UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITÈCNICA SUPERIOR DE GANDIA

“Cuantificación de los golpes de tenistas empleando técnicas de análisis de vídeo”

TRABAJO FINAL DE GRADO

Autor/a: Gabriela Vila Andreu

Tutor/a: Jose Ignacio Herranz Herruzo

GANDIA, 15 de Febrero de 2021

Resumen

En el presente proyecto se realiza el desarrollo de un programa de tratamiento de imagen en Matlab que permite cuantificar la carga de trabajo de un tenista y de esta manera obtener estadísticas.

En primer lugar, se establecen la motivación y objetivos del proyecto además de los requerimientos que se deben cumplir a la hora de llevarlo a cabo.

Seguidamente, se hace un estudio de los distintos posibles métodos de tratamiento de imagen que podrían ser útiles para el proyecto.

Posteriormente se procede a describir la metodología empleada para la toma de imágenes y el algoritmo implementado para el proyecto.

Por último, se exponen los resultados obtenidos.

Abstract

The purpose of this project is the development of an image treatment program in Matlab that allows quantification of tennis players loads and get their statistics.

Firstly, the objectives and requirements of the project that must be met are defined.

Next, a study of the different possible image treatment methods that could be useful for the project is carried out.

Afterwards, the algorithm implemented is defined.

Finally, the results are set out.

Índice general

Indice de figuras

1. Introducción

1.1. Motivación

En la actualidad los sistemas de visión artificial se encuentran en plena expansión. El objetivo principal de estos sistemas es adquirir, procesar y analizar imágenes del mundo real con el fin de generar información que pueda ser interpretada por una máquina. Esta tecnología engloba distintas industrias:

- Industria **automovilística** como por ejemplo para coches de conducción autónoma.
- Industria de la **salud** en el que los diagnósticos por imagen tienen una alta importancia.
- Industria **bancaria** donde se realizan lecturas de contratos para análisis de datos.
- Industria **agrícola** en el que se puede llevar un control de campos o productos.
- Industria **deportiva** como por ejemplo para el control de cumplimiento de las reglas deportivas.
- Otros sectores: seguridad, industrial, etc.

Dentro de la industria deportiva el análisis de imagen se encuentra presente en la mayoría de deportes y su principal función es mejorar la calidad del arbitraje ya que existen casos donde el ojo humano encuentra dificultades para esclarecer lo ocurrido. Además de ayudar en el arbitraje también se utiliza para obtener estadísticas.

Centrando el foco únicamente en un deporte como puede ser el tenis el análisis de imagen abarca tareas como la detección de botes de pelotas (para saber si ha botado dentro o fuera de las líneas establecidas), obtener la velocidad a la que viaja la pelota, concretar si el jugador pisa la línea durante el servicio o la obtención de estadísticas en relación al número de golpes (cantidad y tipo) durante el partido. Por ejemplo, cuando se retransmiten partidos por televisión se pueden apreciar repeticiones a cámara lenta para establecer si la pelota ha entrado dentro o fuera, la velocidad de la pelota en un servicio del jugador o las estadísticas de golpes al acabar un set.

Este proyecto abordará el aspecto de obtención de estadísticas a partir del análisis de imagen, se cuantificará los golpes de dos tipos: derecha y revés.

En primer lugar, se estudia la situación actual del análisis de imagen al deporte del tenis, técnicas empleadas. A continuación, se explican los métodos de segmentación, detección y morfología de imágenes necesarios para la comprensión del proyecto.

En segundo lugar, se explica cómo se ha realizado la toma de imágenes con las que se ha experimentado durante el desarrollo del algoritmo que realiza el tratamiento de imagen y se describe el propio algoritmo. Además, se explica la interfaz gráfica desarrollada para facilitar el uso del algoritmo implementado.

Por último, se explican los resultados y conclusiones obtenidas.

1.2. Objetivos

A continuación, se detallan los principales objetivos de este proyecto:

- Investigación de técnicas ya utilizadas para el análisis de golpes de jugadores de tenis.
- Definición de los métodos de análisis de imagen que se utilizarán posteriormente en la implementación del código.
- Aprendizaje de programación en Matlab y aplicación de dichos conocimientos al presente proyecto.
- Obtención de imágenes a partir de un video en una pista de tenis con un jugador realizando golpes.
- Estudio y aplicación de las diferentes técnicas de detección y segmentación de imágenes.
- Diseño e implementación de un código eficiente y robusto que detecte objetos/personas en movimiento dentro de una secuencia de imágenes y sea capaz de analizar y reconocer la posición y golpes del jugador diferenciándolo respecto del fondo.
- Diseño de una interfaz gráfica intuitiva que facilite al usuario el acceso al código implementado.
- Obtención de estadísticas de golpes del jugador.
- Detectar el mayor número de golpes posibles.
- Conseguir las estadísticas de golpes del jugador con el mínimo error posible.
- Reducir el tiempo de ejecución del algoritmo para sea el menor posible.

1.3. Requerimientos

En este apartado se recoge los requerimientos que deberá cumplir el código implementado en el proyecto para que cumpla el objetivo de cuantificar la carga de un tenista a partir del análisis de un video.

1.3.1. Fuentes de información

La información necesaria para el funcionamiento del programa provendrá de archivos de video grabados con una cámara de resolución 1280x720 por el usuario.

1.3.2. Requerimientos de usuario

El usuario esperará que el programa contenga las siguientes características:

- El programa proporcionará un entorno gráfico sencillo, con una representación visual y clara de los elementos que lo componen.
- El interfaz del programa permitirá al usuario la selección del archivo (video) que desea analizar con facilidad.
- El interfaz del programa facilitará la ejecución del código para que el usuario pueda analizar el video elegido.
- El interfaz del programa proporcionará los resultados de una manera visual y sencilla de comprender.

- En caso de que se produzca un error el usuario deberá ser debidamente informado.

1.3.3. Requerimientos de software

Para el funcionamiento de la herramienta a nivel lógico se deberá contar con:

- Un sistema de cómputo numérico llamado Matlab que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

La figura 1.1 plantea un diagrama de la estructura a grandes rasgos del programa.

Figura 1.1. Diagrama estructura del programa

2. Estado del arte

En este apartado se analizará la situación de la tecnología que afecta a la base teórica del siguiente trabajo. Primeramente, se contará la aplicación de análisis de imagen en algunos casos reales y de actualidad y posteriormente.

2.1. Aplicación de métodos de análisis de imagen en casos actuales

Aunque podríamos pensar que el uso y aplicación de métodos de análisis y procesamiento de imágenes es reciente, lo cierto es que se remonta a la década de 1950, es decir, casi al mismo tiempo de que se comenzara a hablar de computación a escala industrial y de que se empezaran a desarrollar los primeros lenguajes de programación de alto nivel. Es ahora, cuando somos capaces de recopilar y tratar enormes cantidades de datos y ponerlos a disposición de la inteligencia artificial, especialmente a través de campos como las redes neuronales y el aprendizaje profundo, cuando el campo de la visión artificial y análisis de imágenes está aplicándose con un éxito sin precedentes en multitud de campos y disciplinas, no solo del mundo científico y académico, sino también de la vida real, llegándose a batir la capacidad visual del ser humano.

Algunos de los campos en los que actualmente se está investigando, desarrollando y aplicando los métodos inteligentes de reconocimiento y tratamiento de imágenes son:

- la salud, permitiendo el diagnóstico de enfermedades a través de las imágenes recogidas en pruebas de radiodiagnóstico, o incluso directamente de imágenes obtenidas por lo potentes microscopios electrónicos actuales,
- la manufactura de productos, involucrándose en los procesos de control de calidad para la detección de fallos en las cadenas de producción de todo tipo de objetos llegando a ofrecer soluciones perfectas para la fabricación de ciertos productos con cero defectos,
- la seguridad, consiguiendo el reconocimiento biométrico a través de imágenes de las personas, como la huella digital, la cara o el iris,
- la automoción, evolucionando el coche inteligente y especialmente la conducción autónoma, gracias a la detección y reconocimiento de la carretera, los obstáculos, los peatones, resto de vehículos circulantes, etc., así como permitiendo mejorar la gestión del tráfico,
- la ciencia y la investigación en múltiples campos, como la biología, el sector aeroespacial, la medicina, el sector farmacéutico, la astronomía, etc.

El avance que se ha producido en el procesamiento de imágenes es enorme e imparable. La tecnología comienza a ser lo suficientemente flexible, ágil e industrializada como para que en momentos actuales, en los que son necesarias aplicaciones urgentes basadas en estas técnicas, el tiempo de respuesta para poner en funcionamiento su uso sea mínimo. Como respuesta a la pandemia del Covid-19, se han aplicado estas técnicas en casos como

- el control de aforos en lugares públicos, para garantizar que las personas puedan cumplir con las normal de distanciamiento,

- la detección de la temperatura corporal de las personas mediante soluciones termográficas, para controlar posibles casos de infección y controlar a personas que puedan contagiar a otras personas.

El campo del análisis de imágenes seguirá siendo clave en el futuro del desarrollo de la visión artificial. Según se vaya trabajando en la mejora de los sistemas de entrenamiento de los algoritmos involucrados y mejorando así la cantidad de información que se pueda interpretar a partir de las imágenes disponibles, la visión artificial podrá desempeñar una gama más amplia de funciones. A modo de ejemplo, citamos uno de los grandes retos futuros: el análisis de imágenes obtenidas mediante satélite, que permitirá controlar la planificación urbanística, los flujos migratorios de las personas o el cambio climático.

Sin duda, el campo del análisis inteligente de imágenes permitirá alcanzar logros aún mayores si se combinan con otras disciplinas y tecnologías, lo que permitirá a la inteligencia artificial general seguir avanzando hacia su objetivo de igualar o batir a la inteligencia humana, algo que ya se ha alcanzado en muchos de los campos vistos anteriormente [1].

2.2. Software comerciales

Recientemente ha aparecido un nuevo software que ofrece análisis y entrenamientos a tiempo real únicamente con la cámara de un teléfono móvil, su nombre es Swing Vision.

Swing Vision no solo es capaz de analizar el tipo de golpe, sino que además es capaz de identificar la zona de la pista en la que has golpeado y la velocidad del golpe. A continuación, en la figura 2.1 y 2.2 se muestran algunos ejemplos.

Figura 2.1. Ejemplo 1 del software Swing Vision

Figura 2.2. Ejemplo 2 del software Swing Vision

Este software ha sido diseñado por los equipos de inteligencia artificial de Tesla y Apple y ha contado con el apoyo de ex jugadores profesionales de tenis como Andy Roddick y James Blake (que llegaron a alcanzar número uno y cuatro del mundo respectivamente).

Swing Vision actualmente funciona únicamente sobre IOS tanto en Iphone como en Ipad y te ofrece seguimiento del golpe y video análisis [2].

3. Métodos de análisis de imagen

En este punto se estudiarán los distintos métodos de tratamiento de imagen que se utilizan durante el desarrollo del presente proyecto.

3.1. Filtro de Kalman

El filtro de Kalman [3] es un algoritmo que estima el estado de un sistema a partir de datos medidos. El algoritmo del filtro es un proceso de dos pasos: el primer paso predice el estado del sistema y el segundo paso utiliza mediciones ruidosas para refinar la estimación del estado del sistema.

Actualmente existen varias variantes del filtro Kalman original. Estos filtros se utilizan mayormente para aplicaciones que dependen de las estimaciones, incluida la visión artificial, los sistemas de guía y navegación, la econometría y el procesamiento de señales.

En las aplicaciones de visión artificial los filtros de Kalman se utilizan para el seguimiento de objetos con el fin de predecir la futura ubicación de este mismo, para tener en cuenta el ruido en donde se ha detectado el objeto y para ayudar a asociar varios objetos con sus tracks correspondientes.

3.2. Gaussian Mixture Model

El *Gaussian Mixture Model* [4] es una función que se compone de varios gaussianos, cada uno identificado por k ∈ {1,…, K}, donde K es el número de grupos de nuestro conjunto de datos. Cada gaussiano k en la mezcla se compone de los siguientes parámetros:

- Un μ medio que define su centro.
- Una covarianza Σ que define su ancho. Esto equivaldría a las dimensiones de un elipsoide en un escenario multivariado.
- Una probabilidad de mezcla π que define qué tan grande o pequeña será la función gaussiana.

Se puede ver un ejemplo de estos parámetros en la siguiente figura:

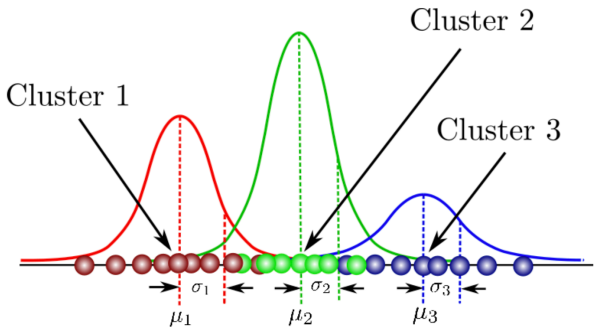

Figura 3.1. Ejemplo parámetros de un gaussiano

Por lo que un *Gaussian Mixture Model* es un modelo probabilístico que asume que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones gaussianas caracterizadas por sus parámetros.

En el presente proyecto el Gaussian Mixture Model se utilizará para la estimación de fondo.

3.3. Algoritmo K-Means

3.3.1. Definición Clustering

El **clustering** consiste en la agrupación automática de datos. Al ser un **aprendizaje no-supervisado**, no hay una respuesta correcta. Esto hace que la evaluación de los grupos identificados sea un poco subjetiva. Las técnicas de clustering intentan descubrir cuál es el mejor agrupamiento de los datos [5].

3.3.2. Definición algoritmo K-Means

El algoritmo de clustering más utilizado es K-Means. Tiene una muy buena escalabilidad con grandes cantidades de datos. Para utilizar K-Means debemos especificar el número de grupos que queremos encontrar. A este número de grupos se le denomina K [6].

El algoritmo K-Means sigue los siguientes pasos [6]:

1. **Inicialización:** se elige la localización de los centroides de los K grupos aleatoriamente.
2. **Asignación:** se asigna cada dato al centroide más cercano.

3. **Actualización:** se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo.
4. Los pasos 2 y 3 se siguen iterativamente hasta que no haya más cambios.

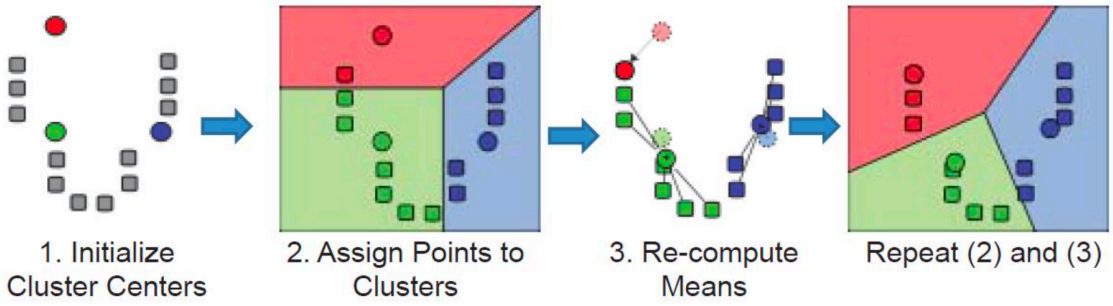

Figura 3.2. Ejemplo Algoritmo K-Means

En el presente proyecto el algoritmo de K-Means se utilizará para el seguimiento del jugador por la pista de tenis.

3.4. Morfología binaria

La morfología es una herramienta matemática para extraer componentes de una imagen y realizar operaciones relativas a su forma.

Se puede utilizar antes o después de la segmentación de la imagen:

• Antes de la segmentación de objetos se emplean las técnicas de morfología de grises.

• Después de la segmentación de objetos se aplica la **morfología binaria**.

En este proyecto se utilizará la morfología tras segmentación de objetos, por lo que será morfología binaria. La aplicación más importante de la morfología binaria es mejorar los resultados de la segmentación [7].

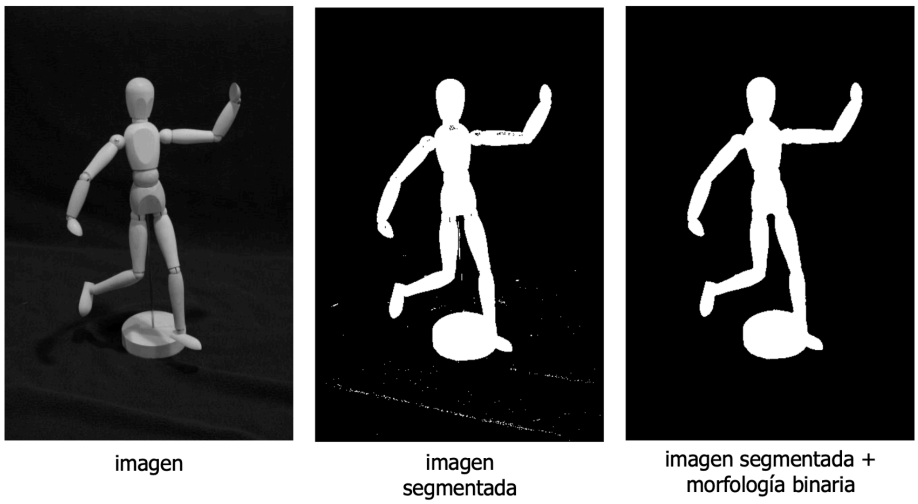

Figura 3.3. Ejemplo morfología binaria

3.4.1. Operadores morfológicos

Un operador morfológico depende del elemento estructural (structuring element) del operador. El elemento estructural define los píxeles vecinos (neighborhood) que se van a tener en cuenta en la operación [7].

3.4.2. Operadores aditivos

Los operadores aditivos convierten a “1” el píxel procesado si sus píxeles vecinos cumplen ciertas condiciones predeterminadas [7].

El operador aditivo más empleado es DILATE, el cual se aplica si al menos uno de sus ocho vecinos es “1” (ver ejemplo de la figura 3.4). El resultado es la dilatación de los objetos.

Otro operador aditivo es el FILL, el cual se aplica si al menos uno de sus cuatro vecinos es “1” (ver ejemplo de la figura 3.5). El resultado es la eliminar pixeles sueltos a “0” dentro de la figura.

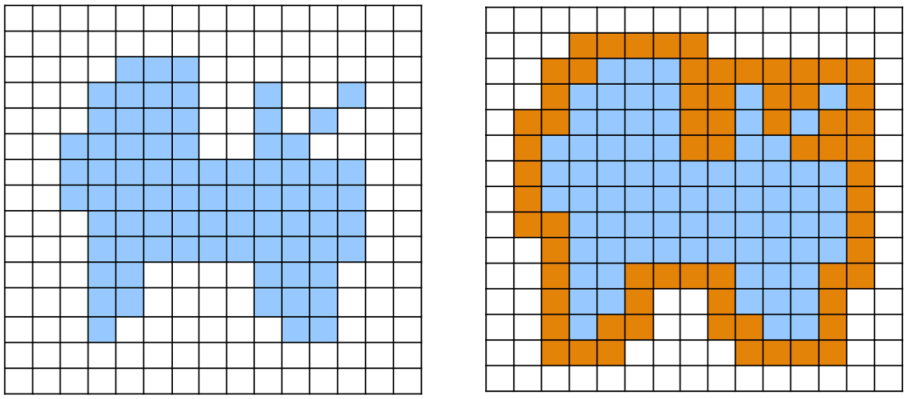

Figura 3.4. Ejemplo Dilate

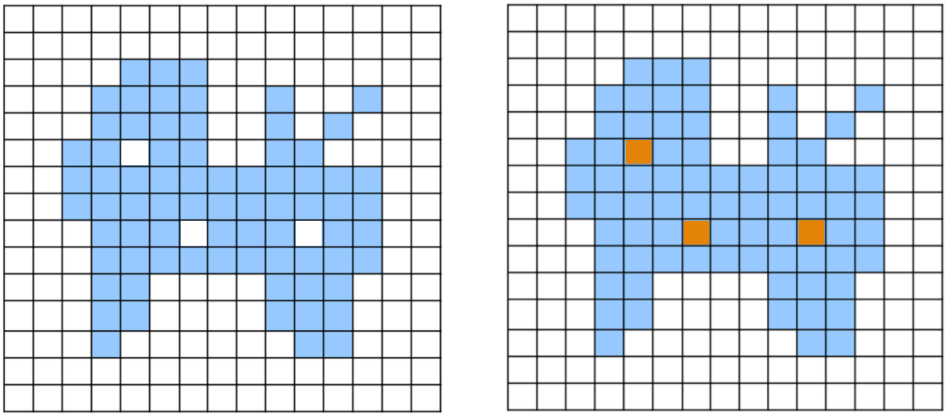

Figura 3.5. Ejemplo Fill

3.4.3. Operadores sustractivos

Los operadores sustractivos convierten a “0” el píxel procesado si sus píxeles vecinos cumplen ciertas condiciones predeterminadas [7].

El operador sustractivo más utilizado es ERODE, el cual se aplica si al menos uno de sus ocho vecinos es “0”. En este caso el resultado es la erosión de los objetos.

3.4.4. Open & Close

Estos dos operadores combinan operaciones simples para obtener resultados más útiles.

El operador CLOSE consiste en concatenar una operación Dilate seguida de una Erode. Su efecto es el de rellenar huecos en los objetos y entrantes estrechos. En la figura 3.6 se puede ver un ejemplo de aplicación.

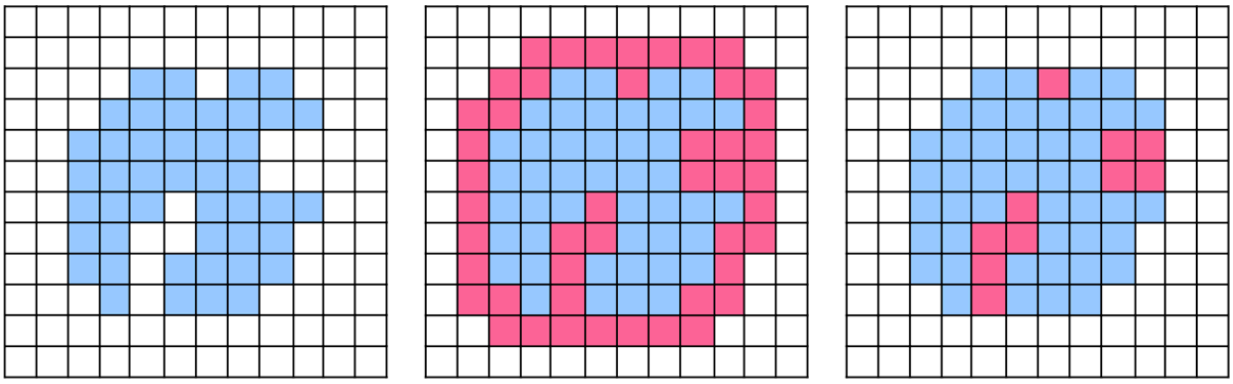

Figura 3.6. Ejemplo Close

El operador OPEN consiste en concatenar una operación Erode seguida de una Dilate. Su efecto es el de eliminar salientes estrechos de los objetos, siendo la operación complementaria al Close, aunque no inversa. En la figura 3.6 se puede ver un ejemplo de aplicación.

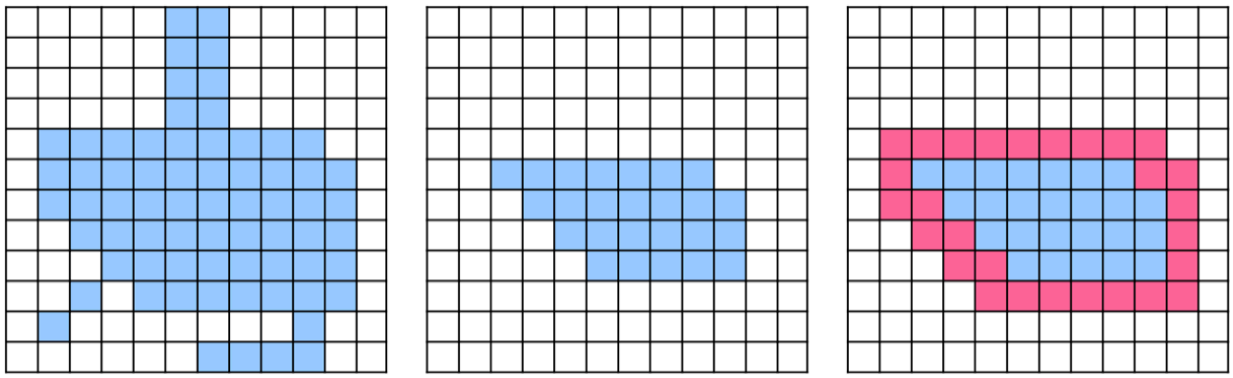

Figura 3.7. Ejemplo Open

La utilización del Open y el Close mejora una segmentación defectuosa tal y como se muestra en la figura 3.8 [7].

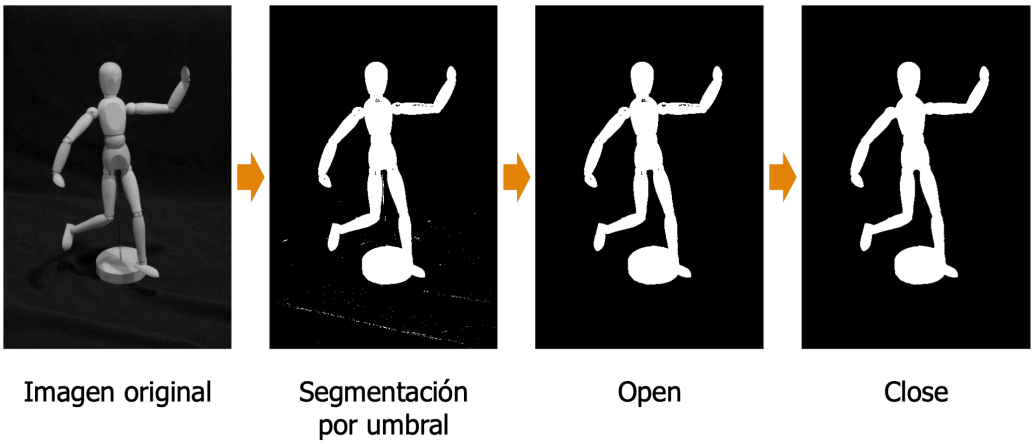

Figura 3.8. Ejemplo mejora de la segmentación con los métodos Open y Close.

3.5. CNNs (Redes Neuronales convolucionales)

El reconocimiento o clasificación es la última fase de un sistema de visión artificial. Su objetivo es proporcionar la categoría del objeto a partir de los descriptores asociados a dicho objeto.

3.5.1. Redes neuronales

Las redes neuronales artificiales son un modelo inspirado en el funcionamiento del cerebro humano, un sistema de clasificación masiva paralelo que intenta simular el comportamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida.

El funcionamiento se basa en que estas redes (conjunto de nodos conectados entre sí) reciben una serie de valores de entrada y cada una de estas entradas llega a un nodo llamado neurona. Las neuronas de la red están a su vez agrupadas en capas que forman la red neuronal. Cada una de las neuronas de la red posee a su vez un peso, un valor numérico, con el que modifica la entrada recibida. Los nuevos valores obtenidos salen de las neuronas y continúan su camino por la red. Este funcionamiento puede observarse de forma esquemática en la siguiente imagen.

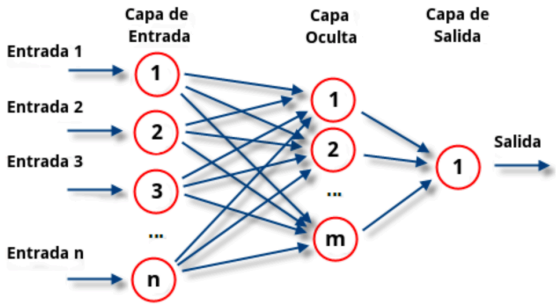

Figura 3.9. Funcionamiento redes neuronales.

Una vez que se ha alcanzado el final de la red se obtiene una salida que será la predicción calculada por la red. Cuantas más capas posea la red y más compleja sea, también serán mas complejas las funciones que pueda realizar.

Para conseguir que una red neuronal realice las funciones deseadas, es necesario entrenarla. El entrenamiento de una red neuronal se realiza modificando los pesos de sus neuronas para que consiga extraer los resultados deseados. Para ello lo que se hace es introducir datos de entrenamiento en la red, en función del resultado que se obtenga, se modifican los pesos de las neuronas según el error obtenido y en función de cuanto haya contribuido cada neurona a dicho resultado. Este método es conocido como Backpropagation o propagación hacia atrás. Con este método se consigue que la red aprenda, consiguiendo un modelo capaz de obtener resultados muy acertados incluso con datos muy diferentes a los que han sido utilizados durante su entrenamiento [8].

3.5.2. Redes Neuronales convolucionales

Dentro de las redes neuronales, la red neuronal convolucional (CNNs) es una de las principales categorías para realizar reconocimiento y clasificaciones de imágenes.

La CNN es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al cortex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y “ver”. Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que **las primeras capas pueden detectar líneas, curvas y se van especializando** hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal.

La red neuronal debe aprender por sí sola a reconocer una diversidad de objetos dentro de imágenes y para ello necesitará una gran cantidad de imágenes para que la red pueda captar sus características únicas -de cada objeto- y a su vez, poder generalizarlo -esto es que pueda reconocer como gato tanto a un felino negro, uno blanco, un gato de frente, un gato de perfil, gato saltando, etc [9].

4. Diseño y desarrollo del proyecto

4.1. Equipo utilizado

Para el desarrollo del proyecto se han utilizado tanto elementos físicos como elementos de software, imprescindibles para llevar a cabo el trabajo. En este apartado se van a presentar los distintos componentes que se han utilizado, así como las distintas herramientas que nos han permitido lograr el correcto funcionamiento de la cuantificación de los golpes del tenista.

4.1.1. Elementos físicos

En lo que respecta al equipo físico se han utilizado un ordenador, una videocámara deportiva, una pista de tenis, un jugador de tenis con una raqueta y un soporte para colocar la videocámara la pista de tenis. A continuación, se van a explicar más detalladamente cuáles son las especificaciones de cada uno de estos elementos.

4.1.1.1.Ordenador:

Se ha utilizado un ordenador MacBook Pro de 2017 que cuenta con las siguientes especificaciones:

- Procesador de doble núcleo a 2,3 GHz Intel Core i5
- Memoria RAM de 8 GB 2133 MHz LPDDR3
- Unidad de memoria sólida de 256 GB
- Sistema Operativo macOS X Catalina versión 10.15.7
- Una resolución de pantalla de 2560 x 1600
- Tarjeta gráfica Intel Iris Plus Graphics 640 1536 M
- Conexiones thunderbolt 3

Figura 4.1. Ordenador utilizado

4.1.1.2.Video cámara deportiva:

La videocámara que se ha utilizado para grabar los vídeos es la una GoPro Hero5 session, ya que permite realizar vídeos de alta calidad gracias a su cámara de 10 Megapíxeles. También permite con su función Wifi controlar la grabación desde el móvil o la Tablet mediante la App GoPro app disponible tanto para Android como para IOS. Asimismo, su pequeño tamaño y su ligereza la convierten en la cámara idónea para llevar a cabo este tipo de proyecto.

Principales características de la cámara:

- Sensor 10 Megapíxeles
- Vídeo: 4K (30 fps), 1080p (90 fps), 720p (120 fps)
- Estabilización de imagen: Electrónica
- Modos de disparo: Foto, video, ráfaga, bucle.
- Resistencia al agua: 10 m
- Batería: 1.100 mAh
- Micrófonos: 3
- Conectividad: Wifi, USB tipo-C
- Almacenamiento: microSD
- Reducción de ruido: si

Figura 4.2. Cámara deportiva utilizada

4.1.1.3.Soporte para la cámara

Para poder grabar en el escenario elegido para este proyecto (pista de tenis) se colocó la cámara en la valla de tela metálica que rodea la pista a partir de un gancho metálico con forma de “S” (figura 4.2.) que se enganchaba a la valla y al soporte de brazo articulado corto de esta cámara (figura 4.3).

Figura 4.3. Gancho metálico con forma de “S”

Figura 4.4. Soporte brazo articulado corto de GoPro

Gracias a la articulación de este soporte se le podrá jugar con la inclinación de la cámara y de esta manera escoger el mejor plano posible.

4.1.2. Software

En lo que se refiere al software utilizado, se decidió utilizar MATLAB R2018b ya que ofrece numerosas ventajas y facilidades. Además, ha sido la herramienta más utilizada a lo largo del grado por lo que ya se cuenta con cierta experiencia y conocimientos a la hora de programar y utilizar las diferentes funciones de esta aplicación.

Matlab es una herramienta de Software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. Está disponible para distintos sistemas operativos como pueden ser Windows, MacOS o Linux. Además, MATLAB es un lenguaje de alto rendimiento para informática técnica. Integra computación, visualización y programación en

un entorno fácil de usar donde los problemas y las soluciones se expresan en notación matemática familiar.

Los toolbox que posee ofrecen funciones que permiten explorar los datos de manera interactiva y generar código de manera automática, lo que significa que no es necesario programar desde cero y simplifica la tarea. En particular para el desarrollo de este proyecto se utilizaron principalmente los toolbox de Image Processing and Computer Vision.

Figura 4.5. Logo Matlab

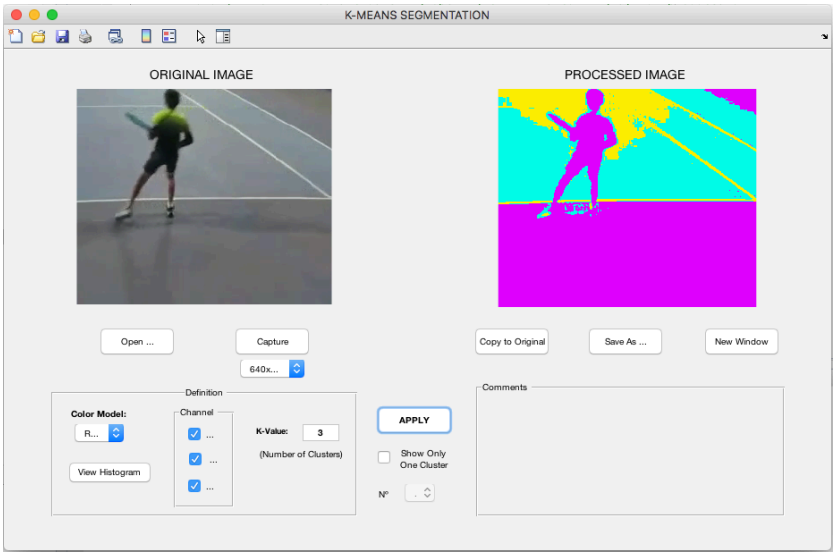

Figura 4.6. Ejemplo de utilización de un toolbox de Matlab con el algoritmo de segmentación K-Means

4.2. Toma de imágenes

4.2.1. Lugar

Para adquirir las imágenes necesarias para el proyecto es necesario desplazarse hasta una pista de tenis y contar con al menos dos jugadores (ya que uno será el que grabaremos y el otro es necesario para lanzar pelotas o que las devuelva). En este caso me desplacé a un club de tenis y me uní en la pista a uno de los entrenamientos de competición de jugadores entre 10 y doce años.

La pista de tenis es una superficie rectangular y lisa que puede estar constituida de varios materiales, en nuestro caso era de cemento. Las dimensiones de la pista son 23.77 metros de longitud por 8.23 metros de ancho (en caso de jugar dobles sería 10.97 metros el ancho). La red divisoria debe encontrarse a una altura de 0,914 m en el centro y 1,07 m en los postes que la sostienen. Esta red divide la pista en dos partes, una para cada jugador o equipo. El recinto en el que se encuentra la pista suele estar delimitado por una verja metálica que rodea la pista con bastante margen, esto se debe a que el jugador tiene que poder desplazarse en caso de que la pelota botara cerca de alguno de los límites.

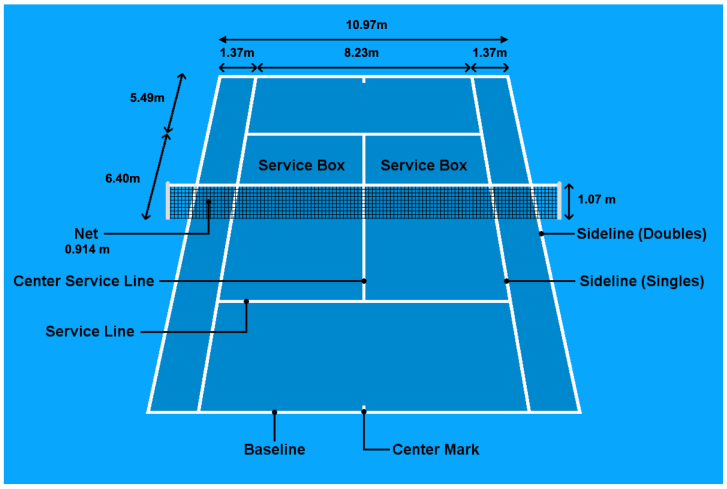

Figura 4.7. Pista de tenis utilizada

4.2.2. Personas en la pista

En la pista se encontraban tres niños y el entrenador. En el momento de la toma de imágenes, que se realizó sin influir en el entrenamiento, únicamente se encontraban jugando dos de los niños, el tercero se encontraba retirado en el fondo opuesto de donde se realizó la toma de imágenes. El entrenador se sitúa en este caso a media pista en el lado sobre el que se tomaron las imágenes poniendo la pelota en juego. Se muestra un esquema en la figura 4.8.

El jugador que va a ser grabado no debe contar con una ropa específica, aunque sí favorece al análisis del video que esta sea de un color distinto al de la pista. En este caso el color de la pista es azul claro en la zona de juego y azul oscuro el resto. Uno de los jugadores analizados llevaba una camiseta amarilla y un pantalón negro como en la figura 4.9.

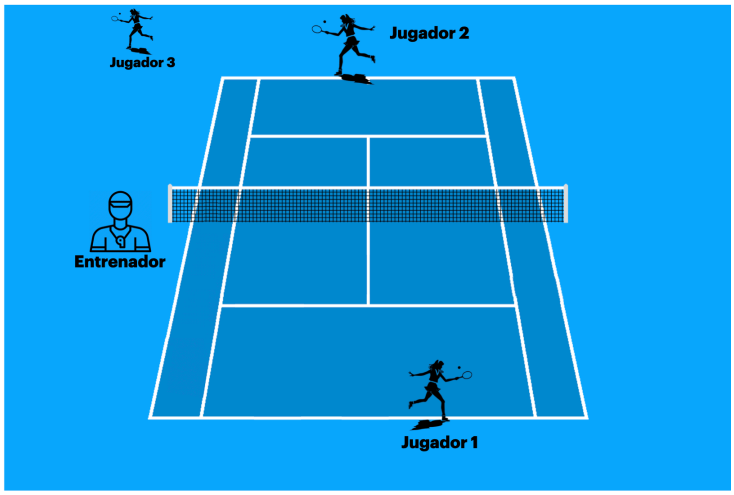

Figura 4.8. Esquema personas en la pista de tenis

Figura 4.9. Ejemplo vestimenta jugador

4.2.3. Plano

Una vez en la pista hay que escoger dónde y cómo colocar la cámara para la toma de imágenes.

En este caso primero se grabó desde el lateral de la pista tal y como se muestra en la figura 4.10. Desde esta perspectiva se encontraron dificultades para seguir tanto al jugador como a la pelota que se salían del plano con frecuencia. Además, la silueta del jugador que se obtenía al analizar esta imagen era difícil de distinguir el tipo de golpe ya que algunas veces era el mismo jugador el que con su propio cuerpo tapaba a la vista de la cámara la raqueta y la pelota.

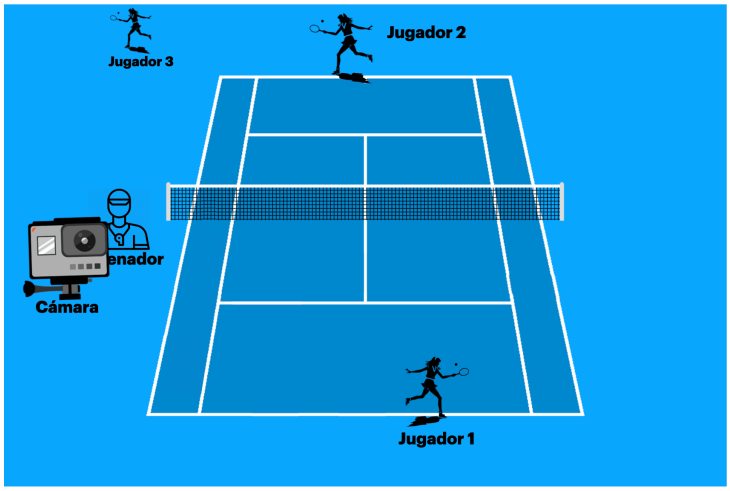

Figura 4.10. Plano fallido

Finalmente se decidió grabar desde el fondo de la pista tal y como se muestra en la figura 4.11 (detrás del jugador 1, que es el elegido para el análisis). De esta manera se consigue coger al jugador por toda “su” mitad de la pista, tanto a lo ancho como en profundidad, sin que se salga del plano de la cámara. Además, desde este ángulo no se producen casi imágenes en las que el jugador tape con su propio cuerpo la raqueta o la pelota. La silueta del jugador es mucho más clara.

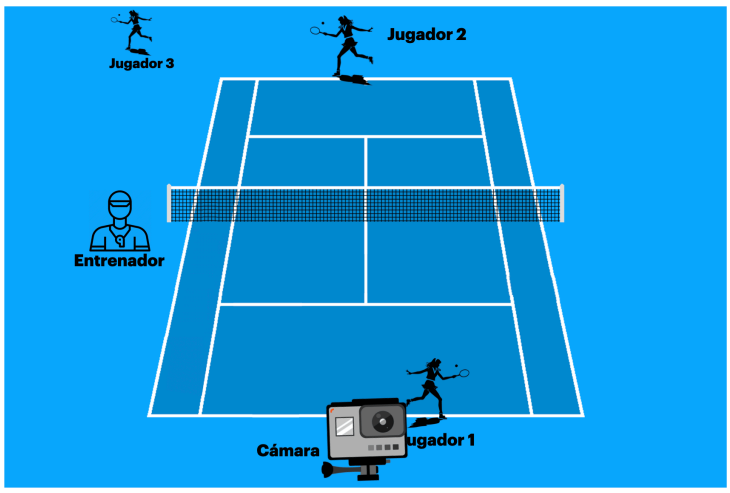

Figura 4.11. Plano correcto

Para colocar la cámara se probaron varias inclinaciones con el objetivo de elegir un plano de fondo homogéneo. Finalmente, la cámara se colocó a una altura de 2.8 metros colgada de la verja que limita por detrás la pista gracias al gancho metálico con forma de “S” figura 4.2. Se le dio una pequeña inclinación hacia abajo con la ayuda del soporte de brazo articulado corto de GoPro figura 4.3. Al darle inclinación a la cámara hacia abajo se consigue que el plano de fondo sea mayoritariamente la pista que es un fondo bastante homogéneo. Si no se le diera esta inclinación el fondo de la imagen sería una mezcla entre la pista, la red, el otro jugador y la verja del otro fondo, lo que dificultaría el análisis de la imagen.

Figura 4.12. Colocación de la cámara en la pista de tenis

4.2.4. Iluminación

La iluminación que más hubiera favorecido el análisis de imagen de este proyecto hubiera sido un día soleado alrededor del medio día para que el sol estuviera en una posición alta. Que el sol esté en esta posición favorecería a que las sombras producidas por el jugador son mínimas y no pudiera llevar a confusión a la hora de realizar el análisis.

Sin embargo, esta no es la situación más común en los entrenamientos de los jugadores no profesionales. Los entrenamientos suelen darse en horarios de tarde entre las 5pm y las 10pm con lo que la iluminación tiene una alta probabilidad de ser artificial y por lo que se ha decidido realizar las pruebas con este tipo de iluminación.

4.3. Definición del algoritmo

Para poder definir el algoritmo desarrollado para este proyecto lo vamos a dividir en dos etapas. La primera etapa consistirá en el algoritmo de seguimiento, basado en una función de Matlab, que muestra cómo realizar la detección automática y el seguimiento de objetos en movimiento en un video desde una cámara fija. La segunda etapa es la que se ha desarrollado con más profundidad y que a su vez se dividirá en dos partes: detección del jugador y la clasificación de los golpes. En la figura 4.13 se puede observar el diagrama del algoritmo desarrollado.

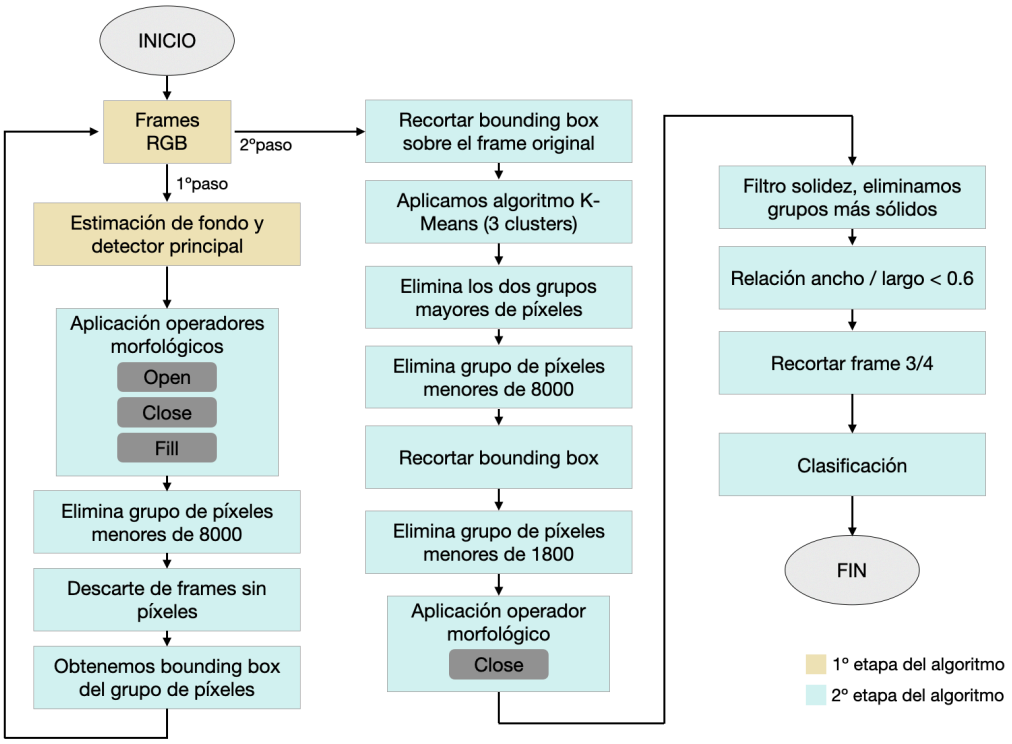

Figura 4.13. Diagrama algoritmo desarrollado

Lo que pretende el algoritmo desarrollado es realizar la detección automática y seguimiento del jugador que se va desplazando por la pista. El problema de la detección se plantea en dos puntos importantes:

- Detectar al jugador en movimiento en cada frame diferenciándolo tanto del fondo como de las sombras de su propia figura.
- Asociar que un grupo de frames con la detección de la figura del jugador corresponden a un mismo golpe de tenis en el tiempo.

La detección de objetos en movimiento utiliza un algoritmo de resta de fondo basado en modelos de mezcla gaussianos. Las operaciones morfológicas se aplican a la máscara de primer plano resultante para eliminar el ruido. Finalmente, el análisis de manchas detecta grupos de píxeles conectados, que probablemente correspondan a objetos en movimiento.

La asociación de detecciones al mismo objeto se basa únicamente en el movimiento. El movimiento de cada *track* se estima mediante un filtro de Kalman [3]. El filtro se utiliza para predecir la ubicación del *track* en cada fotograma y determinar la probabilidad de que cada detección se asigne a cada *track*.

El mantenimiento de *tracks* se convierte en un aspecto importante de este algoritmo. En cualquier fotograma dado, algunas detecciones pueden asignarse a *tracks*, mientras que otras detecciones y *tracks* pueden permanecer sin asignar. Los *tracks* asignados se actualizan utilizando las detecciones correspondientes. Los *tracks* no asignados se marcan como invisibles. Una detección no asignada inicia un nuevo *track*.

Cada *track* lleva la cuenta del número de fotogramas consecutivos, donde permaneció sin asignar. Si el recuento excede un umbral especificado, el ejemplo asume que el objeto abandonó el campo de visión y borra el *track* [10].

A continuación, se describirá la primera etapa del algoritmo desarrollado, que es el algoritmo de seguimiento, y posteriormente la segunda etapa.

4.3.1. Algoritmo de seguimiento

4.3.1.1.Descripción del algoritmo de seguimiento

El algoritmo de detección de movimiento comienza creando un sistema de objetos y una matriz vacía que servirán para la lectura del video, la detección de los objetos en movimiento y mostrar los resultados. A continuación, inicializa una variable contador y, por último, se crea un bucle principal desde el que realiza llamadas a una serie de funciones que son las que proporcionarán el resultado de detección final. En la figura 4.14 se puede ver un ejemplo [10].

```
47  % Create System objects used for reading video, detecting moving objects,
48  % and displaying the results.
49  obj = setupSystemObjects();
50
51  tracks = initializeTracks(); % Create an empty array of tracks.
52
53  nextId = 1; % ID of the next track
54
55  % Detect moving objects, and track them across video frames.
56  while hasFrame(obj.reader)
57      frame = readFrame(obj.reader);
58      [centroids, bboxes, mask] = detectObjects(frame);
59      predictNewLocationsOfTracks();
60      [assignments, unassignedTracks, unassignedDetections] = ...
61          detectionToTrackAssignment();
62
63      updateAssignedTracks();
64      updateUnassignedTracks();
65      deleteLostTracks();
66      createNewTracks();
67
68      displayTrackingResults();
69  end
```

Figura 4.14. Ejemplo algoritmo detección de movimiento en Matlab

A continuación, se definen y explican las funciones que son llamadas desde el algoritmo:

- **initializeTraks**

  Esta función crea una matriz de *tracks* donde cada *track* es una estructura que representa un objeto en movimiento en el video. El propósito de la estructura es mantener el estado de un objeto rastreado. El estado consiste en información que se utiliza en la detección para rastrear la asignación, rastrear la terminación y visualizar el resultado obtenido.

La estructura que representa el objeto en movimiento contiene los siguientes campos:

- **id:** ID entero del *track*.
- **bbox:** cuadro delimitador actual del objeto; usado para visualizar.
- **kalmanFilter:** que hace referencia a un objeto del filtro de Kalman utilizado para la detección de objetos en movimiento.
- **edad:** el número de fotogramas desde que se detectó el *track* por primera vez.
- **totalVisibleCount:** el número total de fotogramas en los que el *track* fue detectado (visible).
- **consecutiveInvisibleCount:** el número de fotogramas consecutivos para los que no se detectó *track* (invisible).

Las detecciones ruidosas tienden a resultar en *tracks* de corta duración. Por esta razón, el ejemplo solo muestra un objeto después de que se haya rastreado durante cierto número de fotogramas. Esto sucede cuando totalVisibleCount supera un umbral especificado.

Cuando no hay detecciones asociadas con un *track* durante varios fotogramas consecutivos, el ejemplo supone que el objeto ha abandonado el campo de visión y elimina el *track*. Esto sucede cuando ConsecutiveInvisibleCount supera un umbral especificado. Un *track* también puede eliminarse como ruido si se rastrea durante un período breve y se marca como invisible para la mayoría de los fotogramas.

- **detectObjects**

  Esta función devuelve los centroides y los bounding box de los objetos detectados, entendiendo por bounding box las coordenadas del borde rectangular que encierra completamente una imagen digital colocada sobre una pantalla. También devuelve la máscara binaria, que tiene el mismo tamaño que el marco de entrada. Los píxeles con un valor de 1 corresponden al primer plano y los píxeles con un valor de 0 corresponden al fondo.

  La función realiza la segmentación de movimiento utilizando el detector de primer plano. La detección de objetos en movimiento utiliza un algoritmo de resta de fondo basado en modelos de mezcla gaussianas. A continuación, realiza operaciones morfológicas en la máscara binaria resultante para eliminar píxeles ruidosos y rellenar los agujeros en las manchas restantes. Finalmente, el análisis de grupos de píxeles detecta grupos de píxeles conectados, que probablemente correspondan a objetos en movimiento.

- **predictNewLocationsOfTracks**

  Utiliza el filtro de Kalman para predecir el centroide de cada *track* en el fotograma actual y actualiza su bounding box en consecuencia.

- **detectionToTrackAssignment**

  La asignación de objetos detectados en el fotograma actual a *tracks* existentes se realiza minimizando el coste. El coste se define como la probabilidad logarítmica negativa de una detección correspondiente a un *track*.

  La función consta de dos pasos:

  - **Paso 1:** Calcular el coste de asignar cada detección a un *track* utilizando el método de Matlab “vision.KalmaFilter” para sistemas de objetos. El coste tiene en cuenta la distancia euclidiana entre el centroide predicho del *track* y el centroide de la detección. También incluye la fiabilidad de la predicción, que es proporcionada por el

filtro de Kalman. Los resultados se almacenan en una matriz MxN, donde M es el número de *tracks* y N es el número de detecciones.

- **Paso 2:** Resolver el problema de asignación representado por la matriz de costes usando la función assignDetectionsToTracks. La función toma la matriz de costes y el coste de no asignar detecciones a un *track*.

El coste de no asignar una detección a un *track* depende del rango de valores devueltos por el método de distancia de visión de KalmanFilter. Este valor debe ajustarse experimentalmente. Establecerlo demasiado bajo aumenta la probabilidad de crear un nuevo *track* y puede resultar en la fragmentación de dicho *track*. Establecerlo demasiado alto puede resultar en un solo *track* correspondiente a una serie de objetos en movimiento separados.

La función assignDetectionsToTracks utiliza la versión de Munkres[11] del algoritmo húngaro para calcular una asignación que minimiza el coste total. Devuelve una matriz M x 2 que contiene los índices correspondientes de los *tracks* asignados y las detecciones en sus dos columnas. También devuelve los índices de *tracks* y detecciones que quedaron sin asignar.

- **updateAssignedTracks**

  Esta función actualiza cada *track* asignado con la detección correspondiente. Llama al método correcto de visión KalmanFilter para corregir la estimación de ubicación. A continuación, se almacena el nuevo bounding box y se aumenta la edad del *track* y el recuento total visible en 1. Finalmente, la función establece el recuento invisible en 0.

- **updateUnassignedTracks**

  Esta función marca el *track* no asignado como invisible y aumenta 1 su antigüedad.

- **deleteLostTracks**

  Esta función elimina los *tracks* que han sido invisibles durante demasiados fotogramas consecutivos. También elimina los últimos *tracks* creados que han sido invisibles durante demasiados fotogramas.

- **createNewTracks**

  Esta función crea nuevos *tracks* de detecciones sin asignar. Se supone que cualquier detección no asignada es el comienzo de un nuevo *track*. En la práctica se pueden utilizar otras señales para eliminar detecciones ruidosas, como el tamaño, la ubicación o la apariencia.

- **displayingTrackingResults**

  Esta función dibuja un bounding box y un ID para cada *track* de cada fotograma del video y en la máscara de primer plano. Posteriormente muestra el marco y la máscara en sus respectivos reproductores de video.

4.3.1.2. Aplicación del algoritmo de seguimiento

El algoritmo de detección de objetos de Matlab comienza creando un sistema de objetos en el que una de sus funciones es sacar una pantalla donde se reproducen dos videos, el original y uno en el que se realiza la resta de imágenes para detectar que pixeles son parte del fondo y cuales son parte primer plano tal y como se muestra en la figura 4.15.

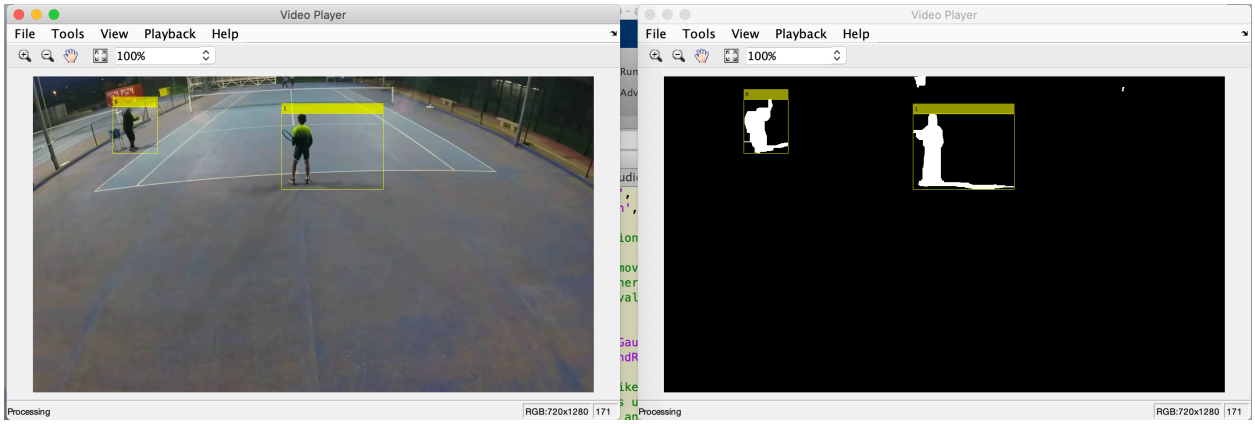

Figura 4.15. Reproductor de vídeo

Para realizar la resta para la diferencia de fondo y primer plano se han ajustado los parámetros.

Además, también se ha ajustado el parámetro de “MinimumBlobArea” para determinar a partir de qué número de pixeles conectados se debe considerar un objeto. Tal y como se ve en la imagen 4.16 lo que correspondería a otra sombra del jugador (abajo a la izquierda) no se ha considerado como objeto.

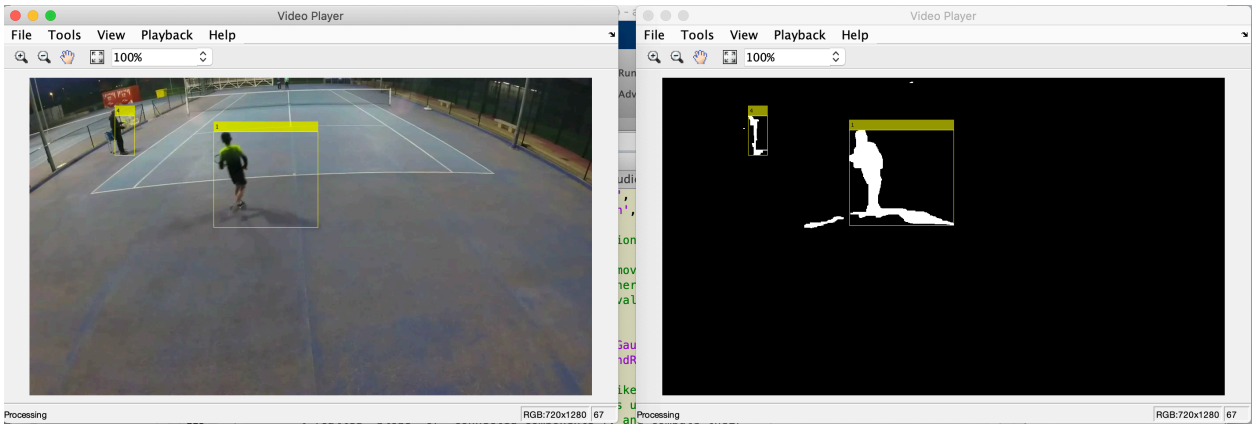

Figura 4.16. Reproductor vídeo con grupo de pixeles en primer plano no considerados

Tras la creación del sistema de objetos se inicializa una matriz vacía que servirá para la lectura del video, la detección de los objetos en movimiento y mostrar los resultados. Dentro de la matriz se almacenarán los siguientes parámetros: id, bounding box, KalmanFilter, age, totalVisibleCount y consecutiveInvisibleCount. A partir de estos parámetros por ejemplo se puede descartar un *track* porque ha sido detectado durante un período de tiempo demasiado corto o porque tras haberse detectado han se ha excedido un número de frames consecutivos sin que aparezca. El no ajustar estos parámetros de manera correcta puede llevar a resultados como el que se muestra en la figura 4.17 en donde se han detectado demasiados tracks.

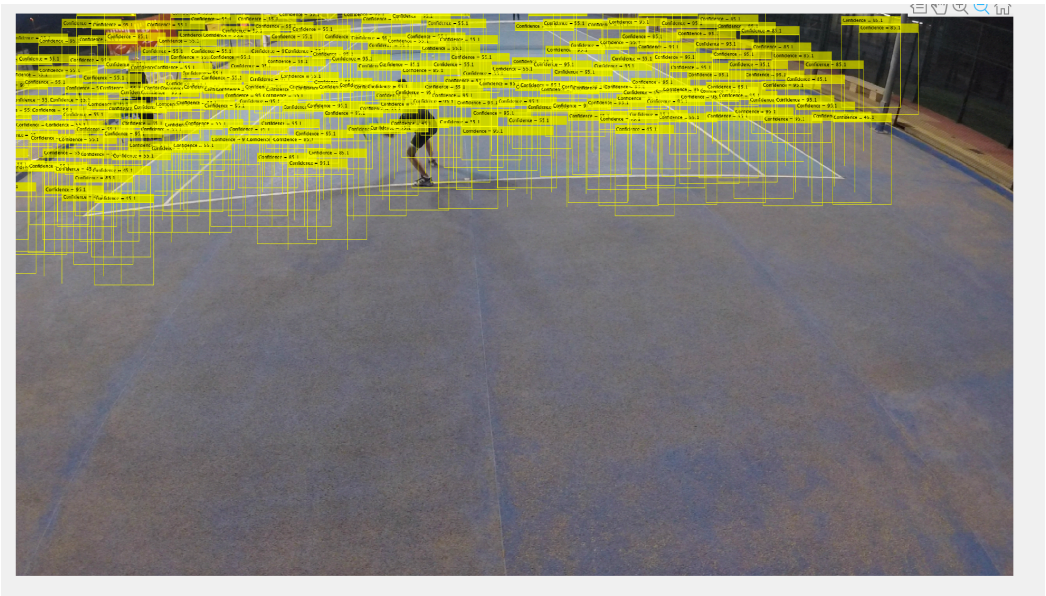

Figura 4.17. Imagen con demasiados tracks detectados.

A continuación, se inicializa una variable contador, tras la cual se inicializan otras ocho variables que servirán para contabilizar golpes y desplazamientos.

Por último, se crea un bucle principal desde el que realiza llamadas a una serie de funciones que son las que proporcionarán el resultado de detección final.

DetectOjects es la primera función que se ejecuta al entrar en este bucle principal del algoritmo. Esta función devuelve los centroides y los bounding box de los objetos detectados. También devuelve la máscara binaria, que tiene el mismo tamaño que el marco de entrada. Los píxeles con un valor de 1 corresponden al primer plano y los píxeles con un valor de 0 corresponden al fondo.

Se realiza la segmentación del jugador utilizando el detector de primer plano. Posteriormente, se realizarán operaciones morfológicas en la máscara binaria resultante para eliminar píxeles ruidosos y rellenar los agujeros en las manchas restantes.

Como ya se ha comentado el primer paso de *detectObjects* es detectar el plano principal (o lo que no es fondo). Se puede ver un ejemplo en la figura 4.18 la cual contiene el mismo frame antes (imagen 1) y después (imagen 2) de aplicar el detector de plano principal.

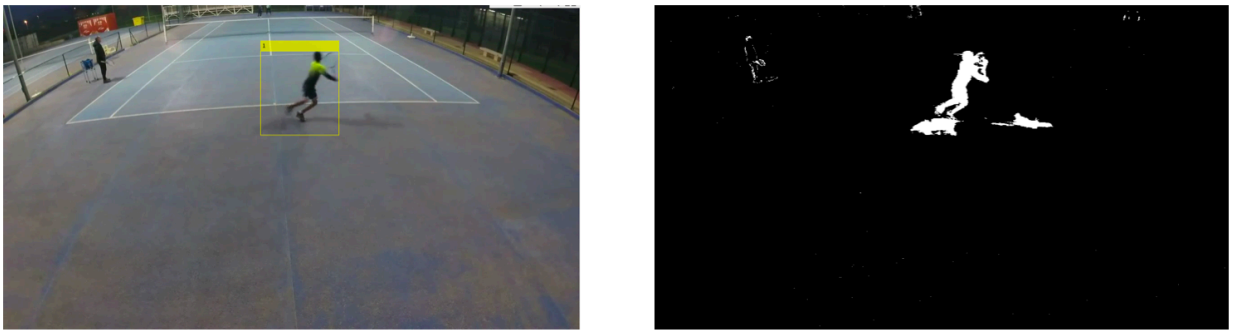

Figura 4.18. Izquierda: Frame inicial. Derecha: Frame tras aplicar el detector de plano principal

En la imagen 1 de la figura está la imagen original en la que se aprecia el *bounding box* (recuadro amarillo) el cual marca y delimita el jugador que más tarde será objeto de análisis. Dentro de esta misma imagen podemos ver una segunda persona en la parte izquierda de la pista de tenis (tal y como se describió en la figura 4.11) y unas sombras en los pies del jugador.

La imagen 2 es una representación lógica de la imagen 1 indicando con *unos* (blanco) los pixeles que se consideran plano principal y con ceros (negro) los pixeles que representan el fondo.Se puede ver que tanto el jugador como las sombras y el entrenador han sido detectados como plano principal. Durante el desarrollo de este algoritmo veremos cómo descartar estos pixeles que han sido considerados en el plano principal para analizar únicamente la figura del jugador.

Es aquí donde finaliza la primera etapa que corresponde al algoritmo de seguimiento.

4.3.2. Detección del jugador

Una vez finalizada la etapa anterior ya se tiene una estimación del fondo y se ha detectado los objetos en movimiento dentro de cada frame. Si se observa el diagrama de la figura 4.13 vemos que se acaba de completar la segunda caja del proceso que corresponde con "aplicación de operadores morfológicos".

A continuación empezaría la primera parte de la segunda etapa que corresponde con la detección del jugador

Tras la detección del plano se aplican los operadores morfológicos open, close y fill (descritos en el apartado 3.4 de este documento) en este orden para eliminar ruido en la imagen y rellenar posibles huecos. Podemos ver su efecto en la figura 4.19 y 4.20.

Figura 4.19. Izquierda: Frame tras aplicar el detector de plano principal. Derecha: Frame tras aplicar open.

Figura 4.20. Izquierda: Frame tras aplicar close. Derecha: Frame tras aplicar fill.

En estas dos figuras podemos ver como la figura del entrenador va desapareciendo y los pixeles arriba a la derecha que son ruido van disminuyendo.

Posteriormente aplicamos un filtro que en el que se eliminarán todos los grupos de pixeles menores a un determinado tamaño (en este caso 8000). Para ello lo que hará Matlab es poner

los a 0 (negro) y de esta manera dejaremos las figuras más grandes. Podemos ver un ejemplo en la figura 4.21.

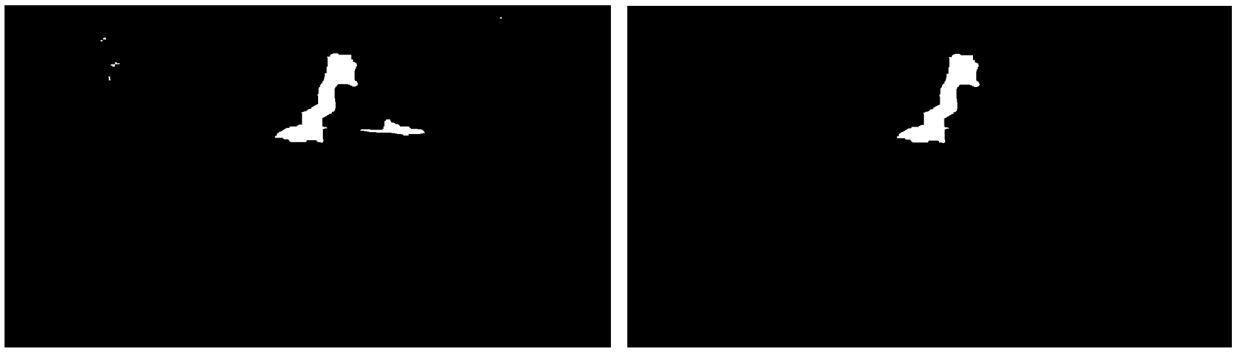

Figura 4.21. Izquierda: Frame tras aplicar fill. Derecha: Frame tras filtrar los grupos de pixeles menores de 8000.

Como consecuencia del filtro anterior algunas imágenes han quedado completamente en negro, por lo que lo siguiente que se ha hecho es poner una condición para que todos estos frames se descarten.

A continuación, se ha obtenido el tamaño y la posición del bounding box que envuelve a la figura del jugador en la última imagen de la figura 4.20 (imagen 2) para recortar con estas medidas y posición en la imagen original figura 4.18 (imagen 1).

Ahora, a esta imagen original figura 4.18 (imagen 1) ya recortada se le aplica el algoritmo k-means, explicado en el apartado 3.3 de este documento, con tres clusters que se espera que sean la pista por la zona de juego, la zona que rodea la pista de juego y el jugador. Se puede ver el resultado en la figura 4.22.

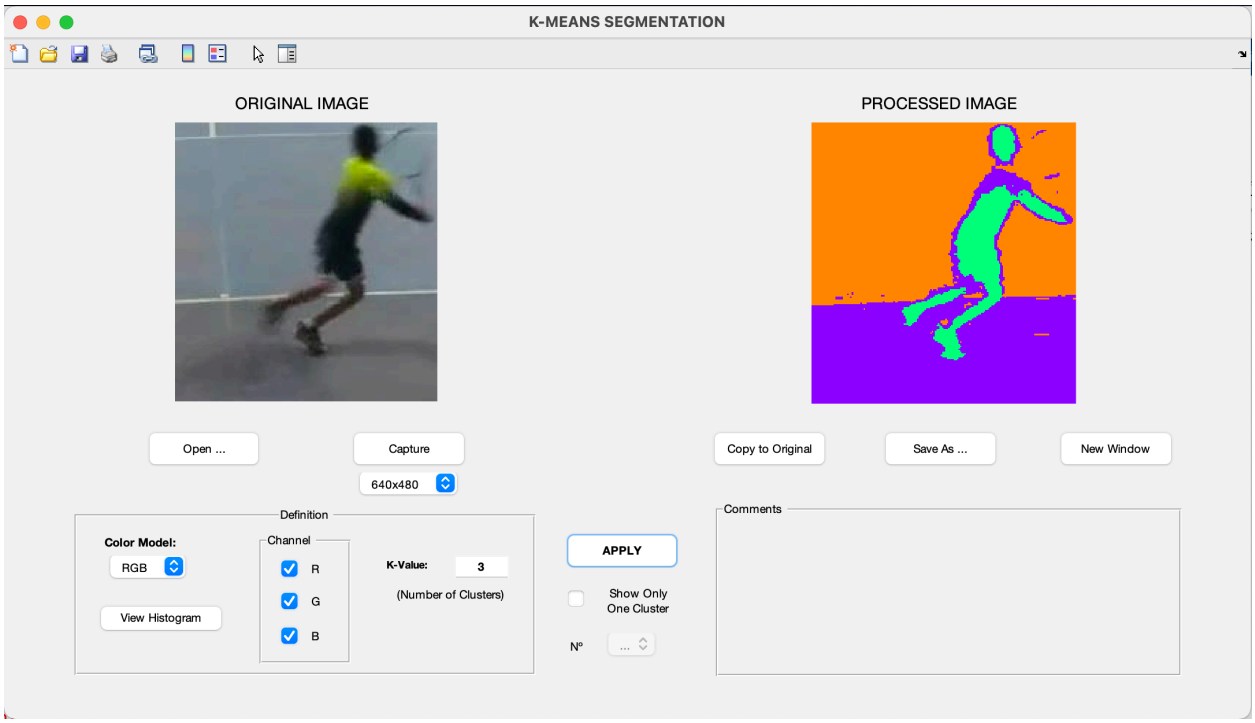

Figura 4.22. Aplicar algoritmo K-Means con tres clusters sobre la imagen original recortada.

Con el resultado de las tres zonas que devuelve el algoritmo, figura 4.22, se coge la menor de las tres, ya que se espera que las zonas de pista siempre sean mayores, se ha puesto a “1” lógico (blanco) y todo lo demás a “0” lógico (negro).

El objetivo de este proceso es obtener recortada la imagen del jugador y que además este más definida de como la encontrábamos en el último paso figura 4.21 (imagen 2). Se puede ver el resultado de este proceso en la figura 4.23.

Figura 4.23. Izquierda: Frame tras filtrar los grupos de pixeles menores de 8000. Derecha: Frame tras recortar el bounding box sobre la imagen original y aplicar el algoritmo K-Means

Una vez tenemos esta última imagen se ha vuelto a aplicar un filtro que en el que se eliminarán todos los grupos de pixeles menores a un determinado tamaño (en este caso 1800). Para ello lo que hará Matlab es poner los a 0 (negro) y de esta manera dejaremos las figuras más grandes. También se ha vuelto a aplicar el operador morfológico close para intentar conectar grupos de pixeles que deberían pertenecer al mismo pero que por poco se han quedado en distintos. Se puede ver el resultado en la figura 4.24.

Figura 4.24. Izquierda: Frame Tras aplicar algoritmo K-Means. Centro: Frame tras aplicar filtro areas de 1800 Pixels. Derecha: Frame tras aplicar el operador close

Destaca en este frame elegido para el ejemplo que a pesar de lo aplicado la segunda pierna del jugador ha quedado descartada. El mismo método no puede ser igual de efectivo para los más de 600 frames que componen el video que se está analizando en este caso. Más adelante se demostrará que a pesar de esto el resultado sigue siendo el esperado.

En el siguiente paso se ha obtenido la solidez de los distintos objetos que se pueden encontrar dentro de la imagen, entendiendo por solidez la densidad de un objeto siendo sólido un objeto más denso y menos sólido menos denso. Después de haber obtenido la solidez de todos los objetos que pudieran aparecer en la imagen, se ha filtrado para quedarse únicamente con el area menos sólida, ya que se considera que la figura del jugador tiene una solidez baja. En la figura 4.25 para no perder la evolución que veníamos viendo hasta ahora tenemos el frame con el que hemos realizado los ejemplos hasta el momento que en este caso tampoco se ve afectado. En la figura 4.26 se puede ver otro ejemplo en el que este filtro ha tenido mas efecto.

Figura 4.25. Izquierda:Frame tras aplicar filtro areas de 1800 Pixels. Derecha: Frame tras filtrar por solidez

Figura 4.26. Izquierda: Frame 2 tras aplicar filtro areas de 1800 Pixels. Derecha: Frame 2 tras filtrar por solidez

Como consecuencia de los últimos pasos algunos frames pueden haber quedado en negro totalmente, por lo que se ha vuelto a filtrar y se han descartados todas aquellas imágenes que no tuvieras alguna figura.

En este punto se ha revisado la relación axial correspondiente a cada frame y se han descartados aquellos que tuvieran una relación ancho partido de largo del bounding box obtenido, que sería la inversa de la relación axial ya que estamos haciendo eje menor partido de eje mayor, menor a 0.6. Con ello se pretende descartar aquellos frames que tuviera una un grupo de pixeles a “1” lógico (blancos) y no fueran la figura del jugador. Podemos ver un ejemplo de frame descartado en la figura 4.27.

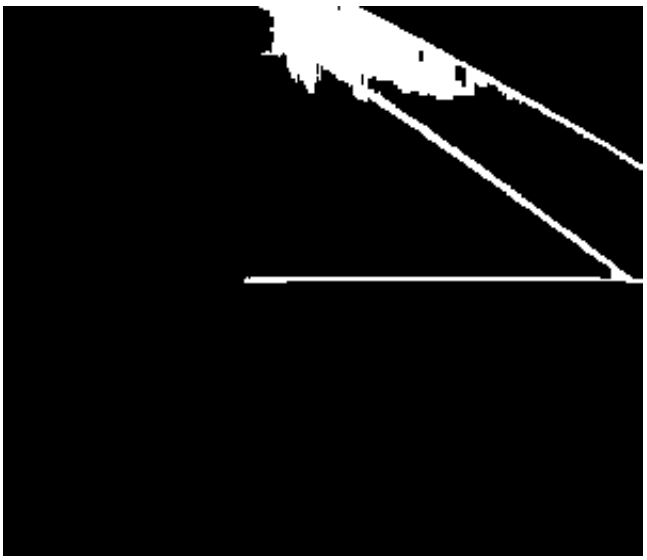

Figura 4.27. Frame descartado por relación ancho largo

Finalmente, se ha recortado levemente por la zona inferior del eje vertical los frames resultantes ya que la parte del cuerpo que determina el tipo de golpe son los brazos del jugador y es sobre esta figura donde realizaremos el cálculo final. Se puede ver un ejemplo en la figura 4.28. Además, se ha filtrado para que se quedara el objeto más grande (poniendo lo demás a cero lógico) y se ha vuelto a filtrar para eliminar grupos de pixeles con un área menor a 500.

Figura 4.28. Izquierda: Frame antes de recortar. Derecha: Frame tras el recorte

4.3.3. Clasificación de golpes

Por último, está la segunda parte de la segunda etapa que corresponde a la clasificación de golpes, último paso del diagrama de la figura 4.13. Para considerar si un frame pertenece a un golpe de derecha o de revés se ha definido un método de análisis adhoc a partir de los descriptores de la figura del jugador obtenida del proceso anterior.

Primero de todo se ha extraído el centro de masas del área y se ha guardado su valor del eje x, y el centro del bounding box guardando también su valor del eje x. A continuación, se ha hecho la resta del segundo sobre el primero y al resultado de esta operación de ahora en adelante lo llamaremos *desplazamiento*. *Desplazamiento* representa la diferencia que hay sobre el eje x entre el centro de masas del área y el centro del bounding box.

Cuando el resultado de *desplazamiento* sea positivo consideraremos que el jugador tiene los brazos extendidos hacia la derecha por lo que estará yendo a por un golpe de derecha. Cuando el resultado de *desplazamiento* sea negativo consideraremos que el jugador tiene los brazos extendidos hacia la izquierda por lo que estará yendo a por un golpe de revés.

Se puede ver un ejemplo en la figura 4.29, en donde en la imagen 1 se observan los brazos del jugador extendidos hacia la derecha lo que resulta en un desplazamiento positivo ya que la diferencia entre el centro de masas y el centro de bounding box será positiva y por lo tanto estará golpeando de derecha. Sin embargo, en la imagen 2 se observan los brazos del jugador extendidos hacia la izquierda lo que resulta en un desplazamiento negativo ya que la diferencia entre el centro de masas y el centro del bounding box será negativa y por lo tanto estará golpeando de revés.

Figura 4.29. Izquierda: Jugador con variable *desplazamiento* positiva. Derecha: Jugador con variable *desplazamiento* negativa.

Se ha establecido un umbral entre -5 y 5 en la variable desplazamiento para descartar los frames en los que la figura del jugador apenas se ha movido. Se considera que cualquier valor para la variable desplazamiento encontrado dentro de este umbral no corresponde a un golpe ya que la figura del jugador no se ha movido lo necesario para realizar un golpe.

Hemos dicho que si el jugador tiene los brazos extendidos hacia la derecha es porque busca realizar un golpe de este lado, en el momento que realiza el golpe busca recuperar el centro de la pista otra vez por lo que en este caso moverá los brazos hacia el otro lado. Es por esto que finalmente para considerar que el jugador ha realizado un golpe se ha tenido en cuenta el cambio de dirección en el movimiento del cuerpo, que se verá reflejado en el cambio de sentido en el valor de la variable *desplazamiento.* Tras detectar un golpe se ha mantenido un margen de guarda temporal para evitar detectar un *falso golpe*.

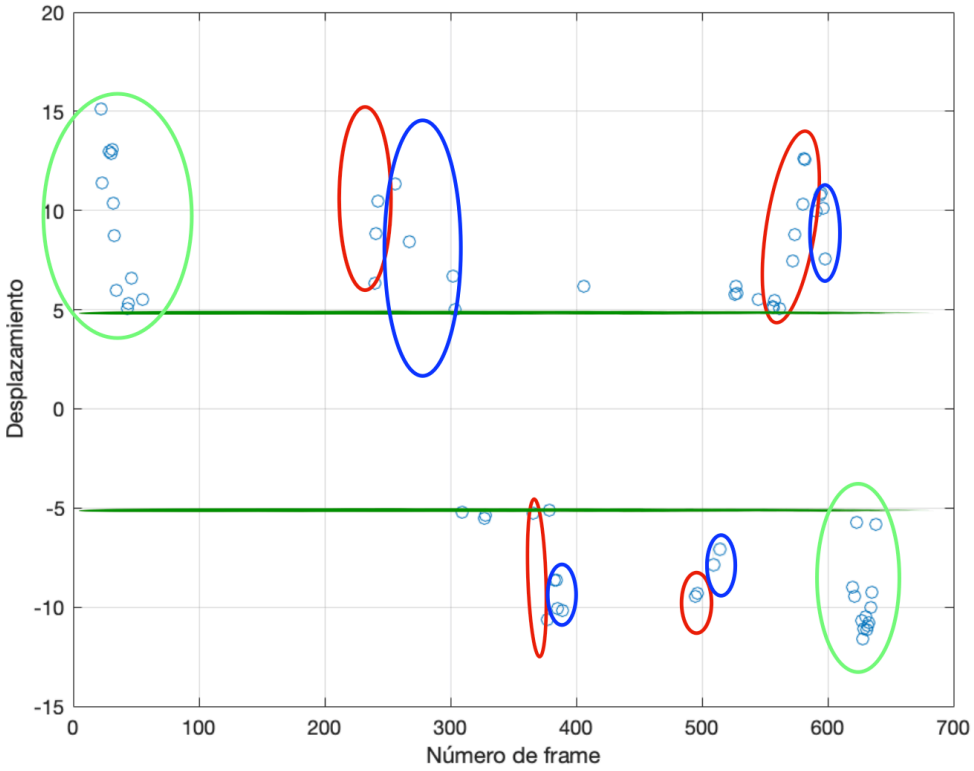

Figura 4.30. Resultado de la variable desplazamiento

En la figura 4.30 distinguimos en verde claro las zonas de comienzo y finalización del juego. Dado que se comienza grabando con la pista vacía y posteriormente entra el jugador en escena estos desplazamientos marcan la entrada y la salida. Las líneas verde oscuro marcan el umbral a partir del cual se ha considerado que se estaba desplazando el jugador para golpear una derecha (5), y el umbral a partir del cual se considera que el jugador se estaba desplazando para golpear un revés (-5). El inicio de un desplazamiento para golpear está marcado en color rojo mientras que el retroceso al centro de la pista tras haber golpeado está marcado en azul. Se ha dicho que se considera un golpe cuando hay un cambio en el sentido de la variable desplazamiento, por lo que un golpe sobre la gráfica sería la combinación de una zona roja con la azul.

5. Métodos alternativos

En este apartado se explican métodos alternativos que se probaron para algunas partes del proyecto pero que finalmente se descartaron.

5.1. Cálculo del número golpes por sonido

Para identificar los golpes se intentó combinar el análisis de imagen con el sonido. Se considera que cuando el jugador realiza un golpe, al encontrarse cerca de la cámara, se registra un sonido corto y alto. En la primera gráfica de la figura 5.1 se encuentran rodeados en azul. Finalmente se desechó esta opción por encontrarse muchos más sonidos como la voz del entrenador o golpes de la pista de al lado.

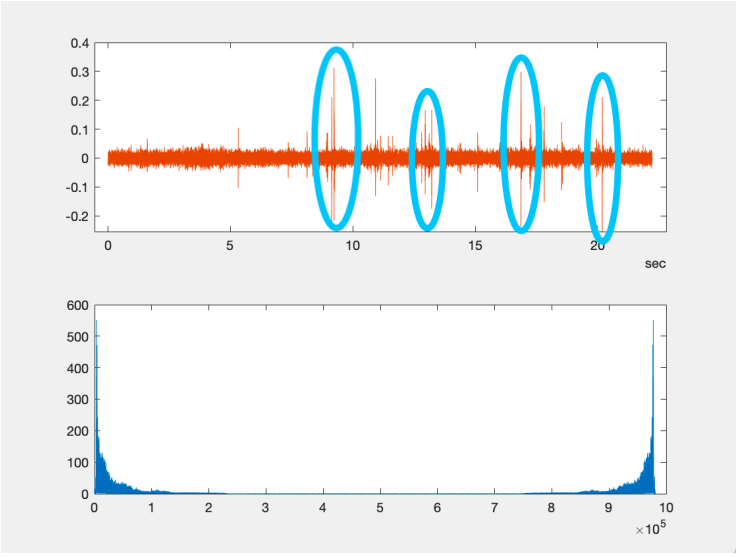

Figura 5.1. Gráfica de sonido del video

5.2. CNN

Tras la etapa de detección del tenista (objeto en movimiento) se proponen las redes neuronales convolucionales como método alternativo al algoritmo K-Means y las distintas operaciones aplicadas sobre los frames para la identificación del golpe realizado. Podemos apreciar la diferencia entre métodos comparando la figura 5.2 que muestra el flujograma para este método alternativo y el de la figura 4.14 que muestra el del algoritmo desarrollado.

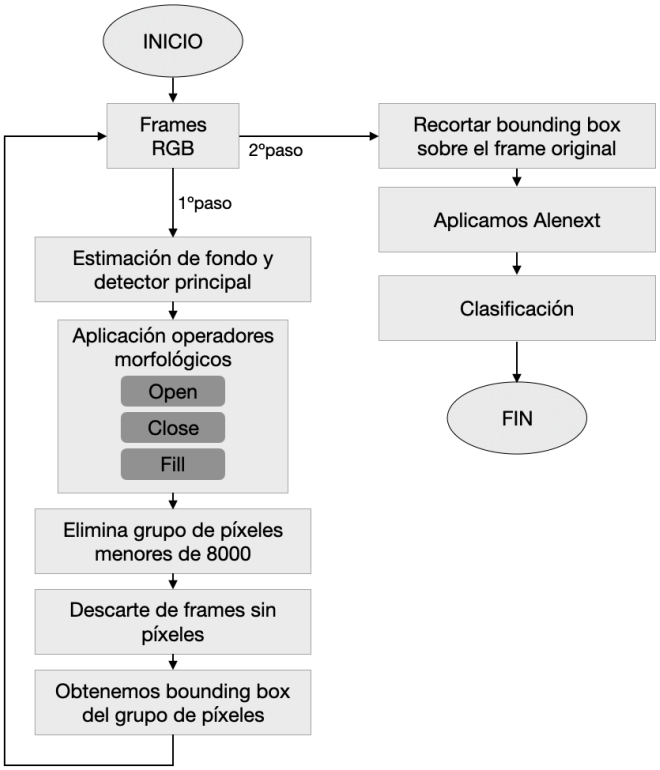

Figura 5.2. Flujograma red neuronal convolucional Alexnet

La red neuronal convolucional que se ha utilizado es la de Alexnet. Alexnet está compuesta por ocho capas, cinco capas convolucionales en la parte de extracción de características del modelo y tres capas interconectadas en la parte de clasificación además de utilizar un

algoritmo de activación ReLu no saturante que muestra un rendimiento mejorado sobre la formación tanh y sigmoide. Las imágenes que se analizan en esta red deben tener un tamaño de 224x224x3.

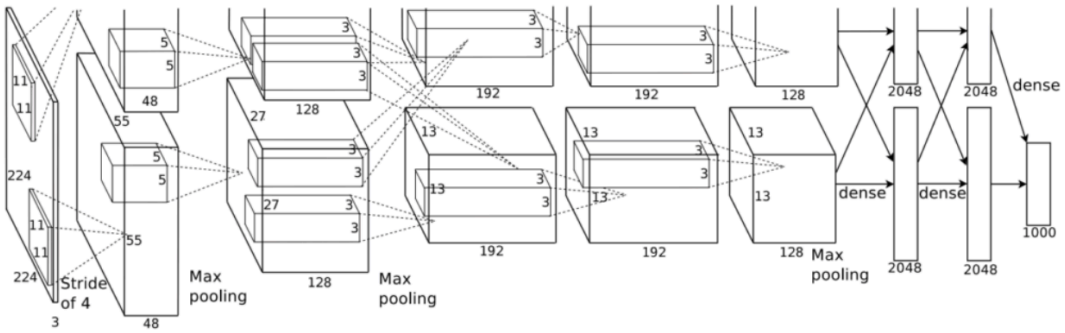

Figura 5.3. Arquitectura red neuronal convolucional Alexnet

Para utilizar Alexnet se ha realizado *transfer learning,* se han utilizado los frames identificados en las primeras etapas del algoritmo de detección de movimiento para entrenarla. Una vez ya entrenada se ha analizado el mismo video que en el algoritmo desarrollado para ver y ésta ha devuelto los frames ya clasificados. Por último, se han comparado los resultados.

Se ha elegido Alexnet por no ser una de las redes neuronales convolucionales más complejas, esperando así que el coste computacional no fuera muy alto. Sin embargo, el análisis con Alexnet de un video de 20 segundos en un ordenador portátil de prestaciones medias cuesta alrededor de 2 minutos mientras que con el algoritmo desarrollado en el proyecto cuesta 1 minuto, reduciendo así el tiempo a la mitad. En el caso de Alexnet analizaría unos 4 frames por segundo mientras que en el algoritmo desarrollado son 9 frames por segundo. Hay que destacar que con un ordenador con una capacidad de procesamiento mayor el tiempo hubiera sido mucho menor, pero en este caso se ha comparado el coste de ambos métodos sobre el mismo ordenador.

Tras el análisis con la red neuronal convolucional Alexnet se han obtenido los resultados que se han representado en la siguiente gráficas de la figura 5.4.

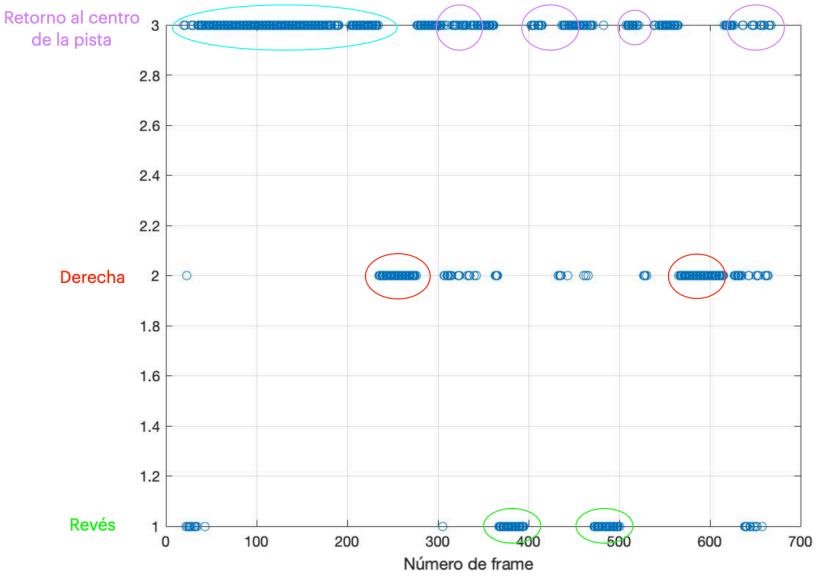

Figura 5.4. Resultados red neuronal convolucional Alexnet

En los resultados de la gráfica observamos tres líneas horizontales a lo largo del eje vertical. La que se sitúa en 1 corresponde a los frames que tras el análisis se han identificado como un golpe de revés, la que se sitúa en 2 corresponde a los frames que tras el análisis se han identificado como golpes de derecha y por último la que se sitúa en 3 corresponde a los frames en los que el jugador estaba retornando de un golpe al centro de la pista. En el eje horizontal tenemos los números de frame. Los primeros frames de la gráfica (marcados en azul clarito) corresponderían al jugador entrando a la pista y situándose en el centro de ella, a continuación realizará un golpe de derecha (marcado en rojo un poco después del frame 200) y retornará al centro de la pista (marcado en morado). El siguiente golpe será un revés (marcado en verde tras el frame 350) y retornará al centro de la pista (marcado morado). A continuación realizará otro golpe de revés, retornará al centro de la pista y realizará un golpe más de derecha. Si se compara con los resultados de la figura 4.30 vemos que son bastante similares

Además de devolver los frames ya clasificados, Alexnet te proporciona el grado de verosimilitud con que lo ha hecho, se pueden ver los resultados en la gráfica de la derecha de la figura 5.5.

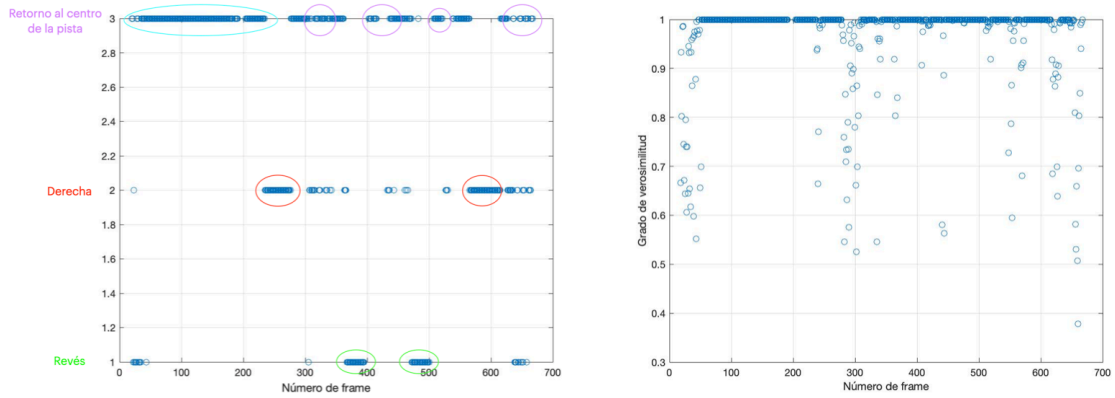

Figura 5.5. Grado de verosimilitud respecto a los resultados obtenidos en la red neuronal convolucional Alexnet

Si se compara ambas gráficas de la figura 5.5 se puede distinguir que el grado de confianza con que la red analiza los frames en los que el jugador está realizando un golpe es muy alta, mientras que en los frames en los que el jugador se está desplazando por la pista se reduce.

Finalmente, este método resultó interesante durante el desarrollo del proyecto, pero se descartó debido al alto coste computacional.

6. Interfaz gráfica

Para facilitar al usuario el manejo del código implementado se ha creado una interfaz gráfica compuesta por tres ventanas.

Al inicio del programa aparecerá la primera ventana, el panel de control, situada en la parte izquierda de la pantalla. En la figura 6.1 se muestra el aspecto.

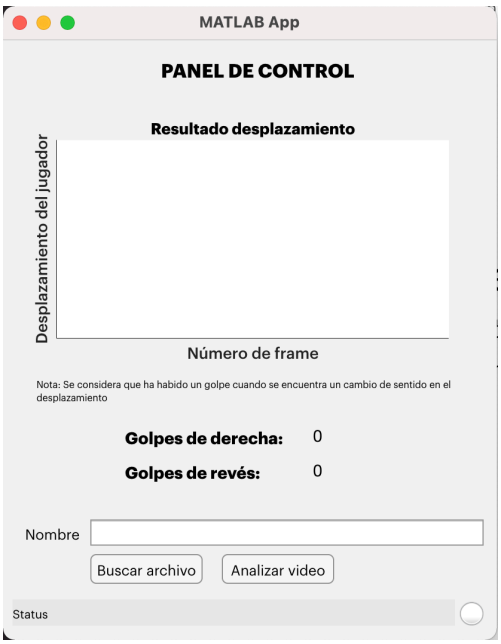

Figura 6.1. Panel de control

Tras iniciar el análisis del video aparecerán dos ventanas más, dos reproductores de video, a la derecha del panel de control tal y como se muestra en la figura 6.2. El reproductor de arriba proyectará el video original recuadrando en amarillo los objetos que identifica como jugadores inicialmente. El reproductor de abajo proyectará la máscara del video original (dejando en negro el fondo y en blanco los objetos en movimiento) recuadrando en amarillo los objetos que identifica como jugadores inicialmente. Tras el análisis completo puede que algunos objetos recuadrados en amarillo en estos reproductores o que estén en blanco en el reproductor de la máscara, ya que se han identificado como objetos en movimiento, hayan quedado descartados por no haber cumplido las condiciones del proceso. Estos reproductores solo pretenden ayudar a la visualización del proceso que se está llevando a cabo, pero no representan el resultado final.

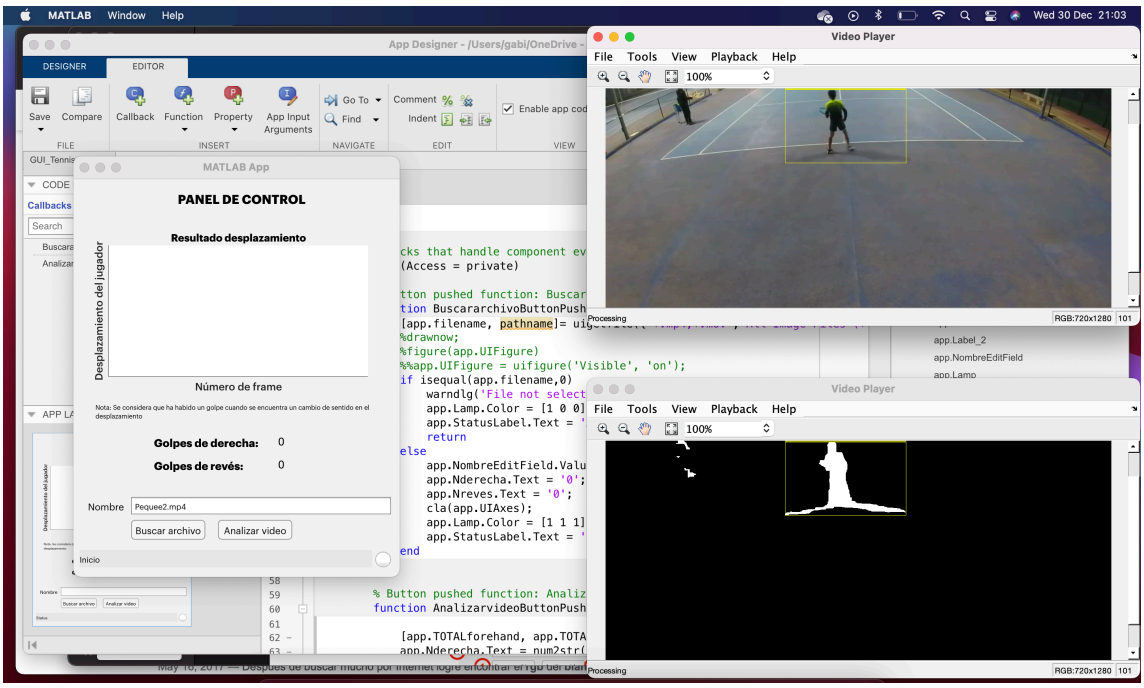

Figura 6.2. Interfaz completa

6.1. Funcionamiento panel de control

Para explicar el manejo del panel de control se han numerado los distintos elementos que lo componen, tal y como se puede apreciar en la figura 6.3, y definido.

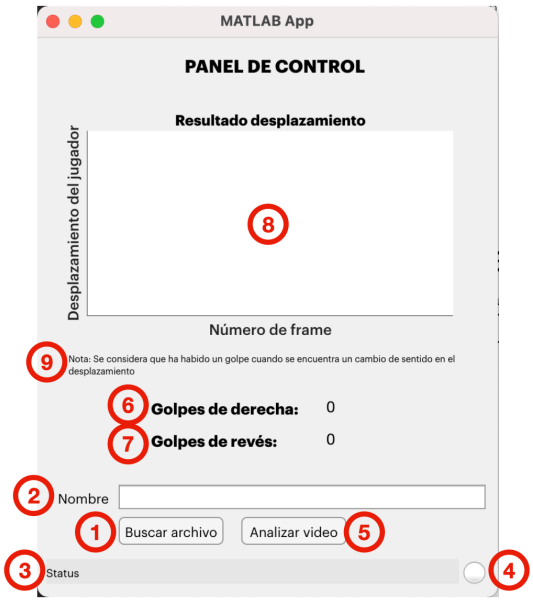

Figura 6.3. Elementos del panel de control

Definición de los elementos:

1. **Botón de buscar archivo**: Botón que permite buscar el vídeo objeto de análisis por los documentos del ordenador. En caso de no seleccionar ningún archivo aparecerá un mensaje de error tal y como se muestra en la figura 6.4.

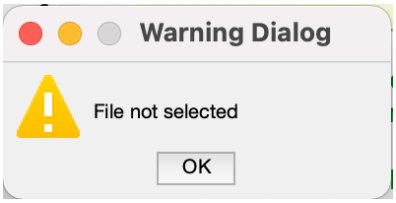

Figura 6.4 Mensaje de error por archivo no seleccionado

2. **Etiqueta Nombre**: En este recuadro blanco se puede visualizar el nombre del archivo que se ha seleccionado.
3. **Status**: En este recuadro gris se indicará el estado en el que se encuentra el proceso que básicamente son 3.
   - Inicio: indica que todavía no se ha iniciado el análisis del video.
   - Análisis completo: indica que el análisis del video se ha completado.
   - Error en el análisis: indica que el proceso de análisis no se ha completado por ejemplo cuando no se ha seleccionado un video.
4. **Círculo de color**: Ayuda en la visualización del estado del proceso. Hay tres colores.
   - Blanco para el estado de “inicio” en el que se indica que todavía no se ha iniciado el análisis del video.

- Verde para el estado de “análisis completo” en el que se indica que el análisis del video se ha completado.
- Rojo para el estado de “error en el análisis” en el que se indica que el proceso de análisis no se ha completado por ejemplo cuando no se ha seleccionado un video.

5. **Botón de analizar video**: Una vez seleccionado el video objeto de estudio se utilizará este botón para comenzar el análisis.
6. **Etiqueta golpes de derecha**: A la derecha de esta etiqueta aparece el número total de golpes de derecha identificados en el video.
7. **Etiqueta golpes de revés**: A la derecha de esta etiqueta aparece el número total de golpes de revés identificados en el video.
8. **Gráfica resultado de la variable desplazamiento**: La variable desplazamiento almacena el valor obtenido de la diferencia entre el centro del área de la figura del jugador y el centro del eje x. Esta variable se pinta en esta gráfica para poder ver un resumen del proceso que se ha analizado. En el eje x se representa el número de frame mientras que el eje y el valor del desplazamiento. Será positivo cuando el jugador se desplace hacia la derecha y negativo cuando se desplace hacia el revés.
9. **Nota**: Indica un mensaje para aclarar que “se considera que ha habido un golpe cuando se encuentra un cambio de sentido en el desplazamiento”.

7. Resultados

En este punto se mostrarán los resultados obtenidos tras aplicar el algoritmo sobre un video, las dificultades encontradas durante el desarrollo del método de análisis del video y las posibles mejoras que se le podrían introducir.

7.1.Resultados obtenidos

Una vez ejecutado el algoritmo sobre el vídeo seleccionado se verían reflejados los resultados sobre el panel de control tal y como se muestra en la figura 7.1. Se puede apreciar la gráfica resultante de la variable de desplazamiento y debajo de esta el número de golpes de derecha y de revés identificados. Por último, abajo a la derecha tenemos el indicador del círculo de color en este caso verde, lo que indica que el análisis se ha finalizado con éxito.

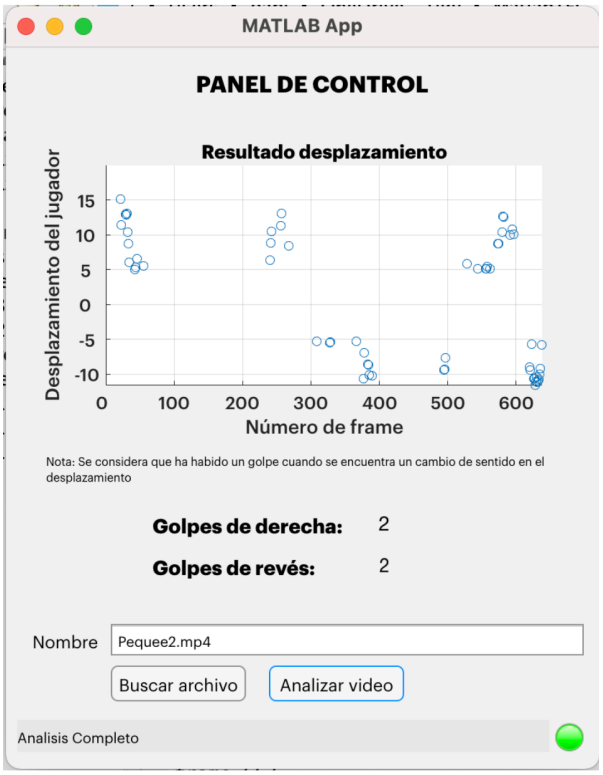

Figura 7.1. Panel de control tras la ejecución.

7.2.Dificultades encontradas

La principal y mayor dificultad encontrada durante el desarrollo del proyecto fue la eliminación de las sombras que provocaban los distintos focos de iluminación de la pista junto con la figura del jugador. Durante el análisis del video se encuentran hasta tres sombras del mismo jugador que tanto al realizar el seguimiento del objeto en movimiento como al aplicar la máscara no desaparecen.

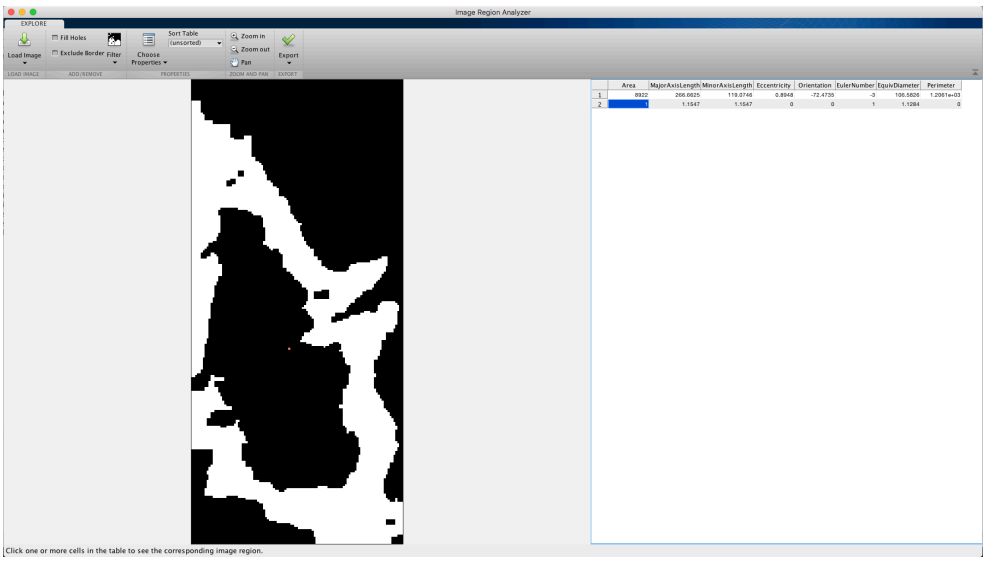

Figura 7.2. Máscara de la imagen del jugador en la que no se ha separa el jugador de su sombra.

En muchas ocasiones el color de la ropa del jugador y la pista no eran lo suficientemente diferentes, y junto con la iluminación artificial ha llevado a que no se identificara por completo la silueta del jugador, sino que se recortara por algún lado como en el caso de la figura 7.2.

7.3. Posibles mejoras

Algunas mejoras que se podrían desarrollar para el proyecto serían las siguientes:

- Análisis en tiempo real. Obtener resultados en el mismo momento o con segundos de retraso para que el entrenador pueda ir sacando estadísticas a medida que pasan los puntos.
- Identificar más tipos de golpes. Actualmente solo se identifican los golpes de derecha o de revés, pero se podría seguir desarrollando el algoritmo para diferenciar entre derecha, revés, boleas, golpes cortos y saque.
- Identificar los golpes quitados. Se dice que un jugador golpea de derecha quitada cuando la bola viene a la zona del revés, pero el jugador se desplaza de más hacia la izquierda para colocarse de manera que la pueda golpear de derecha. Lo mismo pero a la inversa para el revés quitado. En este caso el programa no contaría una derecha quitada sino un revés.
- Diferenciar entre golpe bueno y fallado. En el caso de que el jugador realice el golpe pero la pelota se quede en la red (falle) diferenciarlo del golpe que si pasa por encima de la red.
- Detección y seguimiento de la pelota. En el caso de que el jugador realice el golpe diferenciar de si el golpe es bueno o fallado debido a que la pelota bote dentro o fuera de la zona de juego.

8. Conclusión

Podemos concluir que la detección y el reconocimiento de jugadores de tenis mediante el procesado de imagen es una tarea muy compleja. Esta tarea depende de una gran cantidad de factores externos difíciles de controlar, como la iluminación, el fondo (ya que podría haber más personas en la pista), el color de la pista, el color de la ropa del jugador, la velocidad a la que se mueve la raqueta del jugador, entre otras.

En este trabajo se ha conseguido realizar un código que sea capaz de detectar al jugador e identificar el tipo (derecha o revés) y número de golpes de forma efectiva cuando la iluminación no es la más favorable. En un caso en el que los colores de la ropa del jugador y de la pista hubieran sido muy contrastados y la iluminación fuera natural de un día soleado el análisis hubiera sido mucho más fácil. Se ha querido escoger estas condiciones porque se consideran las más habituales en la práctica del tenis a diario para una persona no profesional. También se es consciente de que este algoritmo se puede mejorar en muchos aspectos; por ejemplo, permitir el análisis en tiempo real, conseguir identificar más tipos de golpes, lograr que cuando el jugador falle un golpe no se cuente como bola golpeada, sino que se identifique como fallo de derecha o revés, o que si el jugador apenas se ha desplazado para el golpe, el algoritmo sea capaz de identificarlo.

La realización de este trabajo me ha permitido afianzar mis conocimientos sobre procesamiento de imagen y adquirir fluidez programando con el Software MATLAB y sus aplicaciones como App Designer o las contenidas dentro del módulo de procesamiento de imagen o visión por computador. Este proyecto me ha hecho darme cuenta de todos los inconvenientes o detalles inesperados que pueden surgir al llevar la teoría a la práctica y de la gran necesidad de “prueba y error” para poder solventarlos.

Este trabajo surgió como inspiración de una de mis pasiones, el tenis, el interés por la imagen y video y el estudio de una asignatura sobre el tratamiento de imagen y video. Se podrían introducir bastantes mejoras ya que es un campo de análisis muy amplio y complejo, aun así,

se considera que el resultado final obtenido es un buen resultado ya que se ha cumplido el objetivo inicial de identificar y diferenciar los golpes de derecha y revés.

Por último, añadir que recientemente un grupo de desarrolladores del autopiloto de Tesla junto con el equipo de IOS de Apple ha desarrollado una aplicación llamada “Swing Vision” solo disponible para iPhone que se asemeja a la idea de la que surgió este proyecto [2].

9. Bibliografía

[1] Naveen Joshi. *The Present And Future Of Computer Vision (2019)*. URL: https://www.forbes.com/sites/cognitiveworld/2019/06/26/the-present-and-future-of-computer-vision/?sh=146d321a517d

[2] Mangolytics, Inc (2020). *Swing vision.* URL: https://swing.tennis

[3] The MathWorks, Inc(1994-2021). Kalman Filter. URL: https://www.mathworks.com/discovery/kalman-filter.html

[4] Oscar Contreras Carrasco. *Gaussian Mixture Models Explained.* Towards data Science. URL: https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

[5] José Martínez Heras. *La Maldición de la Dimensión en Machine Learning*. IArtificial.net. URL: https://www.iartificial.net/clustering-agrupamiento-kmeans-ejemplos-en-python/

[6] Apuntes de la Asignatura “Tratamiento digital de imagen y video”. *Capítulo 2.1 Segmentación de imágenes.*

[7] Apuntes de la Asignatura “Tratamiento digital de imagen y video”. *Capítulo 2.3 Morfología.*

[8] ATRIA Innovation. *Qué son las redes neuronales y sus funciones* (Octubre 2019). URL: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

[9] Juan Ignacio Bagnato. *¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador* (Noviembre 2018). URL: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[10] The MathWorks, Inc (1994-2021). *Motion-Based Multiple Object Tracking*. URL: https://es.mathworks.com/help/vision/ug/motion-based-multiple-object-tracking.html;jsessionid=746aa7d553890452ae0508197022

[11] James Munkres. Algorithms for the assignment and transportation problems. Jo Soc. IDUST. APPL. HATtt. Vol. 5, No. 1, March, 1957 Printed in U.S.A. URL: https://web.eecs.umich.edu/~pettie/matching/Munkres-variant-of-Hungarian-alg.pdf

[12] *Innovaciones arquitectónicas en redes neuronales.* URL: https://sitiobigdata.com/2019/05/01/innovaciones-arquitectonicas-redes-neuronales-clasificacion-imagenes/#

[13] Andrew P. Bradley, Terence Bloom, *Player Tracking and Stroke Recognition in Tennis Video.*

[14] Damien Connaghan, Ciarán ÓConaire, Philip Kelly, Noel E. O'Connor, *Recognition of Tennis Strokes using Key Postures,* ISSC 2010, UCC, Cork, June 23–24.

[15] Jinzi Mao, *Tracking a tennis ball using image processing techniques,* (Agosto 2006). A thesis submitted to the College of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Computer Science in the Deparment of Computer Science University of Saskatchewan Saskatoon.