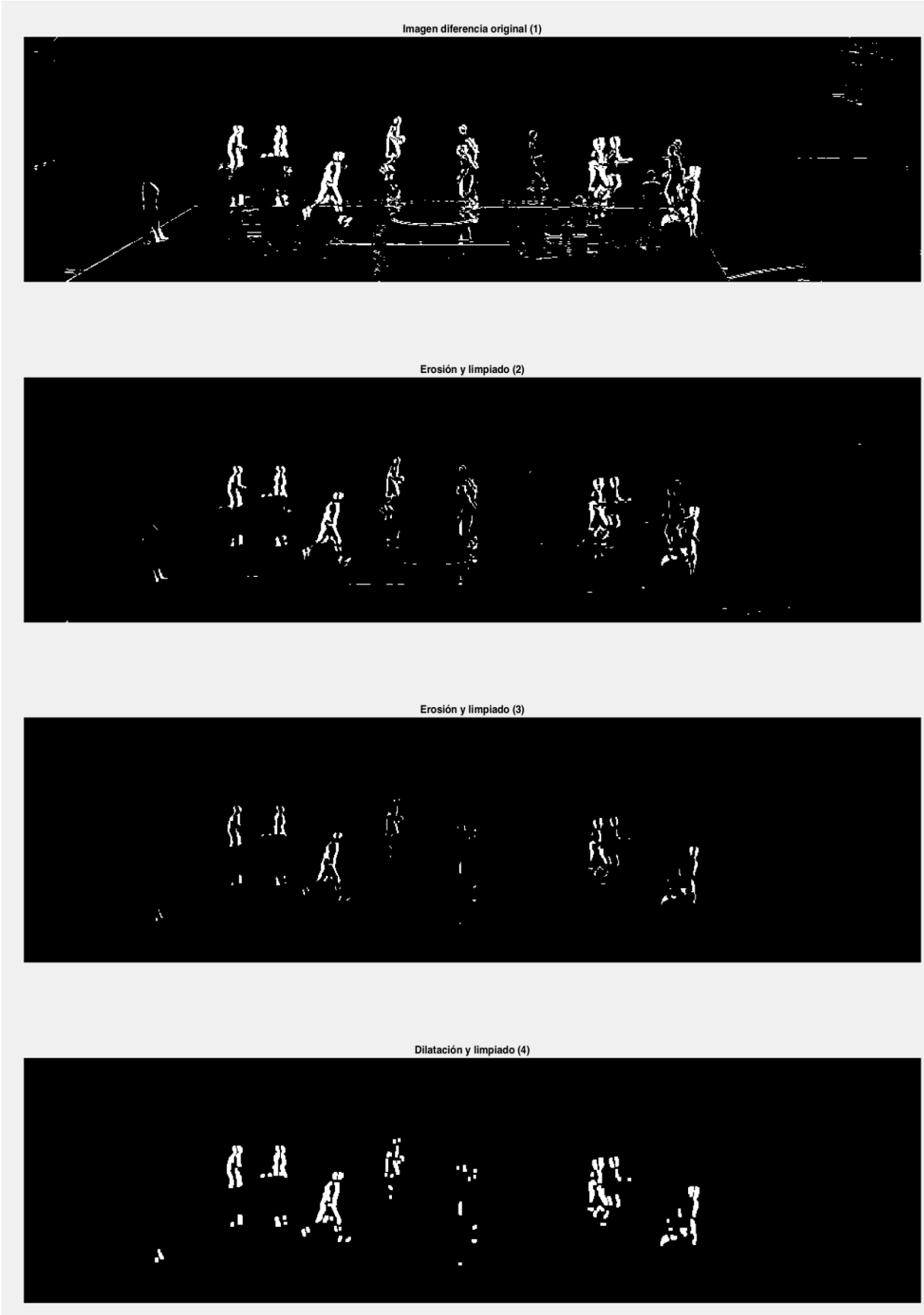

Ilustración 11: Operaciones mofológicas

3.2.6 Obtención del centroide

La razón por la que el paso anterior es tan importante lo vemos en este apartado. Una vez tenemos bien definida la imagen diferencias con la información que queremos realzada, el siguiente paso es encontrar el centroide de la imagen.

Llegados a este punto, nos encontramos ante **dos posibilidades** que arrojarán desultados distintos: calcular el centroide de la totalidad de la imagen diferencias o calcular el centroide del encuadre en cada momento. En el primer caso el cálculo del centroide conllevará un mayor coste computacional además de que solo será factible si tenemos en todo momento un encuadre de imagen que abarque la totalidad del campo. En cambio, el segundo caso nos dará una solución más adecuada a la solución óptima que estamos buscando, aunque tendrá una serie de inconvenientes como el aumento de las líneas de código y la necesidad de definir unas dimensiones del cuadro.

La búsqueda del centroide se realiza con la función de Matlab `regionprops` que mide las propiedades de las regiones de la imagen, en este caso la propiedad que nos interesa es el centroide puesto que los lugares donde haya movimiento serán las zonas en las que el valor de la imagen binaria es uno.

El código para el primer caso es:

```
  s=regionprops(I4,'centroid');

  %   sirve para evitar las imagenes en la que todos los valores son
  %   cero (que solo son obtenibles mediante operaciones morfológicas)

  if 0==isempty(cat(1,s.Centroid))% si está vacío 0==1 por lo que no entra en
el bucle
   cent=cat(1,s.Centroid);
  end

  centx=mean(cent(:,1)); %Punto centroide X
  centy=mean(cent(:,2)); %Punto centroide Y
```

Y el código para el segundo caso de los previamente mencionados es el siguiente:

```
[Filas,Columnas]=size(I4);

if centx==0

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Al tener una grabación que abarca todo el campo hacemos lo siguiente

    ancho=tand(theta/2)*distancia;  %Ancho del cuadrado de encuadre
    ancho=round((ancho*Columnas)/15);  % 15 es mitad de 30 metros que es lo
que mide el campo completo. Round para conseguir un valor de "ancho" entero

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    alto=Filas;   %Alto del cuadrado de encuadre
    ejeY=alto/2;

    s=regionprops(I4,'centroid');

else

    s=regionprops(I4(:, ceil(centx-ancho/2):ceil(centx+ancho/2)),'centroid');
```

```
end

    %Sirve para evitar las imagenes en la que todos los valores son
    %cero (solo son obtenibles mediante operaciones morfológicas)

if 0==isempty(cat(1,s.Centroid))% si está vacío 0==1 por lo que no entra en
el bucle
    cent=cat(1,s.Centroid);


    if centx==0 %Primera iteración
        centx=mean(cent(:,1)); %Punto centroide X

    else
        centx=centx+(mean(cent(:,1))-(ancho/2)); %Punto centroide X

        %Comprobaciones para evitar que el cuadro se salga de los límites
        if (centx-ancho/2)<0
            centx=(ancho/2)+1;
        end
        if (centx+ancho/2)>Columnas
            centx=Columnas-(ancho/2);
        end

    end
end
```

Con el objetivo de inicializar el programa, veremos que se aborda la primera iteración comprobando que `centx=0`, en cambio si estamos en las siguientes haremos que la función `regionprops` solo se aplique al tamaño del cuadro que se haya elegido. Además, en la iteración se realizan cálculos para determinar el tamaño del cuadro[13]

Debido a que puede ocurrir que el centroide no se calcule correctamente, ya sea por que las dos imágenes sean iguales o por algún error de computación, se implementa una medida para que el código siga funcionando. Consiste simplemente en mantener el centroide la imagen anterior o el valor cero si el error ocurre en la primera iteración (no entrar en el bucle).

Por último, para ir modificando la posición del centroide en la propia imagen `I4` nos encontramos con un problema: al utilizar solamente el tamaño del cuadro que hemos seleccionado, reducimos coste computacional pero el centroide que se nos devuelve está escalado al tamaño de dicho cuadro. Para ello sirven las últimas líneas de este fragmento del código que, además, añade una comprobación para evitar el cuadro se salga de los límites del video.

Podría asaltarnos la pregunta de porqué se se ha complicado el problema haciendo que calcule el centroide de únicamente el cuadrado ¿no sería más sencillo calcular el centroide de la imagen I4 en su totalidad? De hecho, si lo sería, pero estamos buscando un sistema que pueda ser aplicado al paneo de una cámara donde **no** tenemos la imagen de todo el campo si no solamente el tamaño del cuadro[14], de este modo hacemos más versátil el algoritmo. Veremos los resultados que arroja cada caso: centroide calculado a partir de una imagen que abarca todo el campo llamado a partit de ahora **"completo"** y centroide calculado a partir de una sección de la imagen denominado **"parcial"**[15].

[13] Explicado extensamente en el apartado 3.2.10
[14] El encuadre en este supuesto
[15] Es una denominación que nos facilitará la lectura en las páginas siguientes

3.2.7 Representación de las componentes del centroide completo

Se ha reprentado la concatenación de los centroides de los primeros 1000 cuadros y el resultado es el siguiente.

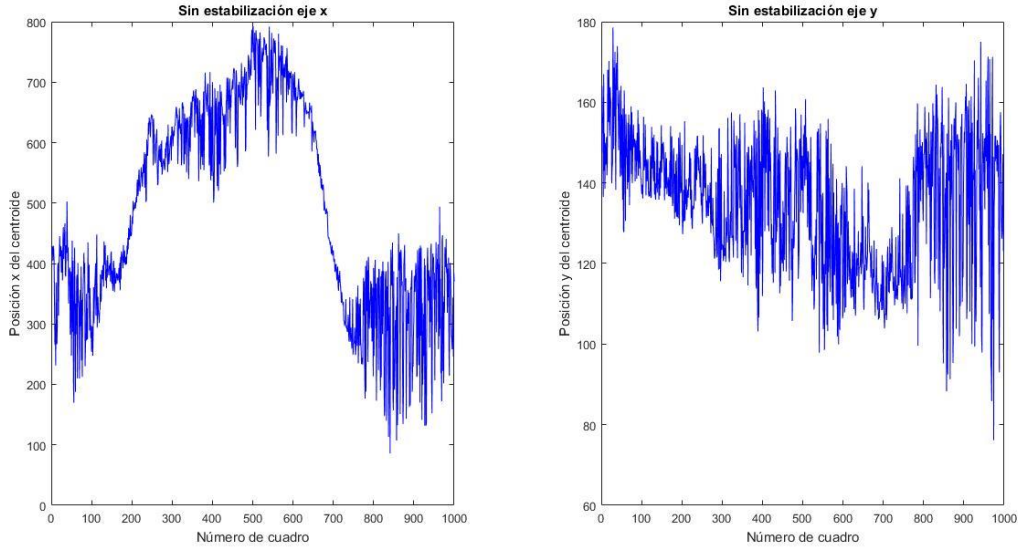

Ilustración 12: Centroide a lo largo de 1000 iteraciones caso completo

La primera gráfica representa la posición de la componente 'x' del centroide en el cuadro a lo largo de 1000 cuadros. En ella podemos ver 3 zonas suficientemente diferenciadas, que en el caso que estamos analizando (un partido de baloncesto), estas zonas representan: la primera es la primera mitad del campo (lado izquierdo) la segunda es el lado derecho y la tercera es de nuevo el izquierdo.

La segunda gráfica representa la posición de la componente 'y' del centroide en el cuadro a lo largo de 1000 cuadros, al igual que la primera gráfica. En esta ocasión nos es más complicado ver zonas claramente diferenciadas, esto es porque durante la grabación de un partido muy raramente se produce un tilteo de la cámara. Por lo tanto, una conclusión directa de este hecho es que esta gráfica debería ser lo mas pacedido posible a una línea horizontal para logar el resultado buscado.

Es muy importante tener en cuenta que la estabilización no la podemos realizar a posteriori puesto que el valor del centroide tiene que ir cambiando durante la grabación por lo que los vectores de datos que generan las gráficas no son conocidos. Esto quiere decir que deberemos trabajar con los valores en tiempo real.

3.2.8 Estabilización de la componente 'y' del centroide completo

En primer lugar, vamos a ver como podemos estabilizar la componente 'y' del centroide.

Como vemos en la segunda gráfica de la ilustración 12, este valor se mueve en torno a los valores 160-120 y puesto que buscamos lo mas parecido a una línea horizontal vamos a proceder del siguiente modo: Creamos una variable llamada *mediaEjeY* en la que iremos almacenando todos los valores de la componente 'y' del centroide y en cada iteración utilizaremos la función *mean* para encontrar la media del vector *mediaEjeY.*

```
mediaEjeY(a-1) =centy;
centy=mean(mediay);
```

De ese modo conseguimos la componente 'y' del centroide estabilizada. Podemos ver el resultado en la siguiente gráfica.

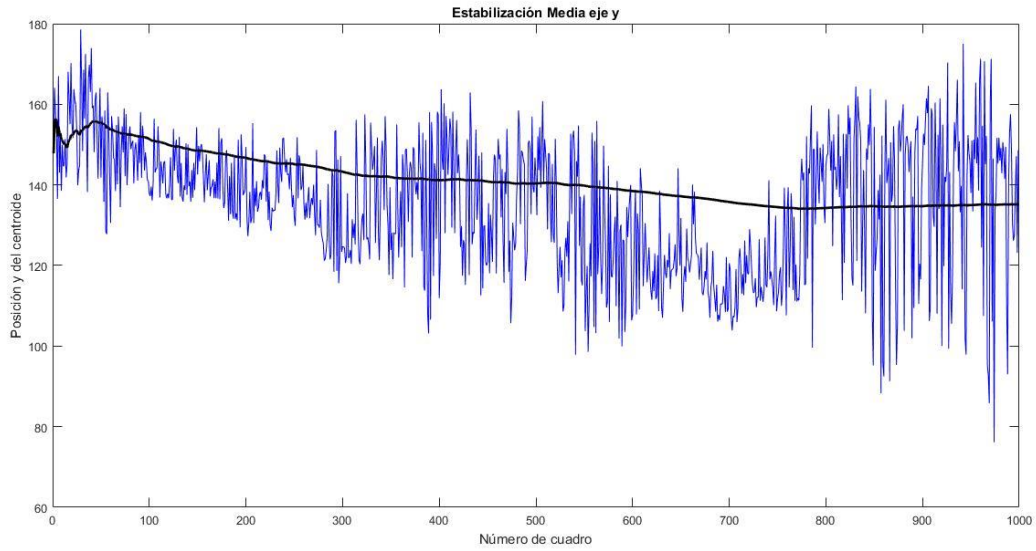

Ilustración 13: Estabilización de la componente 'y' del centroide

Como puede observarse en los primeros cuadros la componente es bastante inestable, pero a medida que se avanza en el número de iteraciones el valor se estabiliza y se hace mas fuerte ante errores o valores dispares.

Un problema grave que se encuentra en esa solución es la acumulación de datos. Si grabáramos un partido real de cerca de dos horas de grabación, es insotenible almacenar un vector con cientos de miles[16] de valores, para solucionar esto simplemente vamos a utilizar la siguiente ecuación:

$$ejeY = \frac{ejeY_A \times (iteracion - 1) + centy}{iteracion}$$

Siendo $ejeY_A$ el valor de la posición del centroide de la iteración anterior (media antigua) y $centy$ será el valor de la posición del centroide de la iteración actual.

Que en el lenguaje de programación Matlab se expresaría del siguiente modo:

```
if ejeY==0
    ejeY=centy; %Si estamos ante la primera iteración
else
ejeY=(ejeY*(a-2)+centy)/(a-1);
end
```

Siendo `ejeY` la posición 'y' del centroide tras la estabilización y `centy` la posición 'y' del centroide en cada iteración. La iteración tiene el valor $a - 1$ ya que, como se ha explicado anteriormente el bucle for comienza por $a = 2$. Este método no nos valdría para analizar los resultados, pero si para evitar el problema mencionado anteriormente.

[16] Para dos horas de grabación el número de cuadros sería $3.600_{segundos/h} \times 2 \times 30_{cuadros/segundo} = 216.000$ cuadros

3.2.9 Obtención de la componente 'x' del centroide completo

Se han desarrollado tres formas distintas para llegar a la estabilización de la componente 'x' del centroide:

- Sin estabilización
- Estabilización de medias
- Estabilización mediante filtro Alpha-Beta

3.2.9.1 Estabilización de la componente 'x' del centroide completo, sin estabilización

Este método de obtención de la componente 'x' consistirá en tomar como centroide el valor que devuelve la función `regionprops`, de este modo tendremos el centroide instantáneo de cada cuadro.

Como aspectos positivos de este primer método, destacamos que el valor de la componente 'x' en cada cuadro tendrá la máxima precisión posible. En cambio, supondrá un problema importante que este valor varie notablemente de un cuadro a otro lo cual aportatá una solución de poca utilidad para el problema del seguimiento, que debe ser una curva lo mas suave posible.

El valor de la componente 'x' del centroide producido por este método se puede ver en la gráfica de la ilustración 12 (izquierda) en color azul.

3.2.9.2 Estabilización de la componente 'x' del centroide completo, estabilización de medias

Para esta segunda forma, si aplicaremos un sistema de estabilización.

No podemos utilizar el mismo método que el caso 'y', ya que el valor x del centroide no tiende a un valor constante a lo largo del proceso, si lo utilizaramos, nos encontraríamos que tras varias iteraciones el centroide se estabilizaría en la parte central del cuadro (como se puede apreciar en la curva roja de la siguiente gráfica) lo cual no proporciona ninguna solución al problema del seguimiento.

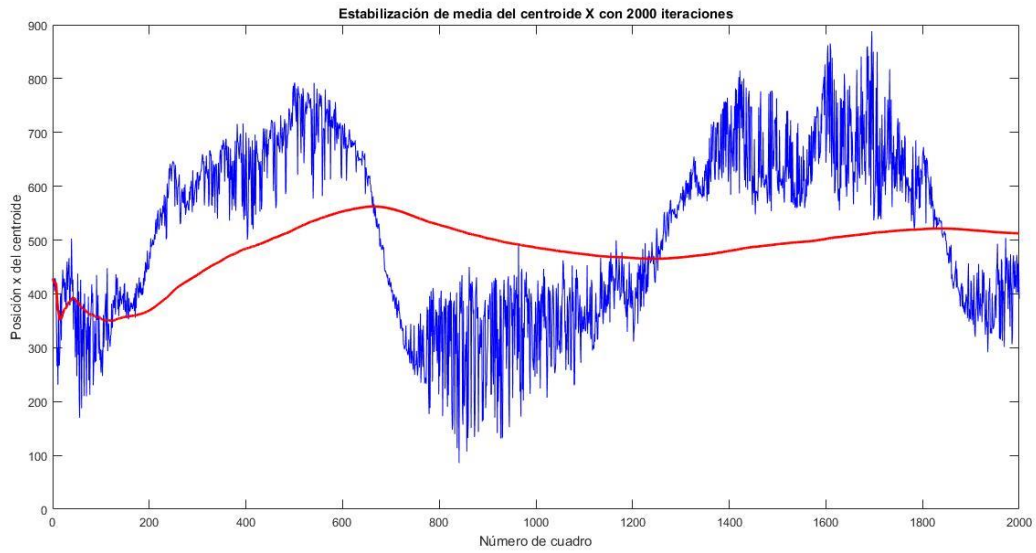

Ilustración 14: estabilizador de media eje "x" (2000 iteraciones)

Para conseguir un resultado fidedigno necesitamos que la curva roja (el valor medio de la componente 'x' del

centroide) esté en todo momento dentro del espacio generado por los picos de los valores azules (valores de la componente x del centroide sin estabilización) y en un punto medio.

Teniendo en cuenta que el algoritmo no puede adelantarse a sucesos futuros, vamos a tener en cuenta solamente valores, de la componente 'x' del centroide, de cuadros anteriores[17]. Para ello vamos a utilizar la siguiente ecuación:

$$ejeX = \frac{\sum_{i=0}^{n} valoresx(iteracion - i)}{n}$$

Siendo $valoresx$ un vector que contiene n valores de la componente 'x' del centoide y n el número de valores previos a la iteración que se utilizan para estabilizar el centroide. En función del valor de n la posición 'x' del centroide tendrá unas características específicas.

Existen funciones en los toolbox de Matlab que hacen ajustes en las gráficas [15], el inconveniente es que utilizan una serie de datos conocidos a priori para producir las curvas, y eso es algo que no tenemos. El algoritmo tiene que funcionar a tiempo real e ir adaptándose activamente.

Para la implementación de este método y procurando no almacenar vectores muy grandes[18] al igual que en el caso de la estabilización de la componente 'y' del centroide, se procedería del siguiente modo en Matlab.

```
c=c+1; %Se va actualizando el puntero
valoresx(c)=centx; % Llena el vector de centroides
if c>=n
    c=0;
end

if length(valoresx)==n % Si el vector está lleno se realiza la operación
    ejeX=sum(valoresx)/n;
end
```

Donde `c` es un contador para ir accediendo al vector `valoresx`. De este modo nunca tenemos un vector de un tamaño mayor que n y ahorramos memoria durante la ejecución del algoritmo.

A continuación, vemos los casos en los que n toma los valores 2, 4, 8, 16, 32 y 64 (1000 iteraciones).

[17] Un suceso que ya ha ocurrido.
[18] Utilizaremos un vector llamado $valoresx$ para almacenar los valores necesarios para hacer el cálculo de un tamaño siempre igual o menor que n

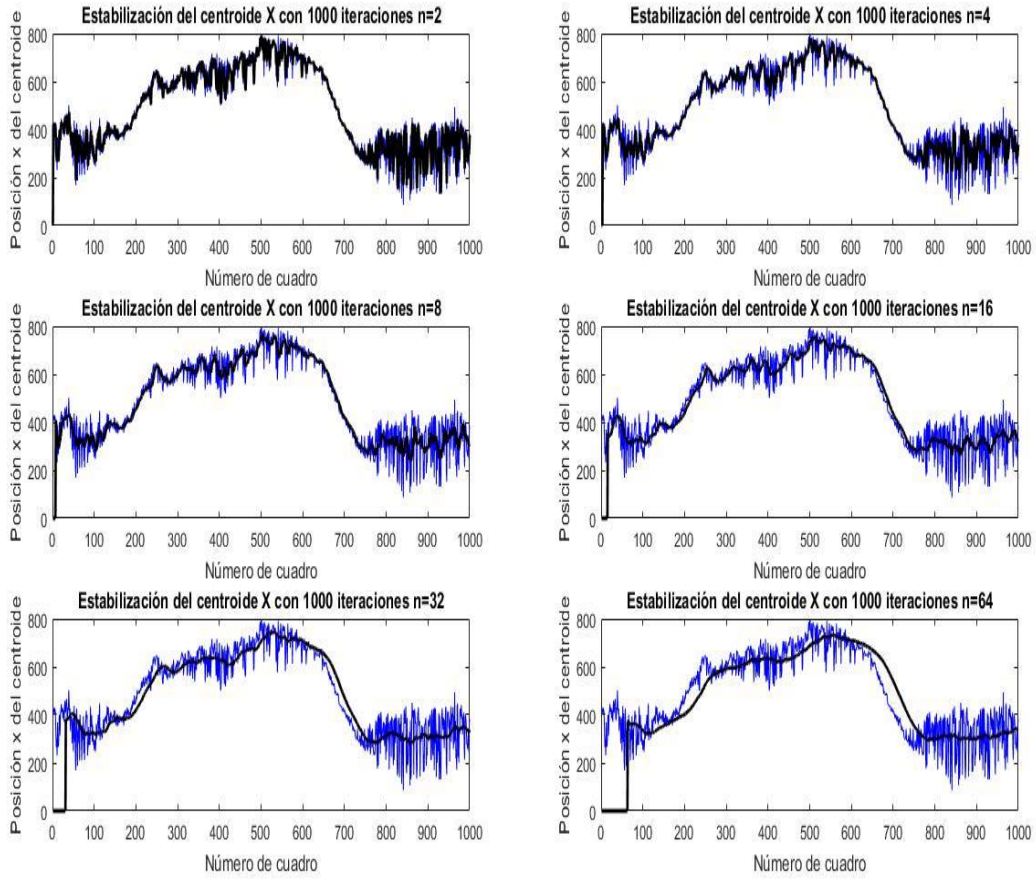

Ilustración 15: valor de la componente 'x' del centroide en función de n (caso completo)

En esta ilustación aparecen 6 gráficas, en éstas la línea roja representa la media, la azul el valor primitivo (original) de la componente 'x' del centroide y la negra la nueva componente 'x' estabilizada.

De estas representaciones podemos sacar varias conclusiones:

- Puesto que n representa el número de iteraciones que utiliza el algoritmo para hacer la estabilización, no se conseguirá el primer valor estabilizado hasta que el número de iteraciones haya superado a n, esto es la línea vertical que aparece al principo de cada gráfica.
- A medida que el valor de n aumenta, la curva que genera la concatenación de las componentes 'x' del centroide es más suave, pero al mismo tiempo es menos fiel al valor orginal ya que las iteraciones previas tienen más peso. Como se ve en el caso en el que $n = 64$ la línea negra experimenta un retraso respecto a la azul, que se hace más evidente en los paneos rápidos. Este es un problema importante puesto que, de forma general, donde transcurre lo más relevante es en los movimientos más rápidos. Así pues, al utilizar este método hay que comprometer la estabilidad[19] con un funcionamiento óptimo. Seleccionaremos un valor de n el cual proporcione una curva suave, pero a su vez proporcione una curva con poco retraso. **Propondremos $n = 24$** que es un valor intermedio entre la gráfica $n = 16$ y $n = 32$.

Tambien podríamos haber utilizado para conseguir este mismo objetivo la siguiente ecuación que se basa en el uso del filtro alpha-beta [16]:

[19] Que en este caso se refiere a la suavidad de la curva.

$$ejex = ejex \times \alpha + centx \times \beta$$

En la que inicialmente el valor $ejex$ será igual a $centx$ para evitar que dicho valor comience desde cero.

Cuanto mayor es el parámetro α, mayor es el efecto de la entrada $ejex$ y menor amortiguación puede apreciarse. Un valor bajo de β es efectivo para un control de los cambios repentinos de velocidad. Además, a medida que el α aumenta más allá de la unidad, la salida se vuelve más rugosa y más desigual que la entrada.

Por lo que deberíamos elegir un valor alto de α (menor que la unidad) y un valor bastante bajo de β. Éstos serán[20] $\boldsymbol{\alpha = .95}$ **&** $\boldsymbol{\beta = .05}$ **.**

La implementación de este método en Matlab se realizaría del siguiente modo.

```
if ejeX==0
    ejeX=centx;
end

ejeX=ejeX*alpha+centx*beta; %ejeX es el centroide con alpha-beta
```

Podemos ver a continuación el funcionamiento del filtro alpha-beta en los casos en que $\alpha = .90$ & $\beta = .10$ y $\alpha = .95$ & $\beta = .05$.

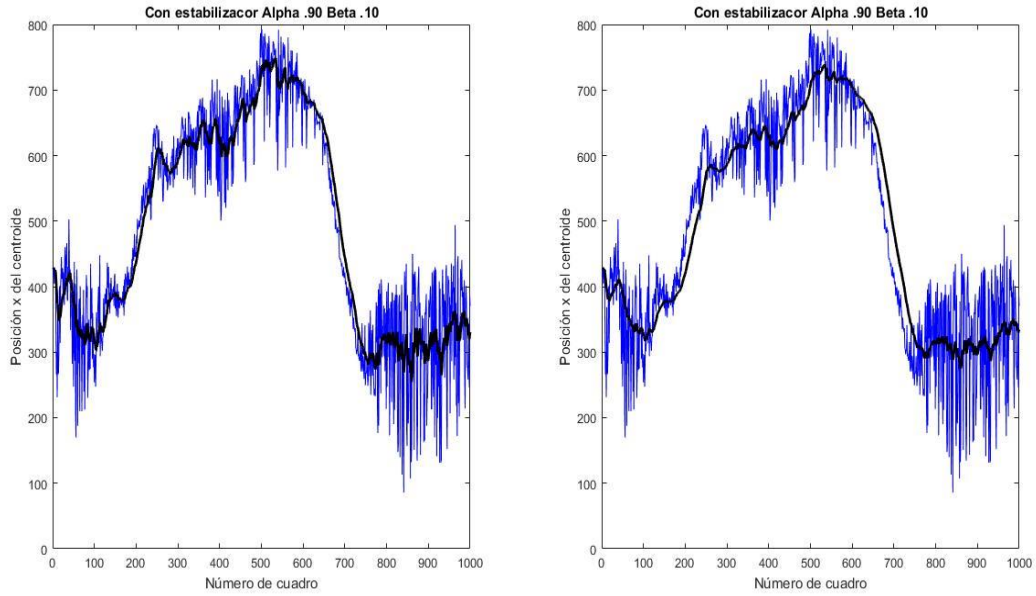

Ilustración 16: Valor de la componente 'x' del centroide en función de α y β (caso completo)

[20] El resultado con estos valores es lo suficientemente suave sin generar demasiado retardo.

3.2.10 Representación de las componentes del centroide parcial

Puesto que este método es una versión adaptada[21] del completo y con el objetivo de evitar caer en la reiteración, reflejaremos principalmente las diferencias con el método completo

Al igual que en el caso anterior, comenzaremos viendo la representación de la concatenación de los centroides de los 1000 primeros cuadros.

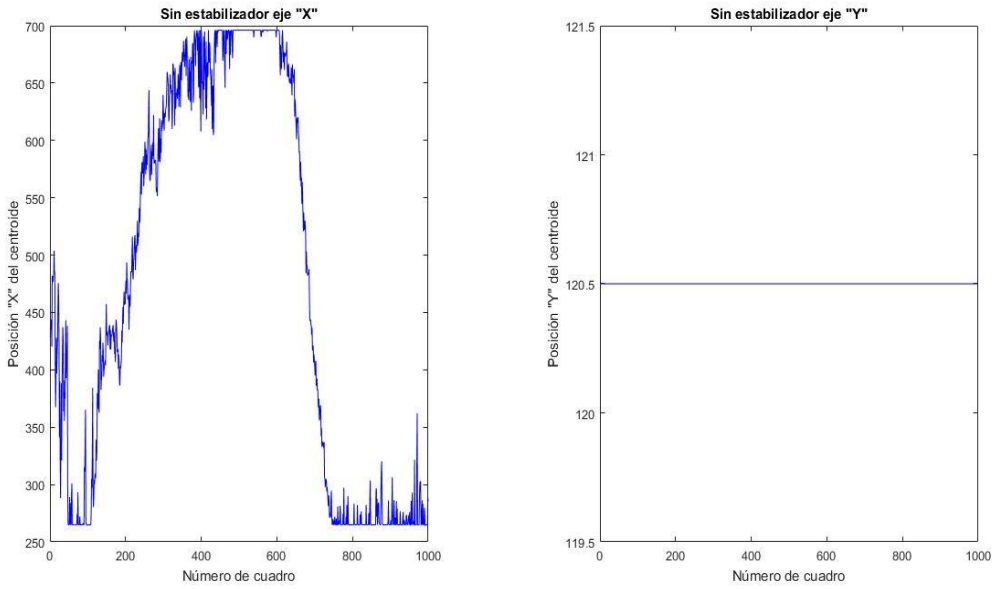

Ilustración 17: Centroide a lo largo de 1000 iteraciones caso parcial

Como puede observarse en la ilustración (izquierda) este método nos proporciona un centroide mucho mas suave a lo largo del tiempo. Lo cual es comprensible ya que movimientos lejos de la zona de interés (fuera del encuadre) no son tenidos en cuenta.

De igual modo, en la ilustración (derecha) vemos como, en comparación con la ilustración 12 (que muestra las posiciones del centroide en el caso completo) , el valor de la componente 'y' del centroide está más acotado.

En el caso parcial nos encontramos con una variable de la que dependerá enormemente el funcionamiento del algoritmo: **el tamaño del cuadro**. En este método, al utilizar el contenido del cuadro para realizar la obtención del centroide, la bondad del programa estará directamente relacionado con el tamaño del cuadro.

Es por ello por lo que para evitar el aparatoso proceso de establecer las condiciones del cuadro y puesto que partimos de una grabación del campo completo vamos a definir del tamaño del cuadro automáticamente:

- La altura del cuadro será la altura de la grabación original, de este modo no habrá movimiento en el eje 'y' y no será necesaria la obtención ni estabilización de la componente 'y' del centroide.
- El ancho del cuadro dependrá de la distancia de grabación 'd' y del ángulo de visión de la cámara 'θ'.

El video que estamos usando abarca todo el ancho del campo (aproximadamente 30 metros) y fue grabado con una cámara cuyo ángulo de visión es 85º por lo que mediante un cálculo trigonométrico determinamos que la distancia a la que se ha grabado es de 16,4 metros[22]. A medida que esa distancia se acorte el encuadre alojará un porcentaje menor del campo según la siguiente ecuación.

[21] Adapatada para el supuesto de disponer de una sola cámara que **no** esté abarcando todo el campo.
[22] $(30 \div 2) \div sen(85 \div 2) = 22{,}2$ y ahora $22{,}2 \times \cos(85 \div 2) = 16{,}4$ metros

$$Ancho = \tan(\frac{\theta}{2}) \times d$$

Siendo θ el ángulo de visión de la cámara y d la distancia a la que coloquemos la cámara. Este valor de la anchura del cuadro que proporciona la ecuación está en metros así que el número de columnas que tenga la imagen captada por la cámara será igual a ese valor. Aunque como partimos de una grabación de campo completo aplicaremos la siguiente transformación en lenguaje Matlab.

```
%Al tener una grabación que abarca todo el campo hacemos lo siguiente

ancho=tand(theta/2)*distancia;  %Ancho del cuadrado de encuadre
ancho=floor((ancho*Columnas)/15);     % 15 es mitad de 30 metros que es lo
                                  que mide el campo completo
```

Siendo `Columnas` el número de columnas de la grabación original, `theta` el angulo de visión de la cámara y `distancia` la distancia a la que colocamos al cámara.

Asi pues para ilustrar nuestro ejemplo vamos a suponer que la cámara se colocó a **siete** metros del campo

3.2.11 Estabilización de la componente 'y' del centroide parcial

Como se ha mencionado anteriormente en este caso no vamos a estabilizar la componente 'y' del centroide, solo nos centraremos en el desplazamiento de éste en el eje 'x'. Por lo que no hay necesitad de estabilización ya que el valor final será igual que el hallado en la figura 17 (izquierda).

3.2.12 Obtención de la componente 'x' del centroide parcial

Como la obtención del centroide nos proporciona un valor más estable y suave que con el método completo, la estabilización de la componente 'x' del centroide será menos invasiva.

Vemos el resultado de aplicar los dos mismos métodos que en el caso completo. En primer lugar, vemos los casos en los que n toma los valores 2, 4, 8, 16, 32 y 64 (1000 iteraciones).

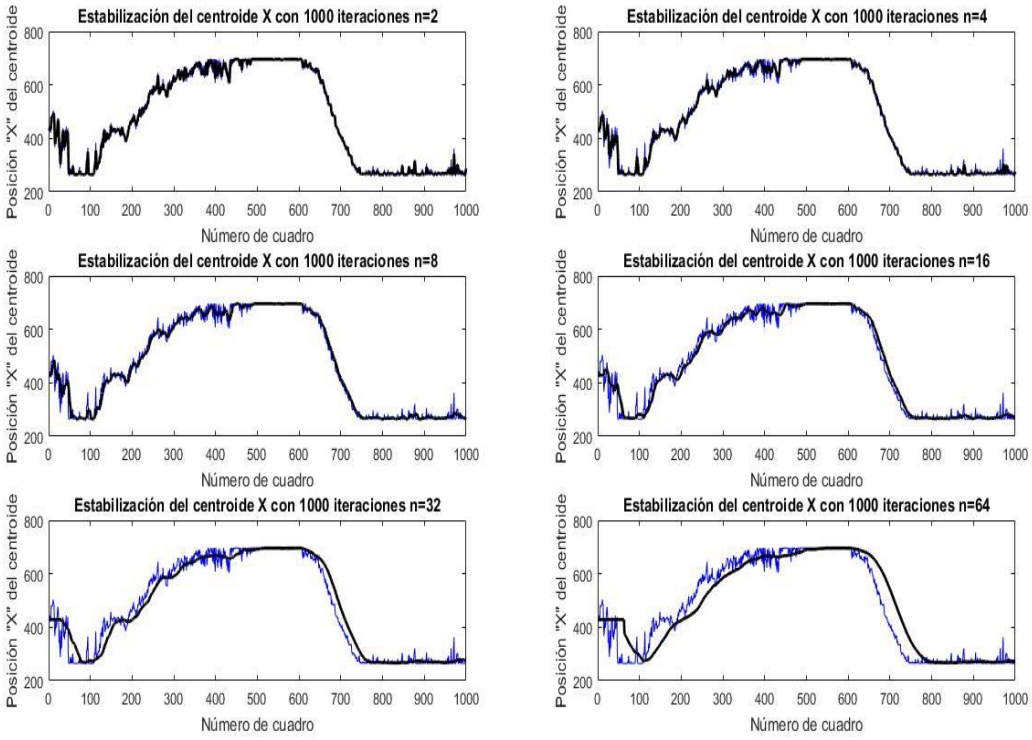

Ilustración 18: valor de la componente 'x' del centroide en función de n (caso parcial)

En esta ilustación aparecen 6 gráficas, en éstas la línea azulrepresenta el valor primitivo (original) de la componente 'x' del centroide y la negra la nueva componente 'x' estabilizada. En este caso por convección lo valores previos al $número\ de\ cuadro = n$ tendrán el valor del primer centroide de la imagen en lugar de cero.

Ya que este método parcial proporciona por si mismo una solución más estable, el valor de n idóneo está al rededor de $n = 16$. Valores más pequeños producen picos no deseados y valores más grandes, producen retrasos demasiado grandes. Seleccionaremos por por lo tanto el valor **n=16** para el algoritmo.

Cabe destacar que estos resultados están relacionados con la anchura del cuadro y por tanto con la distancia. Si la distancia hubiera sido mayor, el cuadro sería mas parecido al caso completo y n debería ser más grande.

Al igual que en el caso completo puede abordarse el problema mediante el uso del filtro Alpha-Beta cuyos resultados con valores $\alpha = .90$ & $\beta = .10$ y $\alpha = .95$ & $\beta = .05$ se muestran en las siguientes gráficas.

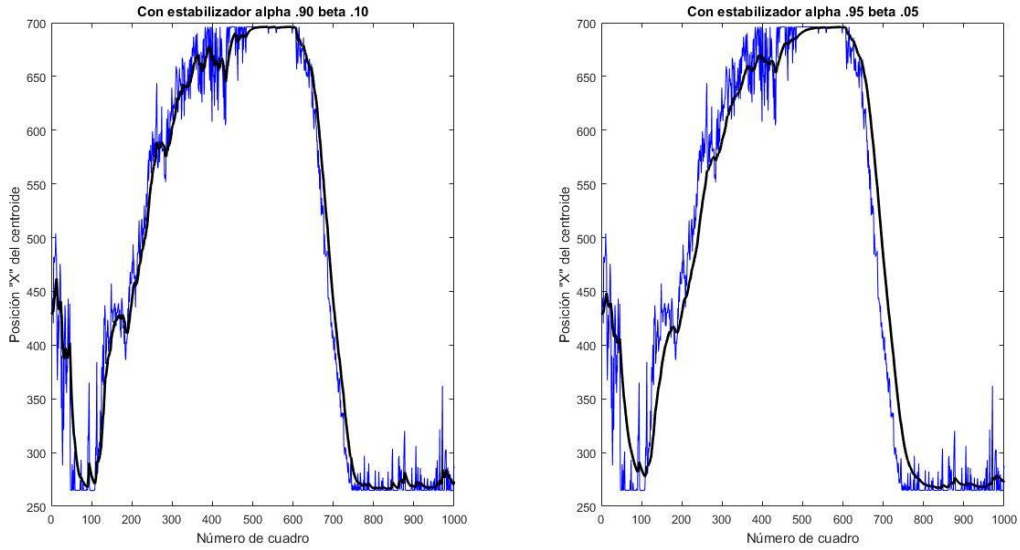

Ilustración 19: Valor de la componente 'x' del centroide en función de α y β (caso parcial)

Elegimos, de igual modo que en el caso completo, el valor $\boldsymbol{\alpha = .95}$ & $\boldsymbol{\beta = .05}$.

3.2.13 Muestra de resultados

En la muestra de resultados hasta el momento se ha realizado en Matlab mediante la función `plot` aunque hay multitud de formas de mostrar los resultados, de hecho la intención original era mostrar simplente un valor que sería el ángulo de rotación del paneo.

Como éste es un documento en el que intentamos mostrar las bondades de ciertos sistemas de seguimiento la mejor opción para mostrar los resultados será aquella que nos haga ver los resultados de la forma más visual posible. Por lo tanto, la muestra de resultados se realizará del siguiente modo:

```
ancho=250;
alto=100;
x=[centx-ancho centx+ancho centx+ancho centx-ancho centx-ancho];
y=[media+alto media+alto media-alto media-alto media+alto];

subplot(2,1,1)
imshow(I01)
hold on
plot(centx,media,'r*')
plot(x,y,'r','linewidth',2) % dibuja el rectangulo
hold off
subplot(2,1,2)
imshow(I3)
hold on
plot(centx,media,'r*')
plot(x,y,'r','linewidth',2) % dibuja el rectangulo
hold off
```

Para empezar, hemos definido el tamaño de un cuadrado que nos servirá para ejemplificar un posible enfoque a la escena, será un rectángulo a una distancia del centroide igual a los valores "ancho" y "alto" establecidos.

Para la muestra he optado por presentar dos imágenes en una misma figura, una estará en formato uint8 a color y la otra será la imagen binaria para apreciar in situ como funciona el algoritmo sobre las dos imágenes.

3.3 Eleción de la mejor opción de preprocesado

Durante el desarrollo del algoritmo se han ido justificando cada una de las decisiones que se han tomado por lo que la pregunta a "¿por qué es este el mejor método de seguimiento?" ha sido contestada, ahora bien. Estamos ante dos soluciones: la completa y la parcial.

Siemplemente dependerá del caso que necesitemos abordar. La solución parcial está destinada a un supuesto en el que no tengamos la totalidad del campo en el encuadre por lo que será útil cuando la distancia de grabación sea menor que 16 metros[23]. Por otro lado, el caso completo lo utilizaremos cuando queramos que a partir de una grabación de campo completo queramos extraer un cuadro de un tamaño concreto.

3.4 Estudio del tiempo

El tiempo es uno de los puntos más importantes en este proyecto pues necesitamos que sea el menor posible para que el algoritmo pueda ser implementado en tiempo real. A medida que disponemos de lenguajes de programación de mayor nivel baja la velocidad de dicho lenguaje. Como ya se ha mencionado anteriormente, Matlab es un lenguaje de alto nivel que empleará un tiempo muy grande en ejecutar este algritmo de seguimiento en comparación al tiempo que emplearía el lenguaje de programación C.

Aun así, se ha estudiado como afecta cada parte del código desarrollado, de este modo se ha logrado depurar mucho éste algritmo para hacerlo más eficiente y veloz al mismo tiempo.

A continuación, se muestran los tiempos que invierte cada sección del programa en primer lugar para el caso completo y a continuación el caso parcial:

Tabla 3–1. Tiempos de ejecución del algoritmo por partes (modo Completo).

Sección del Algoritmo	Tiempo (s)	Dentro del bucle
Iniciación	0,3469	No
Lectura del video	0,4633	No
Lectura de las imágenes	0.1111	Si
Transformación de la imagen a imagen diferencias	0,0072	Si
Transformación de la imagen a imagen binaria	0,0127	Si
Operaciones morfológicas	0,0116	Si
Obtención del centroide	0,0064	Si
Muestra de resultados	0,0934	Si

[23] Esta distancia se establece para un campo de 30 metros. La explicación se da en el apartado 3.2.10

Tabla 3–2. Tiempos de ejecución del algoritmo por partes (modo Parcial).

Sección del Algoritmo	Tiempo (s)	Dentro del bucle
Iniciación	0,3469	No
Lectura del video	0,4633	No
Lectura de las imágenes	0.1111	Si
Transformación de la imagen a imagen diferencias	0,0072	Si
Transformación de la imagen a imagen binaria	0,0127	Si
Operaciones morfológicas	0,0116	Si
Obtención del centroide	0,0021	Si
Muestra de resultados	0,0760	Si

Estos valores se han obtenido mediante la función de Matlab `tic` y `toc` en sus respectivas posiciones en el código.

El tiempo de iniciación no se realizará mas que una vez ya que el resto de iteraciones corrien dentro del bucle `for`. Del mismo modo, la primera lectura del video solo ocurrirá una vez al comienzo de la ejecución.

Por lo tanto, el tiempo de ejecución de un cuadro incluyendo su representación supondrá un tiempo de: 0,24 segundos. Esto podrá variar entre distintas iteraciones puesto que evidentemente los cuadros no son todos iguales y la comparación entre algunos dará lugar a mayor movimiento de objetos, aumentando asi la población de unos en la imagen binaria y complicando algo mas el cáculo.

Como podemos apreciar donde mayor tiempo se invierte es en la lectura de las imágenes y en la muestra de resultados. En el primer caso se trata de la lectura del video, un paso que es exclusivo de la adaptación del código al entorno Matlab. Algo parecido ocurre con el segundo caso en el que para apreciar los resultados Matlab genera unas representaciones gráficas de los datos Si nos basamos únicamente en los demás apartados vemos que el tiempo total en el **caso completo es de 0.0379** y en el **caso parcial es de 0.0333** lo cual es un dato muy importante puesto que el tiempo que debe invertir no debería ser mayor que 0.03[24] para que funcionara a tiempo real.

Los valores arrojados son aproximados que han sido obtenidos una vez el algoritmo había realizado varias iteraciones. Entre una iteración y otra los valores del tiempo van cambiando ya que por ejemplo el paso la obtención del centroide o de la trnaformación morfológica será más complejo.

3.5 Aplicación del algoritmo en casos reales

Veremos el algoritmo utilizando el caso completo aplicado a los distintos videos que se mencionarion en el punto 3.1 sus bondades e inconvenientes:

El primer video con las peculiaridades ya mencionadas vemos que el centroide funciona correctamente, el error es muy pequeño y el algoritmo hace su trabajo sin problemas. Al ser un en enfoque más cercano el cuadro rojo no nos dice demasiado, simplemente habría que aumentar los limites del cuadro. Este ejemplo nos sirve para apreciar muy de cerca las operaciones morfológicas aplicadas a la imagen ya que el rango de movimiento de los jugadores es muy limitado debido al posicionamiento de la cámara.

[24] Este tiempo es el resultado de dividir un segundo entre los 30 ya que como se comentó en el apartado 3.2.2. la tasa de lectura será de $30_{cuadros/segundo}$

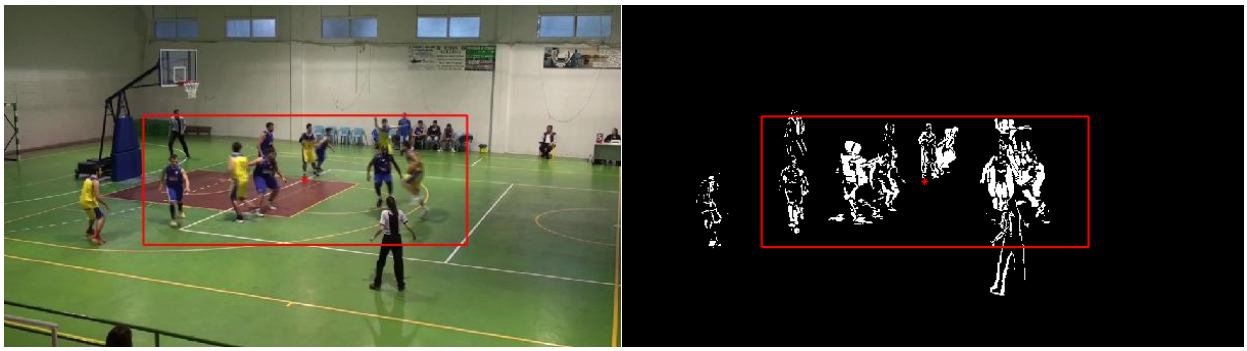

Ilustración 20: Seguimiento en video 1

El segundo video si tiene una colocación[25] adecuada, pero en cuanto a la posición como ya se ha mencionado es pésima, el problema es que llegamos a ver la parte posterior de las cabezas del publico, y sus movimientos (aunque minimos) son ligeramente detectados por el algoritmo.

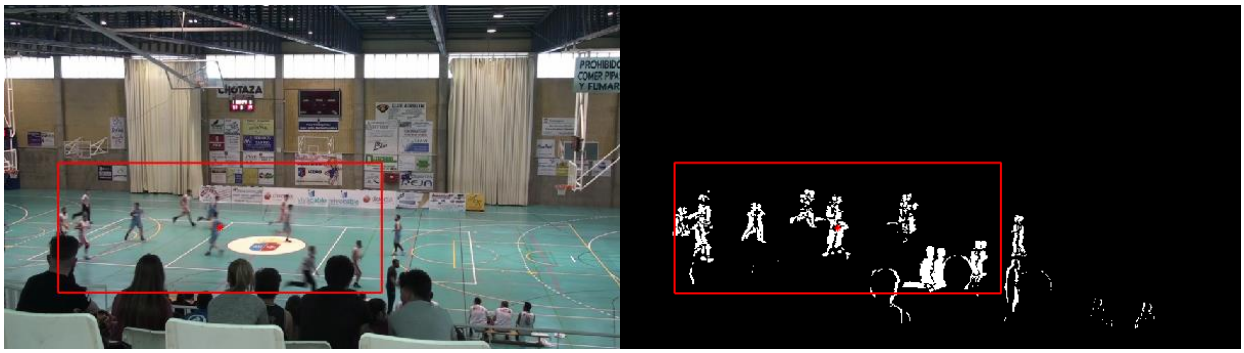

Ilustración 21: seguimiento en video 2

En el tercer video, al ser una grabación desde bastante lejos, podemos apreciar perfectamente el campo completo, el único problema que teníamos en esta grabación es que fue realizada a pulso por lo que la minima variación detecta movimiento. Gracias a las operaciones morfológicas se ha conseguido paliar este efecto y tenemos un sistema de seguimiento aceptable.

[25] En cuanto a la distancia al campo se refiere.

Ilustración 22: Seguimiento en video 3

Por ultimo, en el cuarto video podemos apreciar que el algoritmo no funciona correctamente; debido a las condiciones de grabación la imagen diferencias no tiene una separación clara entre las dos poblaciones por lo que el método de otsu tiene un acierto bastante bajo.

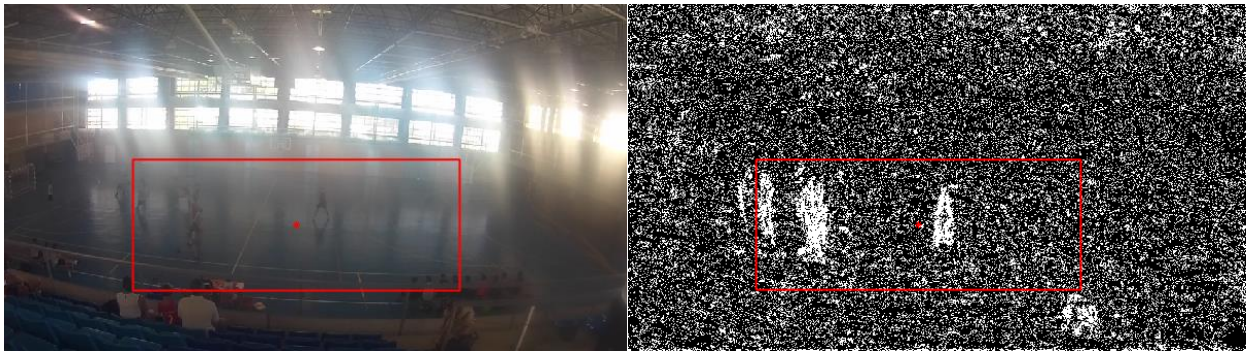

Ilustración 23: Seguimiento en video 4

Con operaciones morfológicas más agresivas conseguiríamos un resultado bastante mejor pero no sería valido para el resto de los videos. Considerando que este cuarto video y sus malas condiciones son una excepción se mantuvo el conjunto de operaciones que proporcionan buenos resultados a los tres ejemplos anteriores.

4 CONCLUSIONES

Las actitudes son más importantes que las aptitudes.

- Wiston Churchill -

Una vez presentado el algoritmo de seguimiento mediante el cálculo del centroide se puede apreciar que es un método bastante válido para la concepción de un sistema de seguimiento autónomo. Se presentan a continuación, a modo de resumen, los conceptos que han resultado más conflictivos, y a más error han conducido, del algoritmo propuesto además de las posibles soluciones.

Previamente introduciremos los conceptos básicos del Baloncesto y porqué este sistema no es válido para otros deportes sin realizar algunos ajustes. Finalmente veremos unas conclusiones tanto de la ejecución del proyecto como de la viabilidad para la implementación del caso real.

4.1 El Baloncesto

Según el Reglamento FIBA[26]: *"El baloncesto lo juegan dos equipos de cinco jugadores cada uno. El objetivo de cada partido es encestar en la canasta del adversario e impedir que el equipo contrario enceste".* Con esta premisa podemos sacar varias conclusiones en cuanto al movimiento que estudiamos.

- La primera es que en este deporte importa mucho la defensa y como el número de jugadores por equipo es de cinco todos ellos tienen que defender, del mismo modo es igualmente importante el ataque. Es una diferencia muy grande con respecto a otros deportes como el Fútbol, Fútbol sala, Rugby … etc en los que miembros del equipo se quedan defendiendo en zonas muy separadas del campo o incluso está la figura del portero. El baloncesto es un deporte con muy poco espacio por jugador en el campo, por lo tanto, podemos concluir **que los jugadores por norma general están muy cerca unos de otros**.
- Otra conclusión será que **lo más importante de este deporte es la canasta**, como del Fútbol es el gol, por lo tanto, cerca de ella será la zona en la que más jugadores habrá, más movimiento ocurrirá, y mayor importancia tendrá para el posible espectador.

Estas conclusiones se tuvieron en cuenta a la hora de diseñar el algoritmo, es esa la razón por la que dicho

[26] Del francés *Fédération Internationale de Basket-ball* (Federación Internacional de baloncesto).

algoritmo no es válido para otros deportes donde no se cumplan las premisas ineherentes al baloncesto. Un ejemplo puede ser el Fútbol, donde movimiento rápido de los jugadores no implica necesariamente que sea la zona de interés.

Gracias a estas peculiaridades es evidente que el cálculo del centroide en una imagen es un dato altamente válido. En el caso de que haya algún tipo de error puntual el algoritmo podría volver a adaptarse al partido, por ejemplo, que se produzca una trifulca y la zona de interés se traslade al lugar en el que miembros del banquillo acompañantes estén causando desorden o camorra. Una vez que el entuerto se solucione (el cual por cierto será también filmado de forma automática) el sistema volverá a trabajar de forma normal al juego.

Para crear un algoritmo destinado a la filmación autónoma de otro deporte habría que tener en cuenta otras variables como por ejemplo la posición del balón en un partido de Fútbol o la posición relativa de los jugadores durante todo el encuento. Por supuesto tener localizados a todos los jugadores en el campo para el deporte baloncesto proporcionaría un sistema de filmación automática perfecto, tal y como nos mostraba el *paper* presentado en el segundo apartado.

Con la exposición de los diferentes los ejemplos con este sistema se pretende demostrar que no hace falta tanta precisión en cuanto a las posiciones exactas de los jugadores para conseguir un resultado aceptable. O dicho de otro modo, no es totalmente necesario la localización precisa de cada jugador para una grabación de plano general de un partido.

4.2 Apartados de mayor interés.

A la hora de esbozar los principales puntos de conflicto de este algoritmo que han sucedido, destacan claramente 3 de ellos:

- El umbral adecuado a utilizar sobre la imagen diferencias.
- Las operaciones morfológicas.
- La estabilidad del centroide.

4.2.1 Umbral

Tal y como se ha expuesto, se solventó mediante el método de Otsu[27] auqnue no fue automático, esta decisión fue precedida de muchos ensayos con otros métodos tales como establecer un umbral manualmente o incluso no establecerlo y trabajar con la imagen diferencias.

Se puede apreciar que no es un método perfecto. En grabaciones de mala calidad en las que hay mucho movimiento relativo o en las que el movimiento es mínimo, el umbral confunde el movimiento con el entorno lo cual da lugar a imágenes binarias en las que se resaltan zonas en las que realmente no ha habido movimiento.

El principal problema del método de Otsu, para este fin, es el caso en el que no hay absolutamente ningún movimiento. `Graythresh` tiene que dar un umbral siempre y si no hay movimiento dará valores erróneos del umbral, conduciendo a errores graves en el resto del código. Podría solucionarse creando excepciones para esos casos como la excepción creada para el caso del centroide cero.

Cuando se utilizó un umbral fijo conseguíamos un funcionamiento correcto si se utilizaban operaciones morfológicas más severas y complejas pero resultó infructuoso ya que si se cambiaban los videos florecían dos nuevos inconvenientes: o las operaciones morfológicas se eternizaban o simplemente no se conseguía un centroide.

Este sistema también tiene una peculiaridad con respeto a la imagen diferencias y es que tanto los valores

[27] Apartado 3.2.4.1

negativos como positivos[28] se ha convertido a unos en la imagen binaria, es decir, no solo se le da importancia a donde va el objeto si no de donde proviene. La decisión de incluir esta capacidad surge, tras muchos ensayos, con el fin de aumentar la zona que denota movimiento. Con el aumento de esta zona (aumento de la población de unos) se pueden realizar operaciones morfológicas que dan como resultado una imagen binaria más representativa del movimiento.

4.2.2 Operaciones morfológicas

Este es sin duda el paso más complejo de todo el código. Hay infinitas combinaciones de operaciones y todas aportan opciones interesantes, pero aumentar el número de operaciones daba lugar a procesados muy largos, al menos en Matlab, y era preferible una imagen binaria un poco menos acertada y mayor velocidad de procesado.

Y se refiere el párrafo anterior cuando dice un poco menos acertada a que no es capaz de diferenciar a todos y cada uno de los jugadores en el campo. De hecho, eso hubiera sido otro método muy interesante para realizar el seguimiento, que se intentó antes de decantarse finalmente por el definitivo.

Dicho método consistía en utilizar las operaciones morfológicas para delimitar individualmente a los jugadores, una vez separados, el paneo incluiría a la mayor cantidad posible de jugadores individuales. Este método arrojaba varios problemas como:

- En los muchos casos en los que los **jugadores** están **pegados** y se mueven a la vez nos entonctramos con la dificultad de diferencias unos jugadores de otros.
- En algunos casos y debido a la iluminación en un partido de baloncesto, que condiciones generales se juega en interior, las **sombras** de los jugadores eran interpretadas como un jugador más cuando se realizaban operaciones morfológicas. En cambio para el método elegido, las sombras no suponen un problema grave ya que hay tantas sombras como jugadores y se compensa la población de unos en la imagen binaria.
- Por último, habría problemas cuando los **jugadores** están **quietos**. Si bien es cierto que no hay casos en los que no haya ningún movimiento, el método de Otsu le da mayor importancia al movimiento de los jugadores que se mueven que a los que no, por eso en la imagen binaria ocurriría que los jugadores estáticos no tendrían representación.

A pesar de estos problemas sería interesante seguir indagando en un futuro en este sistema, proponiendo soluciones como el uso de imágenes a color para conseguir imágenes diferencia o quizás un sitema para memorizar donde se encontraba un jugador y asumir el hecho de que no puede desaparecer entre un cuadro y otro.

A modo de resumen, para realizar un algoritmo de seguimiento basado en la obtención del centroide se presenta esta serie de operaciones morfológicas, en cambio para realizar un algoritmo de seguimiento basado en el número de jugadores habría que realizar unas operaciones morfológicas diferentes destinadas en delimitar el máximo posible a los jugadores.

4.2.3 Estabilidad del del centroide

Este apartado se basa en conseguir una mayor precisión de la posición del centroide y en suavizar el paneo.

Puesto que buscamos que el sistema autónomo sustituya a un operario de cámara, sería un error que el movimiento de paneo transmita sensaciones robóticas, es decir, que el centroide, el cuadro y por tanto el paneo fuera dando saltos entre imágenes. Es por ello necesario estabilizar el movimiento, evitar los cambios abruptos y alcanzar similitudes con el trabajo de un operario de cámara.

[28] Por encima y por debajo del umbral

Este es el apartado que supuso un punto de inflexión a la hora de analizar los datos. Con todos los centroides conocidos podría hacerse una media por zonas, es decir, podríamos coger un centroide dado y calcular la media de su vecindad, pero eso es un paso que solamente podría hacerse a posteriori, por lo que no era una solución si queríamos que trabajara a tiempo real. Así que habría que calcular la media a partir de datos conocidos hasta ese momento, ahí es donde entra en juego el filtro Alpha-Beta y el filtro de medias que sólo utiliza datos ocurridos hasta el momento.

Ademas el número que datos que utiliza está limitado, de este modo el algoritmo no almacena todos los centroides desde su comienzo si no que va variando un único centroide, en el caso Alpha-Beta, o almacena un máximo de n centroides previos, con el filtro de medias.

4.3 Ejecución y viabilidad

Habiendo visto de forma superficial los principales problemas, no hay razón por la que no se pudiese seguir avanzando en el desarrollo de la idea en aspectos de memoria e interpretación de datos, esto complicaría mucho mas el programa haciéndolo como más lento aunque posiblemente mas preciso.

Personalmente considero que unos mejores resultados se hubieran obtenidos embarcándose directamente en intentar el producto final (un sistema autónomo), se habría conseguido una retroalimentación muy fructífera y un dispositivo listo para funcionar. También es cierto que eso hubiera supuesto un aumento muy grande de la complejidad del proyecto, entre otras cosas al introducir aspectos no orientados exclusivamente al análisis de imágenes si no al control automático, programación en un lenguaje más de menor nivel, uso de sistemas electrónicos digitales para introducir el código en un microcontrolador…

Sin ninguna duda habría sido un proyecto que tocase todas y cada una de las áreas que se estudian en el grado, pero el límite de tiempo ha condicinoando muchísimo los planes de ejecución. Finalmente, la decisión de centrarse en el análisis de la imagen fue a raíz de que es el área de especialidad que he elegido en este grado (imagen y sonido). Por eso pensé que era la parte mas interesante.

Como se vió en la sección de estado del arte, este modo autónomo de realizar un seguimiento está sorprendentemente poco investigado. Personalmente, si tuviera la oportunidad de seguir con el proyecto, continuaría implementando los demás bloques.

5 CÓDIGOS

La vida no está hecha de deseos y sí de los actos de cada uno.

- Paulho Coelho -

Como se expuso en los apartados 3 y 4 de este documento se refleja a continuación el código completo, y no seccionado, tal y como se utilizarian en el programa Matlab. Dicho código, aunque no se ha mencionado anteriormente, es totalmente original y las referencias que tiene provienen de los conocimientos que adquirí en las asignaturas del grado que este proyecto finaliza.

Solo habría que establecer:

- El video en concreto del que queramos que realice el seguimiento.
- El tipo de filtro seleccionado (Alpha-Beta o de medias)
- El tamaño del cuadro en el caso Completo

5.1 Algoritmo de seguimiento caso Completo

```
%%  Parte 1. Inicialización Programa

clc
clear all
close all

% Declaración de variables

ancho=500;  %Ancho del cuadrado de encuadre
alto=200;   %Alto del cuadrado de encuadre
%USAMOS ALPHA BETA O FILTRO DE MEDIAS ('0' alphabeta '1' para medias)
filtro=0;

alpha=.95;  %Datos para el filtro Alpha Beta
beta=.05;   %Datos para el filtro Alpha Beta
```

```
cuadroInicial=2;    %Debe comenzar en 2 ya que se comparaá con la imagen
original
cuadroFinal=1005;   %Elegimos hasta donde queremos que funcione el algoritmo
ejeX=0; %Inicialización de la posición del centroide
ejeY=0; %Inicialización de la posición del centroide
n=24;    %Número de componentes que utiliza el filtro de medias
c=0;    %Contador para estabilización del centoide, sirve para el filtro de
medias

%% Parte 2. Lectura del video

%Aquí se introduce el video a analizar. En este caso el video es:'ciba.avi'
video= VideoReader('videos/ciba.avi');


for a=cuadroInicial:cuadroFinal;    %'a' será el numero de iteración
comenzando en 2

% Lectura de las imagenes y Reescalado

I01=imresize(read(video,a),0.5);    %En color
I02=imresize(read(video,a-1),0.5);  %En color siguiente fotograma

%% Parte 3. Transformacion de imagenes a double y resta

I11=rgb2gray(I01);  %Pasamos a blanco y negro
I1=im2double(I11);  %Pasamos a double

I2=rgb2gray(I02);   %Pasamos a blanco y negro
I2=im2double(I2);   %Pasamos a double

I3=I1-I2;

%% Parte 4. Paso a binario

umbral=graythresh(I3);  %Método de otsu
I3(abs(I3)>umbral)=1;   %Pasando a blanco y negro con el umbral de Otsu
I3(I3~=1)=0;    %Si algun valor es distinto a uno será cero

%% Parte 5. Operaciones morfologicas

se=strel('square',2);

I4=imerode(I3,se);
I4=bwmorph(I4,'clean',inf);

I4=imerode(I4,se);
I4=bwmorph(I4,'clean',inf);

I4=bwmorph(I4,'dilate',1);
I4=bwmorph(I4,'clean',inf);

%% Parte 6. Cálculo centroide

[Filas,Columnas]=size(I4);

s=regionprops(I4,'centroid');
```

```
    %Sirve para evitar las imagenes en la que todos los valores son
    %cero (que solo son obtenibles mediante operaciones morfológicas)

if 0==isempty(cat(1,s.Centroid))    %Si está vacío 0==1 por lo que no entra
en el bucle
 cent=cat(1,s.Centroid);
end

centx=mean(cent(:,1));  %Punto centroide X
centy=mean(cent(:,2));  %Punto centroide Y

%Estabilizadores Y
    %Sirve para estabilizar la variable Y del centroide.

if ejeY==0
    ejeY=centy; %Si estamos ante la primera iteración
else
ejeY=(ejeY*(a-2)+centy)/(a-1);
end

%Estabilizadores X
    %Sirve para estabilizar la variable X del centroide.

if filtro==0

    % Método alpha-beta

    if ejeX==0
      ejeX=centx;
    end

    ejeX=ejeX*alpha+centx*beta; %ejeX es el centroide con alpha-beta

else

    %Método suma y medias

    c=c+1;    %Se va actualizando el puntero
    valoresx(c)=centx;    %Llena el vector de centroides
    if c>=n
    c=0;
    end

    if length(valoresx)==n   %Si el vector está lleno se realiza la operación
       ejeX=sum(valoresx)/n;
    end
end

%% Parte 7. Muestra de resultados

%Crea un cuadrado según las condiciones proporcionadas

x=[ejeX-ancho/2 ejeX+ancho/2 ejeX+ancho/2 ejeX-ancho/2 ejeX-ancho/2];
y=[ejeY+alto/2 ejeY+alto/2 ejeY-alto/2 ejeY-alto/2 ejeY+alto/2];

%Muestra de resultados

subplot(2,1,1)
imshow(I01)
```

```
hold on
plot(ejeX,ejeY,'r*')    %Representa el centroide con un asterisco rojo
plot(x,y,'r','linewidth',2) %Dibuja el rectángulo
hold off
title('Cuadro sobre el video original')
xlabel(['cuadro número ',num2str(a), ' segundo
',num2str(floor((a/30)*10)/10)])

subplot(2,1,2)
    %Estas lineas sirven para evitar que en la representación el cuadrado
    %ocupe valores fuera de la imagen original
    if (ejeX-ancho/2)<0
        ejeX=(ancho/2)+1;
    end
    if (ejeX+ancho/2)>Columnas
        ejeX=Columnas-(ancho/2);
    end
    if (ejeY-alto/2)<0
        ejeY=(alto/2)+1;
    end
    if (ejeY+alto/2)>Filas
        ejeY=Filas-(alto/2);
    end
imshow(I01( ceil(ejeY-alto/2):ceil(ejeY+alto/2), ceil(ejeX-
ancho/2):ceil(ejeX+ancho/2)))
title('Imagen final')

end
```

5.2 Algoritmo de seguimiento caso Parcial

```
%%  Parte 1. Inicialización Programa

clc
clear all
close all

% Declaración de variables

theta=85;    %Ángulo de visión de la cámara
distancia=9; %Distancia a la que está colocada la cámara
%USAMOS ALPHA BETA O FILTRO DE MEDIAS ('0' alphabeta '1' para medias)
filtro=0;

alpha=.95;  %Datos para el filtro Alpha Beta
beta=.05;   %Datos para el filtro Alpha Beta
cuadroInicial=2;    %Debe comenzar en 2 ya que se comparará con la imagen
original
cuadroFinal=1005 ;   %Elegimos hasta donde queremos que funcione el algoritmo
ejeX=0; %Inicialización de la posición del centroide
centx=0;    %Inicialización del centroide X
n=16;    %Número de componentes que utiliza el filtro de medias
c=0;    %Contador para estabilización del centoide, sirve para el filtro de
medias

%% Parte 2. Lectura del video

%Aquí se introduce el video a analizar. En este caso el video es :"ciba.avi"
```

```
video= VideoReader('videos/ciba.avi');

for a=cuadroInicial:cuadroFinal;    %'a' será el numero de iteración
comenzando en 2

% Lectura de las imagenes y Reescalado

I01=imresize(read(video,a),0.5);    %En color
I02=imresize(read(video,a-1),0.5);  %En color siguiente fotograma

%% Parte 3. Transformacion de imagenes a double y resta

I11=rgb2gray(I01);  %Pasamos a blanco y negro
I1=im2double(I11);  %Pasamos a double

I2=rgb2gray(I02);   %Pasamos a blanco y negro
I2=im2double(I2);   %Pasamos a double

I3=I1-I2;

%% Parte 4. Paso a binario

umbral=graythresh(I3);  %Método de otsu
I3(abs(I3)>umbral)=1;   %Pasando a blanco y negro con el umbral de Otsu
I3(I3~=1)=0;    %Si algun valor es distinto a uno será cero

%% Parte 5. Operaciones morfologicas

se=strel('square',2);

I4=imerode(I3,se);
I4=bwmorph(I4,'clean',inf);

I4=imerode(I4,se);
I4=bwmorph(I4,'clean',inf);

I4=bwmorph(I4,'dilate',1);
I4=bwmorph(I4,'clean',inf);

%% Parte 6. Cálculo centroide

[Filas,Columnas]=size(I4);

if centx==0

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Al tener una grabación que abarca todo el campo hacemos lo siguiente

    ancho=tand(theta/2)*distancia;  %Ancho del cuadrado de encuadre
    ancho=round((ancho*Columnas)/15);  % 15 es mitad de 30 metros que es lo
que mide el campo completo

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    alto=Filas;   %Alto del cuadrado de encuadre
    ejeY=alto/2;

    s=regionprops(I4,'centroid');
```

```
else

    s=regionprops(I4(:, ceil(centx-ancho/2):ceil(centx+ancho/2)),'centroid');

end

    %Sirve para evitar las imagenes en la que todos los valores son
    %cero (solo son obtenibles mediante operaciones morfológicas)

if 0==isempty(cat(1,s.Centroid))% si está vacío 0==1 por lo que no entra en
el bucle
    cent=cat(1,s.Centroid);

    if centx==0 %Primera iteración
        centx=mean(cent(:,1)); %Punto centroide X

    else
        centx=centx+(mean(cent(:,1))-(ancho/2)); %Punto centroide X

        %Comprobaciones para evitar que el cuadro se salga de los límites
        if (centx-ancho/2)<0
            centx=(ancho/2)+1;
        end
        if (centx+ancho/2)>Columnas
            centx=Columnas-(ancho/2);
        end
    end
end

%% Estabilizadores

%Sirve para estabilizar la variable X del centroide.

if filtro ==0

    % Método alpha-beta

    if ejeX==0
       ejeX=centx;
    end

    ejeX=ejeX*alpha+centx*beta; %ejeX es el centroide con alpha-beta

else

    %Método suma y medias

    if ejeX==0  %Primera iteración
    ejeX=centx;
    end

    c=c+1;    %Se va actualizando el puntero
    valoresx(c)=centx;    %Llena el vector de centroides
    if c>=n
        c=0;
    end
```

```
    if length(valoresx)==n   %Si el vector está lleno se realiza la operación
        ejeX=sum(valoresx)/n;
    end
end

%% Parte 7. Muestra de resultados

%Se crea un cuadrado a partir de las condiciones que establecimos

x=[ejeX-ancho/2 ejeX+ancho/2 ejeX+ancho/2 ejeX-ancho/2 ejeX-ancho/2];
y=[ejeY-alto/2 ejeY-alto/2 ejeY+alto/2 ejeY+alto/2 ejeY-alto/2];

%Muestra de resultados

figure(1)
subplot(2,1,1)
imshow(I01)
hold on
plot(ejeX,ejeY,'r*')     %Representa el centroide con un asterisco rojo
plot(x,y,'r','linewidth',2) %Dibuja el rectangulo
hold off
title('Cuadro sobre el video original')
xlabel(['Cuadro número ',num2str(a), ' segundo
',num2str(floor((a/30)*10)/10)])

subplot(2,1,2)
imshow(I01(:, ceil(ejeX-ancho/2):ceil(ejeX+ancho/2),:)) %Muestra el producto
final
title('Imagen final')

end
```

5.3 Script Green Red

Este script se utiliza para mostrar la imagen diferencia con dos colores: verde para denotar de donde viene el objeto y rojo para mostrar hacia donde va.

```
clc
clear all
close all

video= VideoReader('videos/Recorte.mp4');

for a=2:400



I01=imresize(read(video,a),0.5);%En color
I02=imresize(read(video,a-1),0.5);% En color siguiente fotograma



I1=rgb2gray(I01);%Pasamos a blanco y negro
I1=im2double(I1);%pasamos a double
```

```
I2=rgb2gray(I02);%pasamos a blanco y negro
I2=im2double(I2);%pasamos a double


I3=I1-I2;

umbral=.2;

I4=I3<-umbral;
I5=I3>umbral;
I6=zeros(size(I4));
I7=cat(3,I4,I5,I6);



subplot(1,2,1)
imshow(I01);
title('imagen original')
subplot(1,2,2)
imshow(I7);
title('imagen diferencias')

end
```

REFERENCIAS

[1] Vigilancia en estaciones de ferrocarril, «ViaLibre,» 2016. [En línea]. Available: http://www.vialibre-ffe.com/noticias.asp?not=19553.

[2] Almacenamiento de videovigilancia ¿cuánto es suficiente?. [En línea]. Available: http://www.seagate.com/es/es/tech-insights/how-much-video-surveillance-storage-is-enough-master-ti/.

[3] Cámarra identifica a una persona por su rostro en un segundo, «enter,» [En línea]. Available: http://www.enter.co/chips-bits/gadgets/camara-identifica-a-una-persona-por-su-rostro-en-un-segundo/.

[4] Sistemas de Localización en Tiempo Real (RTLS), 2013. [En línea]. Available: http://securactiva.com/sistemas-de-localizacion-en-tiempo-real-rtls/.

[5] Comparing RTLS infrastructure, «Kontack.io,» [En línea]. Available: http://cdn2.hubspot.net/hubfs/556697/Comparing%20RTLS%20Infrastructure.pdf?t=1482402388659&utm_campaign=RTLS2&utm_medium=email&_hsenc=p2ANqtz-9OazzoMMweKQ0U9DGPSBkkrn4OX-OfJTtr4VAwOzkEQXzhjKbm2VKTvMoocFzs0x24_6GuV8ilzzj67uM_14bqad7VJw&_hsmi=39532053&utm_c.

[6] Información detallada del producto en la industria ganadera, «RFIDcontrols,» [En línea]. Available: http://www.rfidcontrols.com/rfidcontrols/index.php/sample-sites/ganaderia.html.

[7] What is the difference between RFID and RTLS?, «aimglobal,» [En línea]. Available: http://www.aimglobal.org/?page=rtls_faq#What%20is%20the%20difference%20between%20RFID%20and%20RTLS.

[8] Sistemas de localización en tiempo real (RTLS) aplicado al ámbito hospitalario, «Coitt,» [En línea]. Available: http://www.coitt.es/res/revistas/07b_Sistemas_localizacion.pdf.

[9] F. Jerez, «Sistemas de localización en tiempo real (RTLS) aplicado al ámbito hospitalario,» [En línea]. Available: https://tecnobloggers.wordpress.com/2007/12/11/sistemas-de-localizacion-en-tiempo-real-rtls-aplicado-al-ambito-hospitalario/.

[10] SOLOSHOT, «SOLOSHOT,» [En línea]. Available: https://shop.soloshot.com.

[11] JIGABOT, «JIGABOT,» [En línea]. Available: http://www.jigabot.com/products/.

[12] M. ´. SEE, «movensee,» [En línea]. Available: http://www.movensee.com/pixio-robot-cameraman.html.

[13] Mimicking Human Camera Operators, «Disney Research,» [En línea]. Available: https://www.disneyresearch.com/publication/mimicking-human-camera-operators/.

[14] European CIBA showcase 2016, «http://www.cibalions.com,» [En línea]. Available:

http://www.cibalions.com/showcase.html.

[15] Curve Fitting Toolbox, «MathWorks,» 2015. [En línea]. Available: https://www.mathworks.com/products/curvefitting.html.

[16] Alpha-Beta filter, «Wikipedia,» 2017. [En línea]. Available: https://en.wikipedia.org/wiki/Alpha_beta_filter#cite_note-ReferenceA-3.

[17] BQ, «bq cámaras,» 2017. [En línea]. Available: https://www.bq.com/es/bq-camaras.

[18] Sistemas de Localización en Tiempo Real (RTLS. [En línea]. Available: http://securactiva.com/sistemas-de-localizacion-en-tiempo-real-rtls/.