Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Entrenador-Ayudante Digital en Tenis

Trabajo Fin de Máster

Presentado por: Molina Márquez, Gonzalo

Directora: Mancera Valetts, Laura

Ciudad: Madrid

Fecha: 19-09-2019

Resumen

¿Cómo crear un sistema de anotación automática en tenis a partir de videos de baja calidad? Existen herramientas como el “Ojo de Halcón” que dictaminan si una bola ha entrado o no, decidiendo por encima de los jueces, pero requiere de una gran infraestructura. Este Trabajo de Fin de Máster ha creado un piloto que, a partir de vídeos grabados a pie de pista, consigue información sobre los metros recorridos por el jugador cercano a la cámara, las líneas de la pista, la posición y trayectoria de la pelota. Además, gracias a los datos anteriores se dictamina en que frames se produce un golpe, y de qué tipo es, con un acierto del 70%. Los resultados obtenidos demuestran que se pueden crear aplicaciones a partir de imágenes de una cámara de gama media, y montar un sistema de ayuda a jugadores amateurs y escuelas de tenis.

Palabras Clave: Anotación Automática, Tenis, Visión Artificial, Machine Learning.

Abstract

How could we build an automatic annotation system for tennis using low quality videos? Some tools as Hawk-Eye that rules if a ball is in or out, beyond the judges, but it required a big infraestructure. This TFM has code a proof of concept, base on videos filmed behind the player. The code gets total amount meters covered by the player, court´s lines, ball position and her trajectory. Besides, It is possible to get what frames contains strokes, even detecting the type of stroke, with a 70% of accuracy. It is demostrated that apps can be build to solve annotation problems from a low reflex cam, and this system helps to amateur players and tenis academies.

Keywords: Automatic Annotation Tennis Artificial Vision

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN 6

1.1 VISTA GENERAL 6
1.2 PLANTEAMIENTO DEL PROBLEMA 6
1.3 JUSTIFICACIÓN 11
1.4 OBJETIVOS 12
1.4.1 Objetivo general 12
1.4.2 Objetivos específicos 13
1.5 ALCANCE Y LIMITACIONES 13
1.6 CAPITULACIÓN 15

2 CONTEXTO Y ESTADO DEL ARTE 16

2.1 VISTA GENERAL 16
2.2 EL TENIS 16
2.3 INTELIGENCIA ARTIFICIAL 17
2.3.1 Aprendizaje Automático (Machine Learning) 17
2.3.2 Redes Neuronales y Deep Learning 18
2.3.3 Visón Artificial 21
2.4 CAMPOS DE APLICACIÓN DE LA VISIÓN ARTIFICIAL 23
2.5 ESTADO DEL ARTE 26
2.5.1 Investigaciones Punteras Ámbito Visión Artificial 26
2.5.2 Investigaciones Aplicadas al Tenis 28
2.5.2.1 Detección de Movimientos 28
2.5.2.2 Detección de líneas 30
2.5.2.3 Clasificación Golpes de Tenis 30
2.5.2.4 Estadísticas en Tenis 31

3 METODOLOGÍA 32

3.1 ENFOQUE METODOLÓGICO 32
3.2 DISEÑO INVESTIGACIÓN 32
3.3 RECOLECCIÓN DE DATOS: TÉCNICAS Y FUENTES 33

4 DESARROLLO TÉCNICO DEL TFM 35

4.1 INTRODUCCIÓN 35
4.2 DESARROLLO PILOTO 35
4.2.1 Detección del Jugador 36
4.2.1.1 Sprint 1: Detección de Movimientos 36
4.2.1.2 Sprint 2: Selección del contorno del jugador 40
4.2.1.3 Sprint 3: Obtención la posición del jugador e información 41
4.2.2 Objetivo 2: Referenciar medidas de la pista de tenis 42
4.2.2.1 Sprint 1: Combinación de Canny, segmentación color y Transformada de Hough 43
4.2.2.2 Sprint 2: Locura de filtros de color y combinación de máscaras 47
4.2.2.3 Sprint 3: K-Means 48
4.2.2.4 Sprint 4: Modelado de la pista 50
4.2.3 Objetivo 3: Seguir la pelota 52
4.2.3.1 Sprint1: Continuidad a la detección del jugador 53
4.2.3.2 Sprint 2: Eliminación de puntos no deseados 55
4.2.3.3 Spring 3: Creación de trayectorias posibles 58
4.2.4 Objetivo 4: Clasificar el tipo de golpe 61
4.2.4.1 Sprint 1: Primer prototipo de red neuronal para la clasificación de golpes 62
4.2.4.2 Sprint 2: Más datos, profundidad de la red y modelos preentrenados 64

4.2.4.3 Sprint 3: Vuelta a métodos de visión artificial 66
4.3 RESULTADOS 68
4.3.1 Objetivo 1: Detección del Jugador 68
4.3.2 Objetivo 2: Referenciar medidas de la pista de tenis: 70
4.3.3 Objetivo 3: Seguir la pelota 71
4.3.4 Objetivo 4: Clasificar el tipo de golpe 73
4.4 PROBLEMAS ENCONTRADOS 73
4.4.1 Objetivo 1: Detección del Jugador 74
4.4.2 Objetivo 2: Referenciar medidas de la pista de tenis: 74
4.4.3 Objetivo 3: Seguir la pelota 75
4.4.4 Objetivo 4: Clasificar el tipo de golpe 75
5 CONCLUSIONES Y TRABAJO FUTURO 77
5.1 CONCLUSIONES DEL TRABAJO 77
5.2 POSIBLES MEJORAS FUTURAS 78
6 REFERENCIAS 79
7 ANEXOS 83

ÍNDICE DE IMÁGENES

Imagen 1. Medidas Pista de Tenis 17
Imagen 2. Ejemplo Red Neuronal.....19
Imagen 3. Ejemplo Red Convolucional.....20
Imagen 4. Ejemplo Detección Objetos.....26
Imagen 5. Detección Jugador Actual y Delta.....38
Imagen 6. Detección Jugador Blur Delta y Actual.....38
Imagen 7. Detección Jugador Diferencia Imágenes, Threshold y Dilatación Threshold.....38
Imagen 8. Detección Jugador Resultado Sprint 1.....40
Imagen 9. Detección Jugador Resultado Sprint 2.....41
Imagen 10. Detección Jugador Resultado Final 42
Imagen 11. Detección Líneas, Imagen Real y Máscara HSV 44
Imagen 12. Detección Líneas, Izq. diferencia dilatada y grises; Dcha. dilatado anterior 44
Imagen 13. Detección Líneas, Izq. cruce máscaras color y dilatada; Dcha. Canny anterior .45
Imagen 14. Detección Líneas, Resultado Prueba 1 Sprint1.....45
Imagen 15: Detección Líneas, Resultado Prueba 2 Sprint1.....46
Imagen 16. Detección Líneas, Resultado Sprint2.....47
Imagen 17. Detección Líneas, Original truncada zona de interés y Canny anterior 49
Imagen 18. Detección Líneas, Izq. k-mean original truncada, Dcha. Canny And k-mean 49
Imagen 19: Detección Líneas, Resultado Sprint 3.....50
Imagen 20. Detección Líneas, Resultado Final 52
Imagen 21. Detección Pelota, Izq. Original; Dcha. Máscara Color.....54
Imagen 22. Detección Pelota, Izq. diferencia con delta; Dcha.: binarización anterior 54
Imagen 23. Detección Pelota, Dilatación Binarización; Dcha. Color And Binarización.....55
Imagen 24. Detección Pelota, Resultado Sprint 1 55
Imagen 25: Este primer caso es un falso positivo (detecta la raqueta). 56
Imagen 26: Detección Pelota, A la derecha se ve lo que detecta la pelota correctamente. ..57
Imagen 27: Detección Pelota, Se detecta pelota y falso positivo.....57
Imagen 28: Detección Pelota, ejemplo luz de día detección correcta.....57
Imagen 29: Detección Pelota, Izq. Original; Dcha. Inicio trayectoria pelota 60
Imagen 30: Detección Pelota, Izq. Original; Dcha. continuación trayectoria pelota.....60
Imagen 31: Detección Pelota, Izq. Original; Dcha. continuación trayectoria pelota 2.....60
Imagen 32: Detección Pelota, Izq. Original; Dcha. continuación trayectoria pelota 3.....61
Imagen 33: Dos ejemplos de derechas detectadas 68
Imagen 34: Dos ejemplos de reveses detectados, a la derecha erróneamente 68
Imagen 35: Suma Metros Jugador: inicio secuencia.....69
Imagen 36: Suma Metros Jugador: secuencia intermedia 69
Imagen 37: Suma Metros Jugador: secuencia final 70
Imagen 38: Detección Líneas, Resultado final noche 70

ÍNDICE DE TABLAS

Tabla 1. Frames Visibles en Vídeo....71
Tabla 2: Clasificación frames detección pelota....72
Tabla 3: Trayectorias detectadas en video....72
Tabla 4: Golpes detectados correctamente....73
Tabla 5: Tipos de golpes detectados correctamente....73
Tabla 6: Aciertos por tipo de golpe concreto....73

CAPÍTULO 1. INTRODUCCIÓN

1.1 Vista general

Los avances actuales en el campo de la Inteligencia Artificial (IA) han permitido realizar aplicaciones que hace solo 5 años eran una quimera. Avances como las redes convolucionales para el reconocimiento de objetos es uno los más recientes, marcando en 2012 AlexNet el hito de superar a los métodos tradicionales de reconocimiento (Tsang, 2018). El maridaje entre los avances en estas técnicas de procesamiento de imagen y la necesidad de datos estadísticos en el deporte, en concreto el tenis, son las bases sobre las que parte esta investigación. Este apartado presenta la problemática que da lugar a este trabajo, junto con la justificación de llevarlo a cabo, así como los objetivos concretos que se trazaron para realizarlo.

También se definen una serie de limitaciones, en parte impuestas por la propia naturaleza de un Trabajo de Fin de Máster (TFM), que debe tener un alcance limitado, aunque se proponen técnicas y métodos que podrían suplirlas en posteriores mejoras de este trabajo.

El ámbito de actuación de este TFM es hacia los niveles de tenis medios o bajos, puesto que los centros de alto rendimiento y equipos profesionales cuentan con más personal por cada jugador, permitiendo esos análisis más minuciosos. Sin embargo, se pretende que, con unos medios básicos, como puede ser un móvil o una cámara digital casera, se obtenga información valiosa que de otra forma sería muy complicada conseguir, almacenar y consultar.

1.2 Planteamiento del Problema

Los últimos 20 años han visto una explosión de las tecnologías, sobre todo Internet, acompañados de la movilidad ofrecida por las tecnologías inalámbricas, provocando a su vez, una nueva ola de cambios. Han aparecido redes sociales que llegan a número de usuarios más grandes que la población de cualquier país (“Top 20 Facebook Statistics - Updated July 2019,” n.d.). También, se empieza a ver que muchos trabajos serán desempeñados por ordenadores o robots en un futuro cercano. Una gran protagonista de esta segunda ola de revolución tecnológica que se está viviendo, es sin duda la Inteligencia Artificial, IA en adelante. Hace unos pocos años la IA se ocupaba de problemas como resolver eficientemente

el algoritmo de Dijkstra con métodos como A* y derivados. Esto, actualmente, es algo muy básico comparado con las tareas que la IA está siendo capaz de resolver. Aquí cabe mencionar que, parte de estos avances, además de los trabajos teóricos realizados años atrás, se debe a los avances de la computación en la nube o cloud, las cuales han dotado de una gran capacidad de computo que permite resolver problemas que, antes a nivel hardware era imposible (quizás solo en los grandes mainframes).

Entre los avances en el campo de la IA se encuentran el Procesamiento del Lenguaje Natural (PLN), el cual ha pasado en poco tiempo de un sistema que procesaba las palabras del presentador para buscar una respuesta (concurso Jeopardy del 2011 en el que IBM venció con Watson) al boom actual de los asistentes en casa, en los que hablando a un altavoz puedes ordenar tareas básicas como apagar luces, poner el aire acondicionado o incluso realizar compras. Hoy en día, los programas reconocedores de voz funcionan con una gran precisión, y son capaces de "entender" lo que quiere el usuario para realizar la acción que corresponda con una tasa de error muy baja, del 6,8% de acuerdo a (K. Johnson, 2019). Un punto más avanzado en este contexto, son los llamados bots, que son capaces de tener conversaciones con humanos, los cuales, de momento, requieren ser programados para cada contexto como reservar una mesa en un restaurante, aunque cada vez van siendo más flexibles. A la final estos avances del PLN están permitiendo cambiar la forma en la que se relacionan con muchos dispositivos eléctricos y electrónicos, desde una lavadora a maquinaría médica, y la tendencia es que cada vez sea más fácil y común interaccionar por comandos de voz con los dispositivos, y no a través de botones o complicados menús.

Otro avance importante es el tratamiento de datos masivos. La disponibilidad de cantidades ingentes de datos, que se generan a grandes velocidades y de diferentes fuentes, posibilitan aplicaciones en una inabarcable amplitud de campos, siendo probablemente los más explotados los de sector marketing, financieros, médicos y redes sociales. La conjunción de metodologías de aprendizaje automático con los sistemas que permiten manejar, incluso, en tiempo real toda esta información, es lo que ha permitido estos avances espectaculares y que sea una de las áreas de "moda" a nivel empresarial.

También es relevante hacer una mención especial a las técnicas de Deep Learning (DL), de las que no pasa una semana sin que se produzcan noticias sobre ella. La venta del primer cuadro creado por una IA basado en redes neuronales, programas que son capaces de derrotar a humanos en juegos tan diversos como el GO o StarCraft. DL es ahora la punta de lanza de los últimos logros, con redes cada vez más complejas.

Y si de avances se trata, es importante mencionar la evolución de las redes neuronales a las redes convolucionales, que también han permitido avanzar espectacularmente en el

reconocimiento de objetos, dentro de un entorno controlado de un número de objetos limitado, de hecho, los algoritmos de reconocimiento superan a los humanos en estas tareas. Las primeras aplicaciones leían números para envíos de postales o tratamiento digital de formularios, y supusieron una innovación en su momento, aunque comparado con avances como los coches que conducen solos o robots que pueden trabajar en un almacén de logística moviendo y clasificando paquetes, se queda en sistemas simples. El abanico de entornos donde aplicar estas técnicas es, de nuevo, extenso. Algunos ejemplo en seguridad son: el seguimiento de matrículas (Aprimatic, 2019) e identificación de personas de manera automática (Jacobs, 2018) o realizar búsquedas de personas concretas o incluso identificar si alguien está robando en unos grandes almacenes (Ayn de, 2019). En marketing existen analizadores de la edad, sexo o incluso estado anímico de los posibles compradores permite analizar a los clientes y poder hacer acciones para mejorar las ventas. Sanidad como que redes neuronales son capaces de reconocer dolencias a partir de radiografías o escáneres (Johnson, 2019), y han logrado en algunos casos incluso superar a los expertos, y aunque por ahora, no parece que vayan a sustituir a los doctores, serán una herramienta de apoyo que mejore la detección de enfermedades. Más avanzado incluso, un sistema que graba una intervención quirúrgica y estima la sangre que ha perdido el paciente durante la intervención, para tener un dato preciso sobre cuanta sangre es necesaria inyectar al paciente para equilibrar la pérdida (Ayn de, 2019). También, Drones y distintos sensores permiten en agricultura seguir el estado de las cosechas de un modo hasta ahora imposible de realizar en grandes terrenos (lAyn de, 2019). Pueden vigilar diferentes parámetros ayudando, por ejemplo, a optimizar el uso de fertilizantes y obteniendo mejores cosechas, lo que se traduce en una bajada de los costes del cultivo. Por supuesto en la industria hay múltiples usos, como clasificadores de basura, complejos robots multitarea que identifica visualmente las piezas que deben montar, está empezando a crear fábricas con menos personal, lo que ha permitido en casos puntuales rebajar los costes en la fabricación en un país como EEUU, por debajo de países como poca protección laboral, que tradicionalmente eran muy bajos por los bajos salarios (Pastor, 2017).

Dentro de los campos de la IA, es la visión artificial donde se enfoca la investigación presentada en este documento, dado que las diferentes técnicas de análisis de imágenes están los suficientemente maduras. Específicamente, se lleva al terreno del deporte, y es que el uso de estas técnicas permite mejorar, estudiar y analizar el comportamiento de los deportistas con el apoyo de la visión artificial.

Vistos algunos de los avances en IA, se introduce el planteamiento de la parte tenística que también abarcará este trabajo. Cualquier deporte requiere habilidades o facultades distintas, las cuales se podrían agrupar en cuatro categorías muy generales:

1) Estado físico: referente a la resistencia, velocidad, fuerza, coordinación, equilibrio, agilidad y flexibilidad.

2)Técnica: que se refiere a la ejecución correcta de los movimientos, que permita aprovechar al máximo el esfuerzo.

3) Táctica: El “game plan” que utilice el deportista puede ser determinante puesto que, ante contrincantes superiores, buscar una táctica adecuada puede dar la vuelta al resultado.

4) Psicológica: que se refieren a las capacidades mentales que distinguen a los buenos jugadores de los mejores, tales como la actitud positiva y ganadora, saber reponerse a situaciones adversas, empatía que permita entender cómo se encuentra el rival, mantener la calma en momentos importantes, entre otros.

Adicionalmente, es importante mencionar que a su vez cada una de las categorías está compuesta de otros múltiples elementos, lo que implica mayor procesado de información, como por ejemplo dentro del estado físico se tiene en cuenta la nutrición. Estas cuatro categorías son aplicables al tenis en mayor o menor medida

En el tenis, los niños empiezan sus primeras competiciones en las categorías alevines e infantiles, durante las cuales habitualmente un entrenador no puede realizar un seguimiento exhaustivo debido a que suele enseñar a muchos alumnos distintos. Esto provoca que las apreciaciones del entrenador puedan estar sesgadas a las pocas jugadas que haya visto de un jugador, o puede que ni siquiera haya visto el partido y solo tenga referencia de lo que le comente el jugador o los padres del niño, lo que no suelen ser opiniones muy parciales ni técnicas.

Una forma de registrar lo que ha ocurrido en un partido es anotando cada jugada, idealmente cada golpe, para posteriormente poder analizar el número de golpes ganados o perdidos, en función de si es una derecha, revés o volea. El trabajo de (Zbeide Rivera, 2018) muestra una forma de anotar el desarrollo del partido bastante detallada. Esto es solo un ejemplo, pero cada entrenador puede querer fijarse en unos datos distintos, por lo que es difícil decidir en que fijarse, llegando a ser imposible apuntar todos esos datos “en vivo”, obligando a grabar el encuentro y un lento proceso de revisión del partido. Esta labor resulta muy laboriosa y en muchas ocasiones imposible de realizar, por lo que alternativas automáticas que realicen estas tareas serían muy útiles para analizar los partidos, pudiendo ofrecer datos objetivos para posteriores análisis y así mejorar a partir de los datos. Además, los sistemas automáticos pueden recopilar muchas más variables poniéndolas disponibles a jugadores y entrenadores.

Existen diferentes tipos de sensores como detectores tipo radar para detectar la posición de la bola (Foxtenn, 2019; Hulfish, 2017), software comercial que facilita analizar partidos al ofrecer una interface para que una persona vaya recogiendo las estadísticas (“LongoMatch | La herramienta de videoanálisis para entrenadores, analistas deportivos y jugadores,” 2019; “Nacsport | Software de Vídeo Análisis para todos los Deportes,” 2019) o incluso dispositivos que se colocan en la muñeca o raqueta (Johnson, 2016). Una combinación de varios de estos elementos o software comercial con un coste elevado puede proporcionar al menos parte de esta información, pero actualmente no existe una alternativa para la mayoría de los jugadores de nivel amateur o escuelas de tenis fuera de los centros de alto rendimiento (Martínez-Gallego, 2015). Actualmente una empresa ofrece este servicio, pero no parece, al menos no lo comentan, que sea con software automático (ITusa, 2019).

Los datos recogidos y estudiados por un entrenador, o incluso otro sistema automático, permiten estudiar qué ha pasado en el partido: las estadísticas de cuantos golpes de derecha o revés ha realizado, desde que posición de la pista golpea, subidas a la red, velocidad y altura de la bola, dirección de los golpes, el total de metros recorridos en el partido, golpes ganados y fallados, etc. Existen una multitud de datos técnicos a recoger, pero además se podría extraer información sobre la actitud, si andaba con la cabeza agachada entre punto y punto, si corre menos y baja su combatividad, sus expresiones corporales o faciales, e incluso si habla enfadado a sí mismo o golpea la raqueta. Idealmente, esa información podría enlazarse con el resultado del partido en cada momento, pudiendo ofrecer razones por la que, en cierto momento del partido, el jugador ha realizado algún cambio para bien o para mal que ha decidido el partido. Además, es una forma de recoger datos objetivamente, porque un jugador, especialmente si ha perdido, puede no estar de acuerdo con lo que le diga su entrenador, pero si muestra los datos recogidos automáticamente tendrá que aceptarlos. Si cada jugador llevara un registro de sus partidos y/o entrenamientos, al cabo del tiempo podrá comparar y ver la evolución en su juego. Incluso si se enfrenta a un mismo jugador, podría estudiar los datos de los anteriores partidos para entrar a la pista con una idea clara de lo que debe y no debe hacer.

Con base en la problemática planteada y los beneficios que aporta la IA para resolverla, este TFM plantea la siguiente pregunta de investigación: ¿Cómo crear un sistema de anotación automática en tenis a partir de videos captados con dispositivos de gama media?

1.3 Justificación

Este TFM pretende abordar el problema planteado en el apartado 1.2 sobre la anotación de datos técnicos del juego de un tenista, ya sea en entrenamiento o en partidos. Se ha explicado la importancia de contar con un sistema que pueda recopilar información valiosa, tanto para el jugador como para el entrenador, de manera automática. Los centros de alto rendimiento o los equipos profesionales cuentan con muchos recursos, pero en un nivel de iniciación o escuelas más modestas es casi imposible hacer un seguimiento personalizado a cada jugador dado que los costes que acarrean estos estudios son elevados y muchas veces no pueden ser financiados por las familias.

A continuación, se exponen distintos puntos sobre los que este TFM pretender ser una ayuda y solventar al menos en parte, alguna de las carencias comentadas:

- Los deportes cuando se pasa del nivel de entretenimiento a un nivel amateur o competición suelen conllevar una subida exponencial de los gastos. Se debe practicar más horas, más horas de entrenador, más desgaste de material, más viajes, lo que directamente supone costes mayores, en los que sobre todo en los deportes individuales los debe sufragar la familia. Esta razón es una de las que impide a muchos jugadores poder dedicarse más en serio al deporte, y concretamente en el caso del tenis son más altos que en otros deportes. El material es caro y los grupos de entrenamiento reducidos, es típico un profesor para cuatro alumnos, lo que implica que todo se debe pagar entre pocos alumnos. A estos niveles suele ser inviable, económicamente, que el jugador tenga un entrenador dedicado para cada jugador, siendo los padres los únicos que acompañan a los chicos a los torneos. La familia suele tener toda la voluntad del mundo, pero no los conocimientos para realizar un diagnóstico de lo que ha pasado en el partido. Un sistema de análisis automático y económico que pudieran utilizar en estos niveles, permitiría subir la calidad en estas escuelas y mejorar el desempeño de estos jugadores.
- Existen múltiples trabajos sobre tratamiento de vídeos de tenis, en aspectos sobre predicción de que golpe ha ejecutado un jugador, localización del jugador y la pelota en la imagen, si el punto ha sido “Highlight” según los aplausos del público, e incluso el “Ojo de Halcón” se ha ganado por méritos propios ser el mecanismo que resuelve si una bola ha entrado o no dentro de la pista. El primer punto que suele ser común en la mayoría de estos trabajos, es que siempre emplean imágenes de grabaciones de televisiones con una gran calidad. Estos vídeos están grabados por cámaras profesionales, colocadas en el mejor ángulo posible, enfoques y contrastes realizados por profesionales. El tratamiento de estas imágenes con las técnicas actuales está

relativamente resuelto y se obtienen porcentajes altísimos de aciertos, pero llevados al entorno más de escuela se complica, habitualmente nunca se dispondrá de una cámara aérea (el uso de un dron en sitios públicos está regulado al menos en España, además del inconveniente del ruido y necesidad de un operador). Tampoco se tendrá disponible una cámara de alta calidad ni por supuesto una configuración de la cámara profesional. Estos impedimentos hacen que previsiblemente los porcentajes de aciertos de estos trabajos bajen notablemente. Este TFM busca aplicar en casos reales las distintas técnicas realizadas en otros estudios y probar su efectividad, modificándolos o adaptándolos cuando sean necesarios para un mejor desempeño. Se espera que el porcentaje de aciertos obviamente baje, pero que aun así permita obtener datos con valor y pueda utilizarse.

- Premisa de este TFM es que no se emplearán métodos que requieran medios de gama alta, como cámaras superlentas que capten muchas frames por segundo que captan la bola perfectamente y no salga movida en la imagen. Se busca su mayor aplicabilidad al día a día, siendo accesible para cualquier practicante que quiera utilizarlo. Anteriormente se comentaba que hay sistemas actuales que usan diferentes sensores para obtener información con alto detalle, que combinándolos adecuadamente se podría conseguir una buena recopilación de datos, pero en este trabajo se realizará siempre desde la premisa de descartarlos, intentando conseguir los mejores resultados solo con vídeo, estando al alcance de más practicantes.
- Deportes como el running o ciclismo, cuentan con multitud de aplicaciones para el móvil con las que, cualquier persona que lo practique, puede almacenar su rendimiento, compartirlo con amigo y compararlo para revisar su evolución en el tiempo. Si bien, en el tenis esto requiere mucho más que guardar una posición GPS, este trabajo pretende poder servir de ejemplo para la realización de aplicaciones similares, pero en el mundo del tenis.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un piloto que utilice diferentes técnicas de tratamiento de imágenes y vídeos de IA, para obtener la mayor información posible de vídeos de tenis, utilizando cámaras de gama media desde enfoques a pie de pista, para determinar el impacto que tiene un sistema automático como apoya a los entrenadores de tenis.

1.4.2 Objetivos específicos

Los datos que persigue este TFM son cuatro valores distintos, que combinándolos entre ellos, y según la precisión obtenida, pueden dar una información adicional:

1. **Posicionar al jugador principal:** Se posiciona solo al jugador más cercano a la cámara. Se irá haciendo un seguimiento de los movimientos del jugador por la pista, identificando en qué lugar de la pista se encuentra cruzando con la información obtenida en el punto 2. Esta información servirá para conocer por donde se mueve y golpea el jugador, además de cuanto ha corrido el jugador.
2. **Referenciar medidas de la pista:** Obtener dónde están las diferentes líneas de la pista, como línea de fondo, laterales, saque y la red central. Las imágenes se tratan para identificar elementos rectos y de color blanco. La situación de los diferentes elementos de la pista, se utilizarán para situar a los jugadores y la pelota con respecto a la misma.
3. **Seguir la pelota:** Realizar el seguimiento del movimiento de la pelota durante el juego. Además de las técnicas utilizadas en el punto 1, habrá que combinarla con otras como segmentación por color. La información obtenida será la trayectoria de la pelota, e incluso velocidad y altura, que permita conocer las direcciones de los golpes del jugador y del rival, e información útil sobre el golpeo si va con más o menos fuerza.
4. **Clasificar el tipo de golpe:** Identificar cuando un jugador realiza un golpe de derecha, de revés. También intentar, además de la información básica, anotar si el golpe lleva un tipo de efecto liftado o cortado. La identificación del tipo de golpe arroja información, combinándola con los puntos anteriores, para conocer que táctica del jugador.

1.5 Alcance y limitaciones

El objetivo es ambicioso puesto que el tratamiento de imágenes de baja calidad es complejo, y se pretende obtener información diversa, que requerirá diferentes técnicas para conseguirlo. Esta razón hace que se limite el alcance y se establezcan ciertas condiciones para facilitar la consecución de los objetivos.

Con respecto al alcance:

- Partiendo de vídeos de partidos o entrenamientos de tenis en modalidad individual, se realizarán diferentes procesados de las imágenes para obtener información relevante, útil para mejorar en la práctica del deporte.
- Los procesos persiguen información sobre dónde está la pista, el jugador, la pelota y el tipo de golpe que realiza.

- Las imágenes con las que se trabajan no son en condiciones ideales. Hay diferentes situaciones reales, como sombras en las pistas, poca luz solar, o bolas viejas que dificulta distinguirlas por color del fondo o de la pista. Además, al estar la cámara detrás del jugador, en muchas ocasiones se pierde literalmente de vista la pelota. Estas dificultades provocan un nivel de acierto inferior, pero que el muestreo de cada frame combina información con el resto, para obtener informes usables y satisfactorios.
- Realización de informe básico con los datos recogidos en los que se muestre la utilidad del piloto.
- El trabajo tiene afán de ser un piloto, no de un software que pudiera estar operativo para un usuario directamente. Por lo cual, el proceso será realizar las grabaciones y luego aplicar distintas técnicas, para posterior para realizar los análisis. No será en tiempo real, aunque una vez establecidas las bases, para conseguirlo sería optimizar el proceso e integrarlo en por ejemplo una aplicación móvil.
- No se buscan mejorar porcentajes de detección de imágenes alcanzados en otros casos. Este TFM, por el contrario, utilizará las técnicas actuales para obtener una información que sea suficientemente válida, y que los informes que se puedan extraer de la información obtenida, permita un análisis por el entrenador o el jugador para mejorar a partir del mismo.

Con respecto a las limitaciones:

- Las grabaciones se realizarán desde una cámara fija. Por facilitar los procesos, se partirá de que las imágenes de una misma sesión de tenis, tendrán el mismo enfoque, lo que facilita no tener que realizar ajustes constantes para ir ajustando x puntos significativos de cada imagen para recolocar espacialmente los objetos. Esto simplifica el proceso de seguimiento de pelota y jugador. Los vídeos se graban desde una cámara con trípode o similar desde el fondo de la pista. En cualquier caso, si se quisiera generalizar a imágenes desde distintos puntos de vista, hay diferentes soluciones que identifican puntos clave en las imágenes, que son capaces de solucionar este problema con una gran precisión.
- El punto de grabación es desde un fondo, detrás de dónde está el jugador, lo que facilita hacer el seguimiento de este, pero dificulta o directamente descarta realizar lo equivalente con el adversario. Este punto es adecuado porque habitualmente es accesible el fondo de las pistas y es donde se suelen situar los entrenadores y familias en partidos y entrenamientos amateur. Si la técnica funciona, simplemente con añadir una cámara al otro fondo se podrían combinar ambas, conteniendo el cruce la información completa para ambos jugadores. El objetivo puede ser ambicioso para el

tiempo de desarrollo de este TFM, pero se espera ofrecer información útil y que sirva de semilla para posteriores mejoras.

A modo de resumen no se busca mejorar ninguna técnica actual, sino más bien su utilización práctica, y adaptarla a un entorno real, lejos de una situación ideal con vídeos de calidad alta.

1.6 Capitulación

El documento se organiza en siete capítulos, los cuales se mencionan a continuación:

El primer capítulo es una introducción del TFM, donde se presenta el problema y se definen los objetivos a alcanzar.

El segundo capítulo, presenta el estado del arte que da cuenta de qué se está haciendo en estos momentos en el ámbito del tratamiento de imágenes, sobre todo referidas al deporte y en concreto al tenis. Esta parte identifica distintas técnicas disponibles sobre las que se apoya este TFM.

En el tercer capítulo, se indica la metodología de investigación diseñada para la ejecución de este trabajo, desde el enfoque, el diseño hasta los instrumentos de recolección de datos.

El cuarto capítulo muestra el desarrollo práctico del piloto, desde el funcionamiento de la herramienta que integra la técnica de IA hasta el escenario de experimentación. Los resultados se incluyen en este apartado, así como los problemas encontrados durante el proceso.

El quinto capítulo presenta las conclusiones alcanzadas con la realización de la investigación, incluyendo aspectos que pueden ser mejorados y trabajos futuros.

Finalmente, el capítulo sexto incluye las referencias y el séptimo los anexos.

2 CONTEXTO Y ESTADO DEL ARTE

2.1 Vista general

Este capítulo comienza mostrando una pequeña introducción al deporte del tenis, apartado 2.2. Seguidamente se presenta una breve presentación de los términos más técnicos relacionados con la IA utilizados en el TFM, apartado 2.3. También se presentan proyectos en los que se está aplicando la visión artificial, para finalmente revisar el estado del arte sobre visión artificial y artículos enfocados en el procesamiento de imágenes de tenis, apartado 2.4. Se divide en sub apartados según las diferentes técnicas que son necesarias aplicar en este piloto para conseguir los objetivos, además de referencias a la importancia de un entrenador digital que procese los datos automáticamente.

2.2 El Tenis

Aquí se comentan conocimientos básicos sobre el tenis, para poder entender apartados posteriores. El tenis se trata de un deporte normalmente individual, uno contra uno, aunque la modalidad de dobles cuenta con cierta relevancia. Las reglas del tenis son complejas y extensas para explicar, pero a continuación se incluye un pequeño resumen de las más importantes:

- Cada jugador debe golpear la pelota y debe caer en la pista contraria dentro de las líneas que definen el campo. Para ello deberá superar la red.
- Todas las pistas del mundo tienen medidas exactas, solo variando la distancia con las vallas que delimitan la pista en el exterior.
- La pelota, en competiciones oficiales, es siempre amarilla y del mismo tamaño, solo variando la composición y presión de la misma.
- El juego se suele desarrollar en el fondo, es decir, los jugadores golpean normalmente por detrás de la denominada línea de fondo.
- Cuando un jugador golpea una pelota que viene por su derecha se llama a este golpe derecha o forehand en inglés. Si por el contrario viene por la izquierda se llama revés o backhand. Esto, naturalmente, cambia en los jugadores zurdos.
- Se dice que un golpe es “paralelo” cuando más o menos sigue la dirección de las líneas laterales, y “cruzado” cuando por ejemplo va desde el lado izquierdo de la pista de un jugador, al lado izquierdo del contrario.

- Las velocidades de la bola en profesionales son superiores a los 200 km/hora en los saques y de 100 km/hora en los peloteos desde el fondo de la pista. Esta velocidad baja en categorías inferiores, aunque no demasiado, ya que la diferencia entre unos y otros es más de precisión que de fuerza.

Las dimensiones de la pista, así como su forma se pueden ver en la imagen 1.

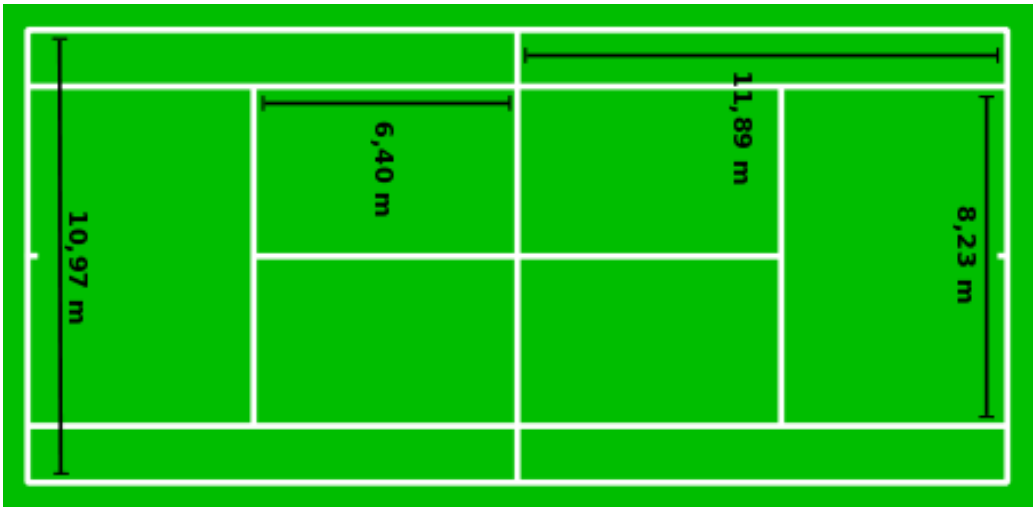

Imagen 1. Medidas Pista de Tenis

2.3 Inteligencia Artificial

No existe una definición única aceptada, ya que la mayoría son ambiguas o incompletas. La definición del English Oxford Living Dictionary es la siguiente:

> La teoría y el desarrollo de sistemas computacionales para llevar a cabo tareas que normalmente requieren de inteligencia humana, como percepción visual, reconocimiento de voz, toma de decisiones o traducción de lenguajes.

La IA aglutina distintas ramas, y se amplía información sobre las que se apoya este TFM en los próximos apartados.

2.3.1 Machine Learning

Una definición de Machine Learning (ML), Aprendizaje Automático en Español, es la expuesta por Rouse (2017):

> El aprendizaje automático es un tipo de inteligencia artificial (AI) que proporciona a las computadoras la capacidad de aprender, sin ser programadas explícitamente. El

> aprendizaje automático se centra en el desarrollo de programas informáticos que pueden cambiar cuando se exponen a nuevos datos (p.1).

Esta área comprende una gran variedad de métodos que, basados en unos datos de entrada dentro un entorno concreto, busca predecir el resultado esperado ante nuevos datos de entrada. Por ejemplo, para la necesidad de conocer si el precio de una vivienda está ajustado a la realidad, se puede basar en datos históricos sobre otros pisos. Si estos datos contienen información sobre el precio, el tamaño, ubicación, y el dato que se quiere obtener, el precio al que se han vendido, se puede preparar un modelo que prediga el valor del piso conocidos los datos de entrada (tamaño, ubicación).

Los diferentes métodos de ML se suelen dividir en dos grandes grupos: aprendizaje supervisado y no supervisado. El supervisado requiere conocer el valor resultado real o etiquetados. Dentro de este grupo están las redes de bayes, árboles de decisión, SVM (Support Vector Machine) o random forest (raona, 2017). Y los no supervisados, donde no se conoce el resultado real, o simplemente se quiere realizar una agrupación por datos de entrada similares o clustering, lo que hacen estos algoritmos es buscar los elementos de entrada más parecidos y juntarlos en conjuntos. Este TFM no cuenta con datos etiquetados, son costosos de conseguir, por lo que se ha empleado una técnica de aprendizaje no supervisado. En concreto, se ha utilizado uno de los más populares, K-Means, al que le das como parámetro de entrada el número de grupos que quieres crear.

Dentro del área del ML, se encuentra un sub área de investigación que se conoce como “redes neuronales”, que son un sistema de computación que puede actuar o simular actuar de la misma forma en la que funciona el cerebro humano. Y, si se sigue profundizando dentro del área de las redes neuronales, se encuentra el campo del Deep Learning, que no son más que redes neuronales artificiales compuestas por una gran cantidad de capas organizadas de forma jerárquica. A continuación se explican con mayor detalles estos conceptos.

2.3.1.1 Redes Neuronales y Deep Learning

Las redes de neuronas artificiales se pueden definir como:

> Paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. Forman parte de los denominados “Sistemas Inteligentes“, dentro de la rama de la Inteligencia Artificial (Marti-Vargas, Ferri, & Yepes, 2013).

Dichas neuronas se configuran en estructuras que pueden variar según la complejidad y tipología del problema a resolver. Matemáticamente, se puede definir como una red computacional.

El funcionamiento matemático de una red de neuronal es como el de una función f que recibe ($x_1x_{2,}$…,x_n) parámetros, para devolver (y_1,y_2,…,y_m) resultados (IBM, 2019). Sin entrar en el funcionamiento de las redes, al ser especialmente complejo, cada neurona tiene una serie de parámetros que se multiplican por los valores de entrada, y a estos se les aplica una función, llamada de activación, para generar una salida. Durante el ajuste o entrenamiento del modelo, se le pasan a la red los valores de entrada, y en la salida de la red se evalúa si ha acertado con el resultado. Si el resultado ha sido erróneo (al principio del entrenamiento será lo habitual), se ajustan los parámetros de cada neurona, para que idealmente, la próxima vez detecte este valor correctamente. Un ejemplo de red de neuronas artificial se ve en la imagen 2:

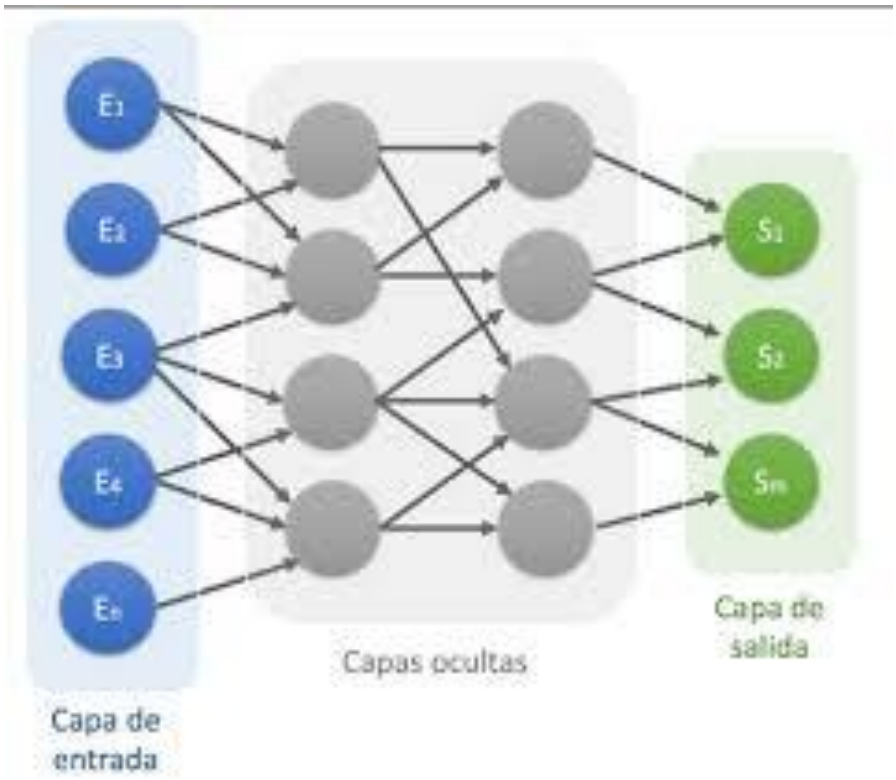

Imagen 2. Ejemplo Red Ne*uronal*

En este contexto, Deep Learning se utiliza para denominar a redes neuronales, pero con mayor profundidad en el número de capas (López Briega, 2017). La mayor capacidad de proceso, sobre todo con el boom del cloud computing, permite calcular en tiempos razonables redes con millones de parámetros a calcular.

Un tipo de red neuronal, que suelen ser redes profundas, son las redes convolucionales. Se utilizan porque se han probado muy útiles en la detección de objetos, y en general en el tratamiento de imágenes (Calvo, 2017). El mayor problema de tratar imágenes con redes neuronales es el número de parámetros de entrada, 777.600 en una imagen de 720x1280. Lo que hacen estas redes es, utilizar una cantidad fija de parámetros en las primeras capas, independientemente del tamaño de la imagen. Es como si a la imagen se le fuera pasando una máscara por cada parte, y el resultado de la máscara se guarda. Visualmente se puede ver en la imagen 3.

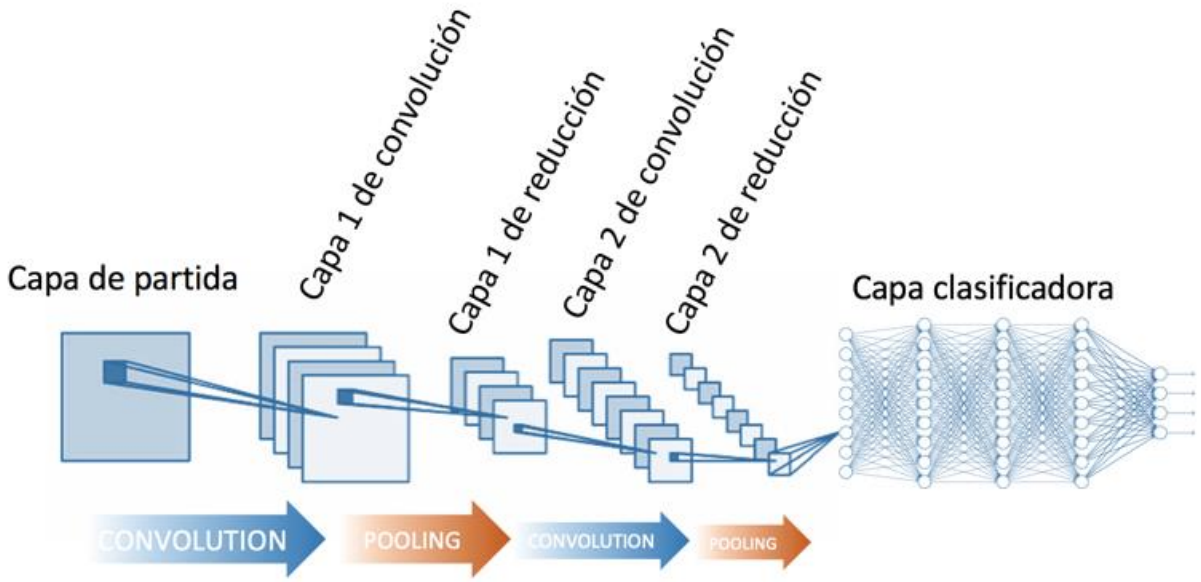

Imagen 3. Ejemplo Red Convolucional

La idea es aplicar múltiples capas, en las que se espera que las primeras detecten detalles de imagen grandes como si es muy obscuro o clara, las siguientes detecten bordes, para finalmente ir detectando pequeños detalles que sirvan para predecir el objeto correctamente.

Otro tipo de red, son las Redes Neuronales Recurrentes (RNN), las cuales son redes especializadas para casos en el que los datos de entradas pueden tratarse como una secuencia. Se utiliza mucho en procesamiento del lenguaje tanto para analizar la “semántica” del texto como en traducciones. También tienen uso en el tratamiento de vídeos al tratarse de una secuencia de imágenes, lo que puede facilitar la detección acciones.

Las redes neuronales, sobre todo en el caso de Deep Learning, requieren muchos ejemplos para entrenar, del orden de miles o millones, y altas capacidades de computación para reducir

el tiempo de entrenamientos. Para solventar la falta de datos o potencia de cálculo, se emplean habitualmente dos técnicas..

La primera es Data Augmentation (Ho, Daniel; Liang, Eric; Liaw, 2019), que a partir de un conjunto reducido de datos de entrada, aumentan el número de ejemplos. El caso concreto de las imágenes consiste en modificarlas ligeramente, como girar un poco la imagen, oscurecerla o invertirla en espejo.

Transfer Learning es la segunda técnica (López Briega, 2017) , en la que se utiliza un modelo ya entrenado por un tercero, normalmente grandes compañías como Microsoft o Google. Existen varias alternativas, pero lo que se suele hacer es aprovechar que ese modelo funciona bien con imágenes similares a las del problema que se quiere resolver, y se reentrena con las que se disponen. Así, se aprovecha una gran parte del conocimiento de una gran red neuronal cuando no se tiene la posibilidad real de entrenar con los medios disponibles.

El entrenamiento de las redes es otro punto importante dentro de esta área, en los que utilizan varios conceptos (Brownlee, 2016; López Briega, 2017) :

- Overfitting o sobreajuste: Esto ocurre cuando la red neuronal “aprende” muchos o todos los ejemplos utilizados en el entrenamiento, pero funciona sensiblemente peor con datos reales.
- Underfitting: Es lo contrario al caso anterior, en el cual no es capaz de predecir ni si quiera los datos de entrenamiento.
- Regulación: Distintos métodos empleados para que el entrenamiento no caiga en overfitting.
- Batch Normalitation: técnica de regulación, que además de agilizar el entrenamiento, ajusta los parámetros de la red con la media de los resultados de n elementos de entrada. Esto evita que la red aprenda el resultado para un solo elemento y sí para un conjunto.
- Learning Rate: Es un valor que se utiliza para indicar a la red cuánto debe aprender. Una técnica habitual consiste en ir reduciendo este valor para que a medida que la red está más entrenada, y en teoría más cerca del ajusto óptimo, aprenda menos cada vez para que no se pase del ajuste bueno.

2.3.2 Visión Artificial

La visión artificial incluye una serie de técnicas que permiten actuar sobre imágenes o vídeos modificándolas, permitiendo obtener información sobre ellas. Puede ser simplemente artística, como realizaría un fotógrafo para ensalzar cierta parte de la imagen que le interesa, hasta el

análisis del iris de una persona en un sistema de seguridad. A continuación, se mencionan las técnicas o conceptos empleadas en este TFM dentro de esta área.

- Entropía: Magnitud física de un sistema que mide su equilibrio. Vocablo utilizado originalmente en termodinámica, aplicado a señales (y por extensión a imágenes si se tratan como una señal temporal), mide la aleatoriedad de la señal (Orduz, 2017). Por ejemplo, una nota constante dará daría una entropía de cero (idealmente) y un ruido aleatorio sería el punto de máxima entropía. Este valor es útil para estudiar cambio en la señal y tiene diferentes usos.
- Diferencias de Imágenes: una técnica muy útil para detectar movimientos en vídeos es tomar una imagen y "restar" la anterior. Esto provoca que solo las partes que distintas en las dos imágenes queden en la imagen resultante (si un pixel tiene un valor de 200 y en la siguiente imagen es igual, el resultado de la resta será 0). Esta técnica útil, no obstante, tiene sus problemas, dado que un elemento negro podría confundirse. Además, la intensidad de la luz no es constante, por lo que la resta en un punto estático puede no ser cero exactamente, lo que implica tener que realizar ciertos ajustes y establecer un umbral a partir del se considere que es cero o no (estático o movimiento).
- Binarización o Threshold: técnica por lo que a una imagen en escala de grises, con valores por ejemplo entre 0 y 255, se convierte para que solo tengo los valores 0 y 255 (negro y blanco). Se necesita un valor, por ejemplo 50, que sea el límite que indique si ese píxel se convierte a blanco o a negro. Se utiliza para resaltar las partes de la imagen que más interesan.
- Segmentación de Imágenes: conseguir de manera automática diferenciar distintas partes de una imagen con algún objetivo. Reconocer objetos, marcar donde está el mar o una línea recta en la imagen serían ejemplos.
- Operadores Morfológicos en Imágenes**:** tratamiento a una imagen que puede ayudar en la segmentación (Barrionuevo, 2009). Se aplica a la imagen algún tipo de estructura espacial (como una cruz o u círculo) con el objetivo remarcar o difuminar ciertas partes de imagen en función de su forma.
- Extracción de Características de un Imagen: proceso que se lleva a cabo para conseguir ciertos datos de una imagen, para que estos a su vez sirvan a otro objetivo. Si se quiere analizar que imágenes son de un campo de fútbol de las que no, se puede buscar si contiene mucho color verde. Características posibles son innumerables, que van desde datos estadísticos como medias y varianzas (se trata la imagen como una señal temporal), a más complejas buscando áreas significativas, como se haría en un detector de huellas.

- SIFT (Scale-invariant feature transform)**:** algoritmo avanzado de detección de características en imágenes. Tiene el objetivo de encontrar que partes de la imagen tiene algún patrón único e identificable (Ning, 2019). Esto permite, por ejemplo, dado una imagen de muestra de un objeto, como puede ser una pelota, encontrarlo en cualquier imagen, buscando esa "huella digital" coincidente con la imagen de muestra.

2.3.3 Campos de Aplicación de la Visión Artificial

La visión artificial es un área de la IA que en los últimos 5 años ha tenido un gran empujón gracias a técnicas nuevas como las redes neuronales profundas, convolucionales y recurrentes. Algunas de estas técnicas no son tan novedosas, pero es desde hace poco tiempo que se están aplicando gracias a las mejoras en las técnicas y a la mayor capacidad de computación de hasta miles de equipos en paralelo, utilizando sus tarjetas gráficas que a su vez están especialmente indicadas para el trabajo con vectores. Por ello, se están empezando a ver aplicaciones reales complejas, más allá de aquellas que resultaron novedosas en su momento como lectura de etiquetas, matrículas o formularios.

A continuación, se describen distintos proyectos implantados o en estado de desarrollo avanzado. Se incluyen ejemplos en segmentos diversos para dar la idea de la gran variabilidad de aplicaciones existentes gracias a la visión artificial (Ayn de, 2019).

Dentro del sector del retail, empresas como Amazon, están aprovechando los últimos avances en IA para mejorar todos los puntos de su negocio principal, la venta de bienes. Amazon Go es un prototipo de tienda en la que los compradores no tienen que esperar cola para pagar, de hecho, no existen cajeros ni automáticos ni humanos. La tienda cuenta con cámaras en el techo que van registrando los objetos que cada usuario va cogiendo. Por ahora secciones como la de bebidas alcohólicas están exentas, además de que hay personas ayudando a los algoritmos a clasificar adecuadamente a los objetos. Una primera tienda abrió en 2018 en Nueva York, y aunque no está claro si el prototipo de tienda se extenderá, Amazon ha registrado el nombre Go trademarks por si tiene éxito el proyecto. Un uso más inesperado es el registro de patente de "Espejo Virtual", en el que se puede mirar una persona y el espejo le muestra con distintos modelos de ropa. Este proyecto, teniendo en cuenta que básicamente se dedican a la venta online, podría ser una "killer app" que haría que se disparara la venta online de ropa, al poder tener una mejor idea sobre cómo quedaría la ropa puesta, e incluso si es la talla adecuada si la aplicación consigue una buena precisión. StopLift ha desarrollado un sistema para chequear si las personas en las de los supermercados pasan por el escáner todos los productos, o si, por el contrario, cuando el cliente es un conocido o familiar, no

chequea todos los productos produciéndose el robo de una manera difícilmente detectable. Este sistema va “leyendo” los productos y compara con lo que el cajero va registrando, detectando las posibles “desviaciones” en la lista de la compra. Más explotado está el uso para la mejora de temas logísticos, como el uso de robots y drones para mejorar la distribución, como ya hace Waltmark. Este sub segmento, junto con los avances en robótica probablemente consigan automatizar una gran parte del trabajo en almacenes y distribución de productos. (Nakhuda, 2019)

Los avances en la automoción pueden poner patas arriba los transportes y las comunicaciones terrestres. El estado actual todavía no son vehículos que puedan conducir sin supervisión humana, son capaces de circular razonablemente bien en circunstancias normales y puede que no tarden en ser sistemas lo suficientemente maduros para empezar a sustituir a conductores. Tesla cuenta actualmente con su sistema autopilot que conduce solo el coche, aunque requiere intervención humana para algunas situaciones. Recientemente ha habido un par de polémicas por este sistema por el atropello mortal de un peatón, en el que el conductor no estaba haciendo caso al tráfico, así como de algunos accidentes de los que no está clara la responsabilidad. También Waimo, antiguamente conocida como Google self-driving car Project, informa que han testeado sus modelos en 7 millones de millas con tráfico real, y que son capaces de entender los elementos necesarios para la conducción como señales, semáforos, rastrear los movimientos de objetos en 300 metros a la redonda, y hasta si un ciclista levanta la mano indicando que va a girar. Actualmente Waimo One (Hawkins, 2018) es un servicio de taxis en Phoenix en el que el vehículo conduce solo, aunque durante el lanzamiento se incluye un piloto experto por si requiriera alguna intervención. No se sabe el momento en el que se excluirá al humano por prevención.

La visón artificial, dentro de la sanidad, está trabajando en muchos proyectos en la detección de enfermedades, ayudas a diagnósticos. Gauss Surgical ha desarrollado un monitor que estima la sangre que pierde un paciente durante una intervención. Durante una intervención es difícil medir las hemorragias del paciente, y este sistema ayuda a optimizar el uso de la sangre almacenada (solo se suministra la necesaria) además de ayudar al cirujano a conocer mejor la situación del paciente. DermLens es un proyecto de una startup que utilizando las lentes de Amazon Web Services (AWS) ayuda a la detección de la soriasis, estudiando el estado de la piel. Otro avance es el uso de la IA para leer escáneres y radiografías automáticamente. Recientemente Google ha presentado un caso en el que la IA era mejor que 6 expertos. Estos casos todavía están en fases experimentales, pero en un futuro probablemente sean una herramienta más de los doctores, y quien sabe si en algún futuro cercano, sustituirles en las labores de diagnosis. (K. Johnson, 2019)

La agricultura más asociada a maquinaria, también empieza a verse favorecida por la IA. Slantrange anuncian que gracias a drones equipados con cámaras, recorren las plantaciones y son capaces de detectar el estado de las cosechas. Hacen una labor que a los humanos es prácticamente imposible realizar en grandes terrenos, ya que pueden inspeccionar constantemente sin un gran coste toda la plantación. Se consigue una optimización en el uso de fertilizantes, ya que se usan cuando son necesarios, y se obtienen mejores cosechas gracias a esta supervisión continua. Esto redunda en ahorros entre 9 y 13 $ por acre, que sumado a mejores producciones amplia los beneficios. Una aproximación similar, pero hacia el ganado, Cainthus consigue mediante reconocimiento facial al animal, obtener información sobre la salud del animal, como temperatura, cantidad de agua y comida que ingiere, enviado alertas a los granjeros cuando un animal tiene algún problema de salud.

El sector de la seguridad es el que probablemente cuenta con más aplicación reales o en desarrollo. Desde reconocimiento facial, objetos perdidos (que pueden ser una bomba), validación para accesos a bancos o edificios, hasta reconociendo documentos como pasaportes. La lista de aplicaciones aquí es muy interminable. Probablemente la más impresionante a la par de inquietante, es el gran hermano chino, que pretende monitorizar y conocer donde se encuentra en cada momento sus ciudadanos. Probablemente falte algo de tiempo para que sea efectivo, pero las técnicas están disponibles, solo falta inversión e ir refinando el sistema (Molins renter, 2018) (Jacobs, 2018). Un proyecto similar, son una gafas que lleva la policía china, en la que les avisa cuando vean a una persona buscada por algún delito (Vincent, 2018).

Las fábricas también será otro de esos sectores en que la automatización entrará fuerte, cambiando el sector para siempre. Robot´s que son capaces de ver las piezas que deben colocar hacen más fácil implementar tareas hasta ahora exclusivas para humanos. En otro casos, como el de inspecciones de pozos petrolíferos, hasta ahora requerían de visitas periódicas de personal especializado y que Osprey Informatics (Ayn de, 2019) consiguen monitorizarlo, consiguiendo ahorros en un sector en el que el precio es fundamental.

A la luz de estos avances, nacen empresas especializadas en visión artificial (Hettena, 2018) como las 4 startups que se mencionan seguidamente. Estas compañías cuentan con pocos empleados, pero son altamente especializadas, y seguro que en los próximos años crecerán exponencialmente. Pilot.ai especialistas en HW y SW de reconocimiento, con una larga lista de clientes anónimos, dado que el nivel de su software puede atentar con la privacidad. Un ejemplo se puede apreciar en la imagen 4.

Imagen 4. Ejemplo Detección Objetos

Consiguen un gran nivel a la hora de clasificar objetos, detectar a un tirador durante un atentado, y ofrecen integraciones personalizadas por proyectos. Shazura de origen español, con un algoritmo propio es capaz de obtener una huella única de un objeto a partir de una sola imagen, lo que facilita enormemente el reconocimiento. 20 Billion Neurons (20BN) son capaces de detectar gestos sutiles en humanos, como saber quién mira a quien, o si alguien bebe agua. Sería susceptible de emplearse, por ejemplo, en reconocimiento de lenguaje de signos, o eliminación del uso de ratones físicos para manejar los ordenadores. EVK usan espectros de la luz que los humanos pueden ver, pero que son útiles para resolver problemas como controles de calidad, diferenciación de plásticos, identificación de suciedad.

2.4 Estado del Arte

2.4.1 Investigaciones punteras del ámbito Visión Artificial

Los artículos relacionados con la visión artificial se cuentan por cientos, pero se han seleccionado los más interesantes o que han sido más premiados y citados recientemente.

El uso de las CNN en la clasificación de imágenes en IA es el método que mejores resultados ha ofrecido hasta ahora para imágenes en 2D. Este artículo aborda la necesidad de genera modelos en 3D, a partir de imágenes en 2D (Cohen, Geiger, Koehler, & Welling, 2018). Al proyectar una imagen esférica en 2D la deforma, como ocurre con los mapas. El framework que han creado ayuda a tratar con este problema. Esto ocurre, por ejemplo, con las cámaras de omnidireccionales de seguridad. Esta evolución permite clasificar objetos en 3D, solucionar problemas de regresión en la industria química o usarse en para modelos climáticos.

Otro estudio explora el estudio de imágenes que consigue engañar a los algoritmos de reconocimiento (sin utilizar el conocimiento del algoritmo concreto, acercamiento de caja negra), y comprobar si estas imágenes también engañan a humanos (Elsayed et al., 2018). Esto ha provocado cierta controversia, ya que, según los opositores, si un humano no reconoce una imagen como un "gato" (cuando así está clasificada la imagen), podría considerarse que efectivamente no es un gato y no debería tratarse como un error en el etiquetado de la imagen.

El debate sigue abierto sobre los casos en los que falla la IA. Una mejora en los procesos de traspaso de estilos entre imágenes queda propuesta en (Li, Liu, Li, Yang, & Kautz, 2018), consiguiendo evitar inconsistencias en el "estilización". Además, mejora el traspase no solo de los colores, si no de pequeñas estructuras. Fotógrafos profesionales hablan sobre lo que pueden conseguir con esta tecnología en entornos reales.

Batch normalitation es una técnica que se utiliza para evitar que los modelos se sobreajusten a los datos de entrenamiento, es decir, que tengan muy buen acierto con los datos de entrenamiento, pero sin embargo funcionen mucho peor con datos no vistos por el modelo. Lo que hace es hacer paquetes de datos, y sobre la media de sus resultados, aplicar cambios al modelo. Esta técnica consigue que valores extraños no afecten al modelo, porque al realizarse una media quedan diluidos. Se puede aplicar cuando hay grandes conjuntos de datos, pero no funciona igual de bien para datasets con pocos datos. El estudio de Wu & He (2018) ataca esta problemática, porque hay ciertos problemas donde es imposible crear tantos datos de entrenamiento. Group Normalitation consigue mejoras frente a otras normalizaciones, sobre todo en entornos de visión artificial como detección de objetos o segmentación.

Finalmente en reconocimiento de objetos, especialmente en entornos reales donde hay miles de objetos posibles, los autores Zamir, Sax, Shen, Guibas, Malik & Savarese (2018) descubren relaciones entre distintas tareas comunes en visión artificial, consiguiendo modelos igual de eficaces con $\frac{2}{3}$ de los datos. Es especialmente útil para entornos reales, donde hay que identificar miles de objetos distintos, facilitando la preparación de los datos. También, han creado un dataset con 4 millones de imágenes de interior, anotadas para las 26 tareas que presentan.

2.4.2 Investigaciones Aplicadas al Tenis

Actualmente hay algunas empresas que desde hace algún tiempo están avanzando para tratar digitalmente el rendimiento de jugadores y equipos, facilitando las labores tradicionales de ojeadores y entrenadores.

Las empresas Nacsports ("Nacsport | Software de Vídeo Análisis para todos los Deportes," 2019) y Logonmatch *("LongoMatch | La herramienta de videoanálisis para entrenadores, analistas deportivos y jugadores," 2019)*, han montado una plataforma, en la que dado un vídeo, se crea una plantilla en la que tendrás botones por cada acción que deseas registrar. Después, cuando se visiona el vídeo, manualmente se va pulsando el botón que corresponda según los lances del juego. Esto permite de una forma digital registrar y luego analizar la información, pero no evita que sea un humano quien etiquete las acciones, además de que monitorizar diferentes parámetros se complicaría mucho a partir de cierto número. Requiere la intervención humana y aunque mejor que en una libreta, está todavía muy alejado de algo realmente automático.

Una escuela de tenis iTusa Tennis (ITusa, 2019), ofrece análisis de videos, partidos completos o un golpe. Dan mucha importancia a estos análisis, aunque no muestran ninguna referencia al software que emplean. Se envía un video, y luego contestan, probablemente utilizando métodos manuales, probablemente llevados a cabo por algún entrenador.

Los siguientes sub apartados, agrupan diversos artículos según sus objetivos.

2.4.2.1 Detección de Movimientos

Las técnicas de detección de movimiento con cámaras fijas son relativamente simples. Básicamente consisten en coger una imagen de la secuencia del vídeo y compararla con otra en otro momento temporal anterior. Obteniendo que partes de la imagen han cambiado sirven para inferir que se mueve. El problema es que en vídeos reales, el color de la pista no será igual, habrá árboles que se muevan por el viento, personas pasando por el fondo de la pista, lo que obviamente complicará la labor de quedarnos con los objetos que interesan. Además, las condiciones de luz, diurnas, sombras en la pista, atardeceres o directamente de noche complicarán conseguir unos parámetros que funcionen en todas las condiciones.

Algunas investigaciones usan trucos para obtener la información, como que la pista tiene un color constante, que las cámaras de televisión después del fallo de un jugador, suele cambiar el vídeo a un plano general (Teachabarikiti, Chalidabhongse, & Thammano, 2010). Adrian Rosebrock mantiene un blog estupendo sobre tratamiento de imágenes, tocando

diferentes técnicas, y siempre montando un pequeño piloto de demostración. Se comentan algunas entradas en el blog que enseñan cómo hacer en Python algunas tareas que resultan útiles. En (Rosebrock, 2016) se enseña cómo implementar una sencilla cámara de seguridad, pero como en el fondo, lo que hace es detectar movimientos, sirve como ejemplo de partida. En (Rosebrock, 2015) se muestra parte de la problemática de querer obtener solo la pelota de todos los objetos que se mueven en la imagen. La técnica que utiliza es segmentar por color, es decir, ir buscando en la imagen donde está el color amarillo de la pelota. Es un ejemplo excesivamente simple, pero que sirve como ejemplo de esta técnica en Python. Otra técnica es buscar las Saliency zones en la imagen. Estas zonas son las más relevantes como como pueden ser bordes, formas concretas, etc (Rosebrock, 2018). La detección de estas zonas puede ayudar a la detección de los objetos importantes en las imágenes, como las líneas de la pista, la pelota o el jugador.

Una técnica más avanzada con las Saliency Zones profundiza en las zonas relevantes de las imágenes, pero aplicadas a video (Chenlei Guo, Qi Ma, & Liming Zhang, 2008). Se utiliza como base a la transformada de Fourier, ofreciendo una nueva técnica para conseguir las Saliency Zones en las imágenes. Además es un método pensado para aplicar a vídeos. La aplicabilidad directa puede ser compleja dado el tipo de cálculos necesarios, pero ofrece otra opción que puede ser útil. Otro artículo más actual sobre la misma área, en el que se evalúan las zonas de atención, en el que utilizan redes convolucionales/recurrentes adicionalmente, consiguiendo mejores resultados que otros modelos (Wang, Shen, Guo, Cheng, & Borji, 2018). También se explora el mismo tema en el estudio de Fang, Cao, Yang, & Xing (2019) donde se vuelve a proponer un nuevo modelo de redes neuronales con las saliency zones. Los tres artículos pueden ayudar a otros métodos para fijar los puntos importantes en la imagen, permitiendo optimizar el procesamiento y mejorar la precisión.

Otro estudio centrado en el seguimiento de pelotas de tenis en vídeos es el desarrollado por los autores azi, Mukherjee, Srivastava, Lall, & Chauhan (2015), quienes proponen una metodología en distintos pasos para seguir la pelota en las imágenes. Cada paso usa distintas técnicas, e incluso las combinan; entre ellas usan un algoritmo para identificar puntos clave en las imágenes, puesto que los videos que utilizan no son de cámara fija, por lo que en cada frame deben identificar como se ha desplazado la cámara. Después de varias fases en las que combinan técnicas como PQFT, segmentación por color, uso de aprendizaje automático con random forest partiendo de imágenes etiquetadas durante el entrenamiento, comprobación de la esfericidad de los objetos candidatos. Consiguen un porcentaje altísimo en la detección, aunque al ser un método muy elaborado multicapa, partiendo de vídeos grabados con cámaras profesionales con buena calidad de imágenes. Usando el concepto de entropía, Chen, Wang, Wang, & Hu (2008) ofrecen otra posible técnica para utilizar en vídeos

deportivos. Proponen tratar el vídeo como una función temporal y estudiar la curva de la entropía, para así detectar los cambios que puedan suceder en cada zona, y donde más varíe esta indicará movimientos.

Otra alternativa que combina distintas técnicas está pensada específicamente para bolas a altas velocidades, combina la técnica básica de diferencias de imágenes, con los momentos de Hu aplicado a los contornos detectados y la esfericidad (Lyu, Liu, Li, & Chen, 2015). La combinación de los tres métodos arroja buenos resultados.

Finalmente, y desde otro enfoque distinto, los autores El Khattabi, Tabii, & Benkaddour (2016) proponen el uso del método clásico SIFT adaptado a vídeos deportivos para detectar movimientos y encontrar los límites de los objetos. Se trata de un proceso complejo que se aplica a cada canal RGB (Red Green Blue).

2.4.2.2 Detección de líneas

La detección de elementos lineales en imágenes es un proceso que usando filtros de realce de bordes como Canny y la transformada de Hough se consigue con cierta facilidad, por lo que no existen demasiados artículos recientes que aborden el tema. Además, al tratarse de imágenes de una pista de tenis, donde se conoce a priori su forma y medidas, con obtener una sola línea y proyectar el resto puede ser suficiente.

Aquí se incluye un artículo que sobre todo trata sobre el tracking de las pelotas de tenis, en las que ofrece un método para poder calcular la trayectoria de la bola y descartar falsos objetos que haría obtener trayectorias falsas (a la par de inverosímiles). Esa técnica puede resultar útil, pero además para conseguir la segmentación de las líneas de la pista, se ayuda del color de las líneas, siempre blanco, lo que mejora y facilita la detección de las mismas (Zhou, Xie, Huang, Cox, & Zhang, 2015).

Una alternativa que evita el uso de la transformada de Hough, método que pesado en cuanto a computación. Primero calcula el background de la imagen (elementos estáticos), y a partir de esta, apoyándose en morfologías matemáticas aplica diferentes estructuras, consiguiendo el resultado buscado (Erol, 2017).

2.4.2.3 Clasificación Golpes de Tenis

La clasificación de imágenes se realizaba tradicionalmente con métodos basados en características de la imagen (como la transformada de Fourier) siendo SIFT un método que ofrece grandes resultados. Los últimos 5 años se ha visto que añadiendo profundidad a las

redes neuronales se consiguen resultados fantásticos, siendo los algoritmos actuales que consiguen los mejores resultados en las competiciones sobre detección de objetos.

Las técnicas que se utilizan en las redes neuronales profundas para el tratamiento de imagen suelen ser redes convolucionales, normalizaciones y distintas capas en profundidades distintas como salida (permite detección de objetos más o menos simples según profundidad). El problema que suelen tener es la necesidad de grandes conjuntos de datos para el entrenamiento, a diferencia de SIFT, que con una sola imagen puede conseguir buenos resultados. A continuación, se muestran artículos que trabajan sobre imágenes de tenis para detectar tipos de golpes. Se descartan artículos que usan tecnología como Kinect para ayudarse en la detección, los cuales se sirven de sensores adicionales

Una opción es el uso de redes neuronales recurrentes dónde partiendo de redes neuronales profundas, se analizan las frames consecutivas apoyándose en redes recurrentes, lo que ayuda a la red a entender la secuencia de imágenes en un golpe dado (Mora & Knottenbelt, 2017). Esto ofrece resultados razonables, pero este tipo de red tiene el problema del tiempo que necesita para entrenar, además de la necesidad de una gran cantidad de ejemplos etiquetados. Otra técnica se vuelca en identificar las articulaciones de los jugadores para que después entrenar un modelo, estimar si el jugador ganará, perderá o seguirá jugando ese punto (Kurose, Hayashi, Ishii, & Aoki, 2018). Muy interesante, y ofrece un punto de partida para, por ejemplo, comparar la técnica de un jugador con modelos perfectos, como podría ser Federer.

2.4.2.4 Estadísticas en Tenis

Este apartado recoge dos artículos que recomiendan el uso de estadísticas en tenis. El primero trabaja con el uso de videos para tratarlos con ayuda del software Kinovea para analizar la efectividad del jugador en cada parte de la pista según el golpe realizado (Losada, Casal, & Ardá, 2015). Doce jugadores profesionales son analizados por grabaciones con cámaras especializadas, para apoyados en el Kinovea averiguar donde es menos efectivo cada jugador, y por tanto donde debería mejorar. Usan una selección previa de puntos escogidos en donde hay un golpe ganador. El artículo es interesante, aunque no entra en demasiado detalle que valores en concreto tienen en cuenta, por lo que es difícil valorar su grado de avance.

Este otro estudio, con el sello de la Real Federación de Tenis, trata sobre la necesidad de obtener información automáticamente sobre los partidos, para estudiarlos posteriormente (Martínez-Gallego, 2015).

3 METODOLOGÍA

3.1 Enfoque Metodológico

Este TFM es un piloto práctico que busca utilizar diferentes técnicas actuales con el objetivo de realizar análisis de vídeos de tenis. El carácter del trabajo es eminentemente práctico y la experiencia previa del autor en metodologías ágiles, marca que el trabajo se desarrolle adaptando el paradigma de gestión de proyecto SCRUM.

La metodología SCRUM aglomera una serie de buenas prácticas para el trabajo en equipo para conseguir los mejores resultados en la ejecución de proyectos (no exclusivamente de desarrollo de software).

Siguiendo las recomendaciones de SCRUM, el trabajo se plantea como una serie de distintos entregables desarrollados por el autor y validados por la directora del trabajo. Estas reuniones sirven para revisar el avance concreto, si se han cumplido con los objetivos marcados y se agendan nuevas tareas para el siguiente sprint. La consecución de estas reglas ayuda que los avances se vayan revisando poco a poco, pudiendo identificar y corregir posibles problemas al principio del desarrollo y no en una fase final como en una metodología clásica. Además, la propia naturaleza de los proyectos de IA, en los que hay un componente muy fuerte asociado a la experimentación, y poca predictibilidad de la calidad de los resultados que se obtendrá al final, hacen especialmente indicados seguir este tipo de metodologías ágiles en el desarrollo de los proyectos o trabajos.

3.2 Diseño Investigación

La forma de afrontar este TFM debe tener en cuenta la parte experimental del mismo; se partirán de vídeos grabados por el autor, simulando situaciones reales en la práctica del tenis, desde los que realizarán los tratamientos necesarios para la obtención de informes analíticos. Por tanto, además de los pasos habituales en cualquier TFM, será necesaria una fase obtención de los datos. Las distintas fases que se irán siguiendo son las siguientes:

- Búsqueda de proyectos y artículos para estudiar la viabilidad del trabajo, así como para recopilar posibles métodos y tecnologías a emplear.
- Análisis de cómo llevar a cabo el trabajo, identificación de posibles objetivos a conseguir, posibles métodos a emplear de los analizados en el paso anterior.

- Obtención de los datos sobre los que trabajar, que se deberá realizar una vez estén claros los objetivos y métodos a emplear. Es una actividad la realiza el autor del TFM, y se realiza en diferentes sesiones y buscando la mayor variedad de los jugadores.
- Inicio de elaboración del documento del TFM en los capítulos de introducción y estado del arte.
- El procesamiento de las imágenes, aplicando las técnicas identificadas para conseguir los objetivos marcados. Esta fase se realiza en distintas iteraciones, buscando obtener o mejorar un objetivo concreto en cada una de ellas.
- Combinación del procesamiento de las imágenes del punto anterior, para poder crear informes con mayor valora al unir la distinta información.
- Plasmar los resultados de las fases prácticas en este documento, incluyendo las gráficas y tablas que se estimen para plasmar de la mejor forma los resultados conseguidos. Incluir imágenes con ejemplos sobre las detecciones incluidas.
- Añadir conclusiones y próximos trabajos a partir de este.
- Revisión del TFM e incluir correcciones y mejoras.

Estas fases, aunque en parte secuenciales, siguiendo la citada metodología SCRUM no seguirán un estricto flujo lineal sino más iterativo y dinámico, en los que los avances o problemas encontrados en cada sprint, provocan cambios a lo planteado inicialmente.

3.3 Recolección de Datos: técnicas y fuentes

Todos los datos utilizados en este trabajo están recopilados por el autor, en situaciones reales dentro del marco definido en el TFM.

Los datos sobre los que se basa el TFM son vídeos de tenis grabados con una cámara réflex Canon EOS 1100D (gama medio-baja) y un trípode colocado en un fondo de la pista de tenis, Se han realizado dos tipos de grabaciones bien diferenciadas: la primera vídeos grabados de dos jugadores jugando al tenis, bien en entrenamiento o en partidos, siendo los segundos, vídeos utilizados para entrenar un modelo que reconozca los diferentes golpes de tenis. Se ha buscado la diversidad de los jugadores, en la medida de lo posible, y utilizado técnicas de data augmentation en la parte de clasificación de golpes.

Los vídeos destinados para clasificación de golpes, además incluyen otros en los que no hay ningún golpe, para que el modelo identifique adecuadamente los momentos en los que no hay golpeos. El lugar donde se almacenan es gDrive, puesto que para el procesamiento de los vídeos se utilizará la herramienta Colab ofrecida por Google, en la que permite

implementar notebooks en Python, aprovechando su infraestructura en la nube, más potente que un portátil de gama media o con tecnología de hace unos pocos años. La GPU que ofrece Google tiene 12 Gb de memoria, comparados con los 2Gb un portátil normal.

La elaboración de los vídeos se realiza bajo diferentes circunstancias, como diferentes pistas, luz ambiente disponible, solar o artificial, bolas nuevas o usadas (que tienen distinto color), distintos colores de las camisetas de los jugadores

Los videos se convierten a listas de imágenes para tratarlas individualmente y posteriormente relacionarlas con las adyacentes. El número de frames por segundo de la cámara utilizada es de 25, lo que, unido a una calidad media de la cámara, los análisis de las imágenes se vuelven relativamente complejos, dónde en algunos frames no se aprecia dónde se encuentra la pelota o la raqueta del jugador, apareciendo como una sombre muy difuminada. Además, la pelota cuando está lejos de la cámara tiene un tamaño muy pequeño, haciendo difícil la tarea de segmentarla, incluso a humanos, pareciendo más una mota de polvo o una sombra de un pájaro más que la bola.

4 DESARROLLO TÉCNICO DEL TFM

4.1 Introducción

Este capítulo aborda las técnicas y resultados obtenidos para dar respuesta a los objetivos planteados en el apartado 1.4.

Se aborda cada uno de los cuatro objetivos específicos por separado, para una posterior integración que aporta la integralidad de la solución propuesta, la cual conlleva al alcance del objetivo general de ofrecer un piloto que recopila datos útiles para el jugador o entrenador, a partir de los vídeos.

El apartado 4.2 explica cómo se desarrollado y las técnicas empleadas para cada objetivo. Se explica técnicamente como se aborda cada objetivo y se incluye las partes de código en Python más significativas, ya que incluir todos los procedimientos auxiliares y pasos intermedios restaría legibilidad a las explicaciones.

El apartado de resultados muestra el nivel de éxito obtenido por el piloto, y finalmente los problemas encontrados, especialmente en cuanto al reconocimiento del tipo de golpe del jugador.

4.2 Desarrollo piloto

El desarrollo del piloto se ha realizado enteramente en Python sobre Google Colab. Esta herramienta online facilita el desarrollo al poder acceder a ella desde cualquier equipo, además de apoyarse en las capacidades de computo gratuitas que la compañía pone a disposición de los desarrolladores/investigadores. Cuenta con ciertas limitaciones como un uso máximo de 12 horas continuadas, y el hecho que incluso se compartan recursos con otras personas, lo que obliga a guardar resultados parciales del entrenamiento de las redes neuronales implementadas para retomarse en una nueva sesión. Se ha utilizado un pequeño script para conocer si todos los recursos de las máquinas virtuales de Google (GCC) son en exclusiva, y en caso contrario, solicitar un nuevo entorno hasta conseguir exclusividad. Además, en lugar de leer las imágenes directamente desde Google Drive en el entrenamiento de la red neuronal para clasificador de golpes, se copiaban en un archivo comprimido (.Zip) a la máquina virtual para hacer el entrenamiento más rápido.

Como se menciona en el capítulo 3, se sigue un ciclo iterativo en el desarrollo, aplicando la metodología Scrum, en el que cada etapa refina y avanza lo conseguido en la anterior. Lo que

permite, desde el inicio del TFM, detectar las dificultades para la consecución de cada objetivo, permitiendo valorar acciones correctivas al plan inicial en caso de ser necesario.

Los siguientes apartados exponen el trabajo realizado separado por cada uno de los objetivos específicos. Además, dentro de cada objetivo, se indican los sucesivos acercamientos a la solución o iteraciones (sprints).

4.2.1 Detección del Jugador

Esta fase corresponde al primer objetivo específico planteado. Se elige el objetivo de la detección del jugador más cercano a la cámara como el primero a implementar, dado que a priori es más asequible que el resto de objetivos. Este consiste en,conseguir la posición del jugador, a través de una secuencia de imágenes de un peloteo de tenis. La cual extrapolándola con la relación pixel metro de la imagen, según la resolución de esta, permite calcular los metros recorridos por el jugador.

4.2.1.1 Sprint 1: Detección de Movimientos

El paso inicial es encontrar una manera de pasar los vídeos a imágenes para poder tratarlas individualmente. Este primer paso se consigue utilizando la librería pims, que con una sola instrucción convierte el video, pasado como parámetro, en un array de imágenes:

pims.Video(video_path)

El código base del que se parte es el del ejemplo comentado en el apartado 2.4.3.1 de detección de movimientos. El procedimiento que se realiza para cada imagen se divide en dos partes: primero se procesa la imagen para conseguir una imagen en blanco y negro con los movimientos detectados, y en una segunda encontrar los contornos de las partes con movimiento y mostrarse junto con la imagen original.

El Procesado de la imagen se realiza siguiendo las siguientes actividades:

1. Se reduce el tamaño de la imagen desde el tamaño original de 720x1280 píxeles a un ancho de 500x281 píxeles. Esto se hace para tratar con imágenes más pequeñas, con lo que es más eficiente el procesarlas, pero además al comprimir los píxeles se eliminan pequeños detalles de la imagen que facilita la detección en los pasos posteriores:
   frame = imutils.resize(images[i].copy(), width=500)

2. Se convierte la imagen a escala de grises, dado que no importan los colores (un jugador podría tener camisetas de distintos colores), además para procesos posteriores requiere este formato de imagen:

   gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

   La imagen se “emborrona” utilizar la función gaussiana para eliminar más detalles de las misma:
   gray = cv2.GaussianBlur(gray, (21, 21), 0)

3. Los tres pasos anteriores se realizan en la imagen en la que se quiere detectar el movimiento, y en otra anterior que se utiliza para obtener que ha cambiado entre ambas. Este cambio en las imágenes indica que se ha movido, que es justo lo que buscado. La imagen anterior se escoge por parámetro delta que indica cuántas frames hacia atrás seleccionamos. Esto es importante, puesto que de elegir justo la anterior en muchas ocasiones no cambia prácticamente nada y no sirve. Al contrario, si se escoge una imagen muy lejana en el tiempo, los cambios son tantos que no es fácil localizar al jugador con fiabilidad. Inicialmente se marca el valor de delta en 3.
   Se obtiene la diferencia de la imagen actual y delta, para obtener otra imagen que solo contenga lo que se ha movido entre ambas. Primero se calcula la una media de los valores en cada punto en las imágenes y posteriormente la diferencia de imágenes:
   cv2.accumulateWeighted(current_image, previus_image, 0.5)
   frameDelta = cv2.absdiff(current_image, cv2.convertScaleAbs(previus_image))

4. La imagen *frameDelta* obtenida está en escala de grises y contiene puntos de diferencias de los que necesitamos, por lo que se realiza la operación de threshold para que quede solo con valores de blanco total o negro total, eliminando los intermedios:
   thresh = cv2.threshold(frameDelta, 5, 255,cv2.THRESH_BINARY)[1]

   Después el operador morfológico de dilatación “unirá” los puntos cercanos. Esto sirve para áreas que se han detectado, y que posiblemente pertenecen al mismo objeto se unan formado un área mayor y más parecida a la forma real del sujeto, aunque esta operación agranda el objeto segmentado:
   thresh = cv2.dilate(thresh, None, iterations=2)

La secuencia de los pasos significativos son los siguientes. La imagen 5 muestran dos capturas originales después del resize. La captura de la izquierda es en la imagen que se usa como delta, y la de la derecha es la actual sobre la que se calculan las diferencias:

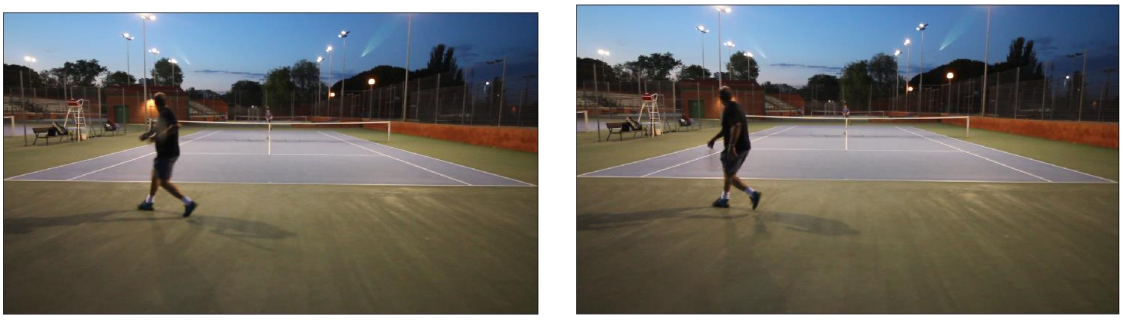

Imagen 5. Detección Jugador Actual y Delta

En la imagen 6 se muestran las mismas capturas de la imagen 5 justo después de hacer la operación blur a ambas imágenes. Se puede apreciar que se ha emborronado la imagen, eliminando detalles que podrían dificultarnos (pasos 1,2,3,4):

Imagen 6. Detección Jugador Blur Delta y Actual

Ahora se puede apreciar en la imagen 7, de izquierda a derecha, la diferencia de imágenes (paso 5), la binarización de la imagen (paso 6) y la operación de dilatación (paso 7), que se utiliza para unir contornos cercanos. Si se amplía la imagen, se puede apreciar que ha unido todos menos un pequeño contorno que pertenecerá a los pies del jugador.

Imagen 7. Detección Jugador Diferencia Imágenes, Threshold y Dilatación Threshold

Ahora se explican los pasos de procesar movimientos y mostrar resultado:

1. Lo primero es encontrar los contornos en la imagen obtenida en el paso 7. Esto consiste en ir recorriendo la imagen y en los puntos donde hay píxeles blancos, se busca si también son los colindantes, consiguiendo así el contorno del objeto:

   ```
   cnts=cv2.findContours(img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
   cnts = imutils.grab_contours(cnts)
   ```

2. Se obtienen normalmente más objetos de los deseados, por lo que se elimina los que tengan un área menor a un parámetro, que en este TFM se fija en 500:

   ```
   for c in cnts:
     if cv2.contourArea(c) < 500:
       # lo descartamos
   ```

3. Para cada contorno, se calcula el menor rectángulo que lo contiene y se pinta en la imagen original, para encuadrar en verde los movimientos detectados:

   ```
   (x, y, w, h) = cv2.boundingRect(player_contour)
   cv2.rectangle(image_color_original, (x,y),(x+w,y+h), (0, 255, 0), 2)
   ```

En este punto se consigue detectar los movimientos en la imagen, pero según las diferentes imágenes, también se consiguen sombras, sobre todo en la noche por los focos, la pelota, el jugador contrario, por lo que se requiere quedarse solo con el jugador que interesa (el que está en primer plano). Los resultados del primer sprint se ven en la imagen 8. En la captura de la izquierda se obtiene un único contorno (meta deseable). Otros casos, como en la captura de la derecha, se aprecian múltiples contornos que habrá que eliminar:

Imagen 8. Detección Jugador Resultado Sprint 1

4.2.1.2 Sprint 2: Selección del contorno del jugador

Esta segunda iteración busca seleccionar el contorno de los obtenidos anteriormente que más se adapte al jugador. Se estudian las imágenes para ver de qué manera obtener la mejor segmentación del jugador, o al menos que sea válida para obtener la posición del jugador. Como el jugador está en primer plano, el contorno más grande debería ser este, pero sobre todo con las sombras, a veces no se obtiene el resultado deseado.

Partiendo del código anterior, básicamente se prueban diferentes parámetros de tamaño mínimo del área de cada contorno, así como diferentes valores delta. Se prueban valores entre 500 y 1200, buscando el equilibrio que elimine contornos no deseados, los más pequeños, pero que a su vez siempre coja al jugador. También se prueban diferentes valores delta, desde 1 a 5.

Después de muchas pruebas se llega a la conclusión de que los de un área mínima de 700, un delta de 3 y seleccionando el contorno mayor de los resultantes, segmenta adecuadamente al jugador cuando se encuentra en el fondo (cerca de la línea de fondo), y se pierde según se acerca a la red. Si se baja el parámetro de mínimo de área sí se detectan cuando el jugador se aleja de la cámara, se obtenían falsos positivos. Por este motivo, se centra el reconocimiento del jugador cuando se encuentra jugando de fondo, que, aunque es una limitación, es donde se desarrollan la mayor parte de los puntos, y más en el tenis actual.

El movimiento del jugador se detecta de una manera más fina, en la que solo se obtiene un contorno, y funciona correctamente por la noche, donde las sombras es un factor que complica la tarea de fijarse en el jugador. Como puede apreciarse en la imagen 9, en algunos casos solo se recoge parte del jugador, dado que en ese momento el tenista está estático. Estos casos se tratarán en el siguiente paso para conseguir la posición correcta del jugador.

Imagen 9. Detección Jugador Resultado Sprint 2

4.2.1.3 Sprint 3: Obtención la posición del jugador e información

Una vez que se tiene el contorno del jugador de manera fiable, se busca ir obteniendo la posición (x,y) de la posición del jugador en cada momento. Del rectángulo que contiene al jugador, interesa la mitad de la arista más cercana a la cámara porque es donde aproximadamente se encuentra los pies del tenista.

Un primer paso es obtener la (x,y) de cada frame y probar como de fiable es para posicionar al jugador. Desde un primer momento se ve claramente que hay que ir almacenando posiciones de varios frames consecutivos y calcular la media de ellas, porque la variabilidad entre una y la siguiente es demasiado grande. Por ejemplo, cuando un jugador está golpeando la pelota, la mayor parte del cuerpo no se detecta. Sin embargo, sí se detecta el brazo y la raqueta. Después de ciclos de prueba/error se marca en almacenar las últimas 5 posiciones, lo que ofrece unos buenos resultados en el eje x (horizontalmente). La variabilidad en la coordenada hace no utilizarla para procesar información, puesto que no es suficientemente estable.

Se observa en todos los vídeos que se han grabado para las pruebas, que el enfoque es prácticamente calcado en todas, y que sabiendo la posición x del jugador se puede estimar cuantos metros se desplaza el jugador. La posición del jugador siempre varía un poco entre frame y frame, pero muchas veces es más por el proceso de detección que porque realmente el jugador se esté desplazando.

Se establece que no interesan los movimientos que no sean constantes, es decir, si las posiciones de frames consecutivas indican que se mueve como en el siguiente ejemplo, las descartamos: derecha, izquierda, izquierda, derecha.

Un pequeño algoritmo va almacenando la dirección del movimiento del jugador, y cuando en x frames consecutivas mantenga la dirección, entonces, se calcula la diferencia en píxeles de la frames actual con la x frame anterior.

La conversión de píxel a metros es sencilla; el fondo de la pista, por donde se suele mover el jugador es de casi 12 metros, y ésta, por el enfoque realizado, cubre prácticamente todo el ancho de la imagen. Se divide el ancho de la imagen entre los 12 metros y para conseguir el parámetro de conversión.

Ahora resta, que cuando se detecte un movimiento en píxeles se divida por la constante de conversión a metros y ese valor acumulado es la distancia total recorrida por el jugador. A continuación, un ejemplo en la imagen 10, aunque en el apartado de resultados hay una secuencia completa.

Imagen 10. Detección Jugador Resultado Final

4.2.2 Referenciar medidas de la pista de tenis

Esta fase corresponde al segundo objetivo específico planteado. Una vez obtenido la posición (x,y) del jugador, se busca referencias las líneas de las pistas. Este proceso ha resultado bastante complejo dado el enfoque de la cámara y buscando una generalidad del proceso para que funcione con diferentes condiciones de luz y tipos de pista.

El primer paso es detectar bordes en las imágenes (realmente cambios bruscos en el color) según una figura geométrica. Se utiliza el algoritmo de Canny que perfila muy bien objetos, si obtener un número de perfiles excesivo. Esta perfilación no es suficiente, por lo que hay que ayudarse segmentando las líneas por su color, blanco en todas las pistas independientemente del tipo de superficie (cemento, hierba, tierra, sintética).

Una vez obtenida una imagen en la que contenga idealmente las líneas, se aplica la trasformada de Hough, que detecta las coordenadas de elementos lineales en las imágenes. Por cada vídeo solo será necesario aplicar este cálculo, pues al ser la cámara fija, no es necesario repetirlo pues tendrá un resultado muy similar.

A continuación, se divide el proceso en sprints, en los que se refina progresivamente.

4.2.2.1 Sprint 1: Combinación de Canny, segmentación color y Transformada de Hough

Partiendo de una imagen cualquiera del video, da igual que esté el jugador o no, se van montando los pasos necesarios para detectar las líneas, e idealmente la red de la imagen, descartando el resto de elementos lineales de la imagen. Esta primera iteración tiene como objetivo el probar los diferentes métodos para comprobar si son útiles para la segmentación de las líneas. Durante esta fase solo se emplea una imagen de día, que como se verá después en la segmentación por color, afecta mucho el tipo de luz y las sombras para Canny.

Para la prueba 1:

1. Se crea una máscara de color para intentar que queden solo con las partes blancas de la imagen; por sí misma no es suficiente, puesto que tonos como grises a azules alto de tono también pueden aparecer en el filtro, pero sirve para descartar gran parte de la imagen:

   ```
   hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
   low_white = np.array([0, 0, 140])
   up_white = np.array([40, 77, 200])
   mask = cv2.inRange(hsv, low_white, up_white)
   ```

   En la imagen 11, se puede apreciar la imagen original y la máscara de color resultante. Se puede ver que las líneas las captura bien, pero también muchas partes de la imagen no deseadas, como el edificio anexo o las nubes:

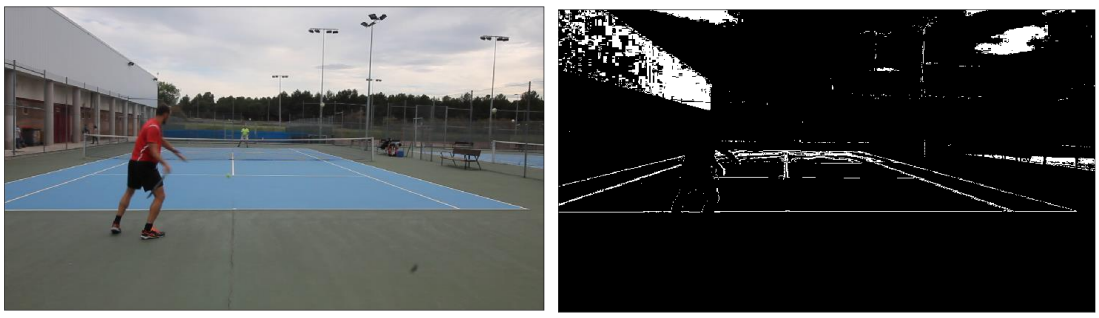

Imagen 11. Detección Líneas, Imagen Real y Máscara HSV

2. Seguidamente, se convierte la imagen a grises y se crea una versión dilatada a la que se resta la de grises; posteriormente se crea otra máscara Threshold que binariza la imagen:

   gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
   dilate_minus_gray = cv2.dilate(gray, disk(3), iterations=1) - gray
   thresh = cv2.threshold(dilate_minus_gray, 5, 255,cv2.THRESH_BINARY)[1]

   En la imagen 12, la captura de la izquierda es la resultante de la operación de diferencia de la imagen dilatada en gris menos la imagen en gris. A la derecha se binariza la anterior:

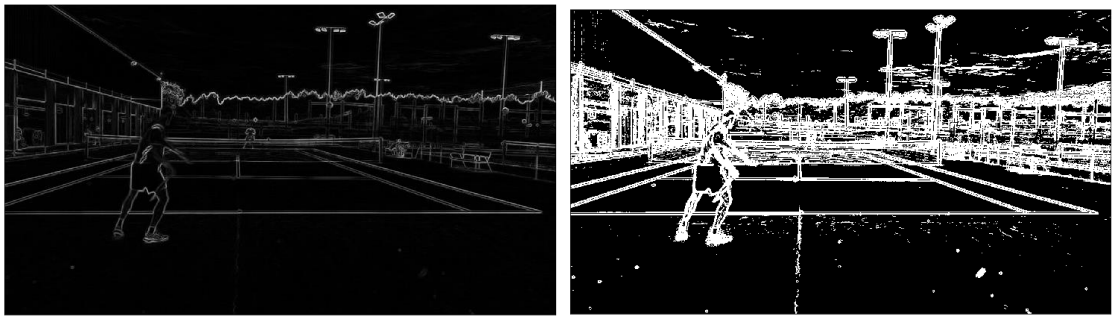

Imagen 12. Detección Líneas, Izq. diferencia dilatada y grises; Dcha. dilatado anterior

3. Finalmente se combinan las máscaras anteriores y se pasa el filtro Canny, la que se procesa con la versión probabilística de la transformada de Hough, para obtener las partes de los filtros:

   and_mask = cv2.bitwise_and(thresh, thresh, mask=mask)
   edges = cv2.Canny(and_mask, 75, 150)
   lines = cv2.HoughLinesP(edges, 1, np.pi/180, 50, maxLineGap=150)

Este paso arroja una lista con las líneas encontradas, que se pintan en la imagen en color origen; como se puede apreciar en la imagen 13, se encuentran muchas de las líneas de la pista, pero también muchas otras, demasiadas, que no interesan.

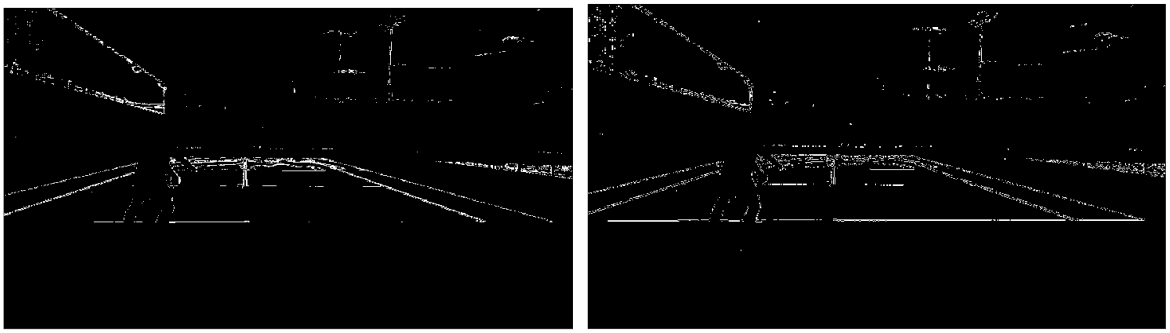

Imagen 13. Detección Líneas, Izq. cruce máscaras color y dilatada; Dcha. Canny anterior

Así, el resultado de las líneas detectadas sobre la imagen original se ve en la imagen 14:

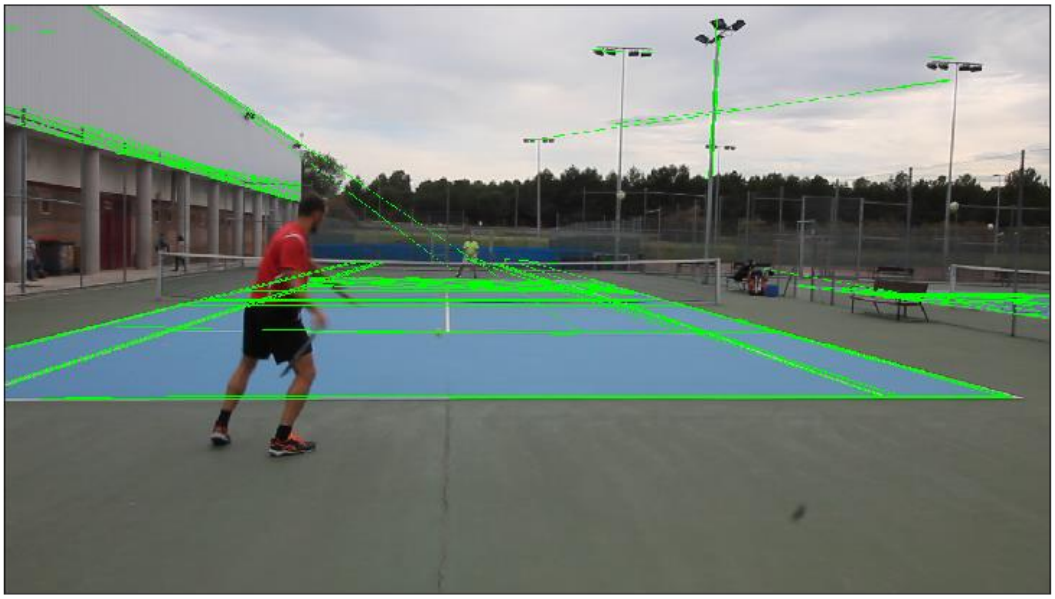

Imagen 14. Detección Líneas, Resultado Prueba 1 Sprint1

Para la prueba 2:

1. Aquí se eliminan los filtros de colores para independizar los colores y luz de la imagen. Se convierte la imagen a grises, y se prueba pasarle un filtro gaussiano para eliminar detalles, pero que preserva los bordes:

   gray_filtered = cv2.bilateralFilter(gray, 7, 50, 50)

2. A este filtro se aplica Canny y Hough y los resultados obtenidos aparecen muchas menos líneas, aunque también se pierden algunas. También se prueba un método

implementado en OpenCV de CornerHarris, que busca esquinas en las imágenes, pero con poco éxito. Probablemente la imagen que le paso a este método no es la más adecuada para que funcione correctamente. La imagen 15 además de las líneas en verde, en azul se pintan las esquinas de Harrys, que aparecen en lugares que no son de interés.

Imagen 15. Detección Líneas, Resultado Prueba 2 Sprint1

Después de estas dos pruebas, se aprecian las líneas de la pista, pero es necesario alguna acción adicional para eliminar las líneas sobrantes.

Además de las pruebas detalladas, se probaron otras dos técnicas, pero no se consiguió mejorar el proceso:

- Entropía de la imagen: Es una medida de cuan predecible el siguiente valor de una función dado el anterior. Aplicado a imágenes, mide cuanto se parecen los píxeles adyacentes, y por tanto, es más predecible su valor. El estudio de la entropía puede usarse para estudiar donde hay cambios en la imagen, es decir, bordes, un jugador, la pelota. Se hicieron pruebas, pero no aportan mejora en el reconocimiento de las líneas, por lo que se desestimó su uso.
- Saliency zones o zonas de relevancia: Estas zonas son las que se cree que el ojo humano se fija antes, como esquinas, bordes, cambios de luz, etc… Igual que con la entropía, no se consigue mejorar el procedimiento y se descara igualmente.

4.2.2.2 Sprint 2: Locura de filtros de color y combinación de máscaras

El objetivo de este sprint es de filtrar las líneas que no pertenecen a la pista y probar con imágenes distintas. El primer paso para simplificar el problema es eliminar zonas de la imagen que no pueden contener líneas; al saber la posición de la cámara, centrada en el fondo de la pista hacen algunas asunciones que facilitan la tarea.

Esta iteración investiga distintos rangos de colores, diferentes espacios de colores, que efectivamente restringen el número de líneas detectadas, pero siguen sin ser perfectos, puestos que para reducir las líneas reconocidas se pierden algunas de la pista. Aquí no se puede explicar un proceso exacto, porque es una sucesión de probar distintas combinaciones y cruces de máscaras, que no consiguen un buen resultado global para las distintas imágenes.

Se prueban filtrar tanto en el espacio de colores RGB como en HSL, el cual funciona muy bien en algunas imágenes. Se cruzan el filtrado de color en ambos espacios y se consigue mejor resultado que solo con uno, y sobre todo, es más genérico.

Analizando cuidadosamente porque no se consigue generalizar el procedimiento, utilizando un color picker (software usado para conocer el color del píxel de una imagen), se aprecia cómo incluso en una misma imagen el color blanco de la línea cambia sustancialmente a lo largo de la imagen, a tonos grises, incluso, hasta parece azulado o verde. Parte de este problema probablemente viene dado por la interpolación de pixeles que realiza la cámara, pero también por sombras o incidencia de la luz, especialmente por los focos incandescentes en la noche. Este proceso, bastante frustrante, deja claro la necesidad de utilizar un enfoque distinto puesto que después de varios días de pruebas no se consigue una mejora sustancial. En la imagen 16, se muestra un ejemplo donde la máscara RGB + canny funciona bien en imágenes de día, pero casi no detecta líneas en la derecha con luz artificial.

Imagen 16. Detección Líneas, Resultado Sprint2

4.2.2.3 Sprint 3: K-Means

El primer paso para simplificar el problema, es eliminar zonas de la imagen que no pueden contener líneas; al saber la posición de la cámara, centrada en el fondo de la pista hacen algunas asunciones que facilitan la tarea.

La zona de exclusión de la imagen donde no están las líneas de la pista, en lugar de venir marcada por una y arbitraría, se crea un polígono trapezoide que se adapta más ajustado a la pista y facilita eliminar parte de esas líneas indeseadas. Los vértices del polígono son:

{(0, 3/4 * alto), (0, ½ alto), (1/3 * ancho, corte), (2/3 * ancho, corte), (ancho, ½ alto), (ancho, 1/3 * ancho)

Según la posición y enfoque de la cámara se puede eliminar todas las posiciones con valor de la y entre los píxeles 0 y 250, puesto que corresponde al suelo (Nota: la posición (0,0) en las imágenes, se encuentra en la esquina superior izquierda).

También, visto que dificultad de extraer solo las líneas según los diferentes entornos de luz y pistas, se prueba el algoritmo k-meas de aprendizaje no supervisado, para intentar agrupar los píxeles por sus colores. Este algoritmo recibe como parámetro el número de grupos que quieres obtener, y en sucesivas iteraciones va colocando cada uno de los píxeles en el grupo que más se parezca. El resultado es que el color de la pista, dominante en las imágenes utilizadas, tienda a aparecer en el mismo grupo, aunque los colores menos dominantes, se agrupan en otros que igual no son los más significativos.

Después de varias pruebas con diferente número de grupos en k-means, el mejor resultado se obtiene aplicando dos veces sucesivas el algoritmo. La primera vez se agrupa a la imagen en color original en diez clústeres. El resultado de esta se ordena de mayor a menor por número de píxeles que contiene, y se eliminan de la imagen los 5 con más elementos. Estos coinciden con lo que se conoce como fondo de imagen, el cielo, los colores de la pista. Al resultado de reducir los primeros cinco grupos, se le vuelve aplicar k-means y se vuelven a eliminar los cinco grupos más números. De esta manera, se consigue separar, razonablemente el blanco de los colores de la pista, en la mayor parte de la imagen, y aunque queda algo de ruido, suele ser la camiseta del jugador, o partes sueltas de la imagen, que al juntar con el filtro Canny da un resultado razonable.

El filtro k-means más Canny, además de eliminar las zonas altas de la imagen, ofrece un resultado más general que otras opciones al no depender de las máscaras de color. aunque siguen apareciendo algunas líneas no deseables. Algunas como grietas en la pista, aparecen insistentemente a pesar de la variedad de filtros aplicados. La imagen 17 muestra dos capturas del proceso completo.

Primero se ve la imagen original de la que se ha eliminado las zonas de la imagen que no interesan. Después, se convierte a grises y se aplica el filtro Canny para detectar los bordes.

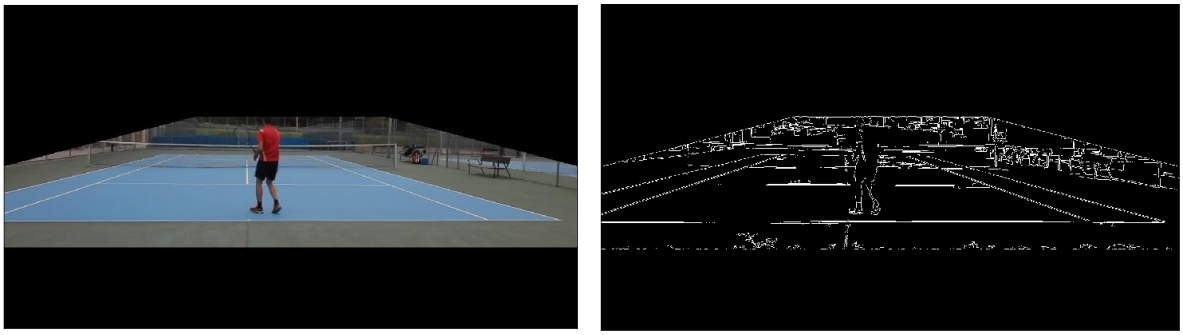

Imagen 17. Detección Líneas, Original truncada zona de interés y Canny anterior

En la imagen 18, se ve en la captura de la izquierda, la resultante de aplicar dos veces el algoritmo de k-means. La captura de la derecha, es la unión de las máscaras de Canny y k-means, donde se puede apreciar una imagen mucho más limpia.

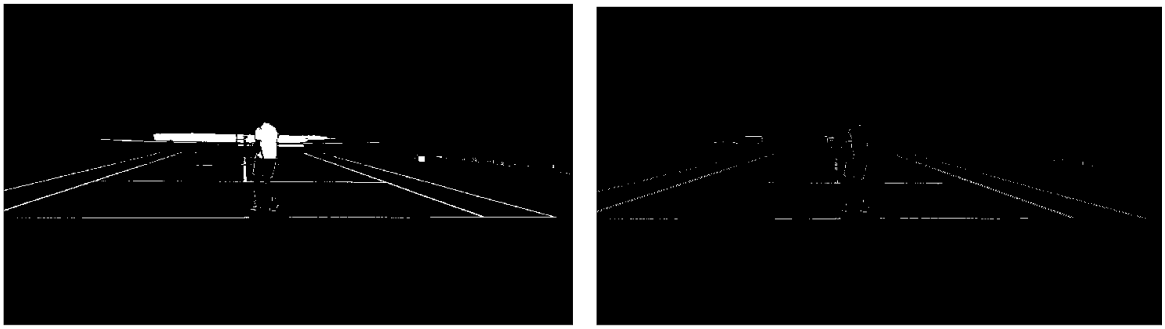

Imagen 18. Detección Líneas, Izq. k-mean original truncada, Dcha. Canny And k-mean

El resultado final sobre la imagen truncada se muestra en la imagen 19. Se detectan líneas laterales y horizontales de la parte de la pista más cercana. Se observa alguna desviación hacia el jugador y una línea en el fondo que detecta un plástico que hay en la valla del siguiente campo.

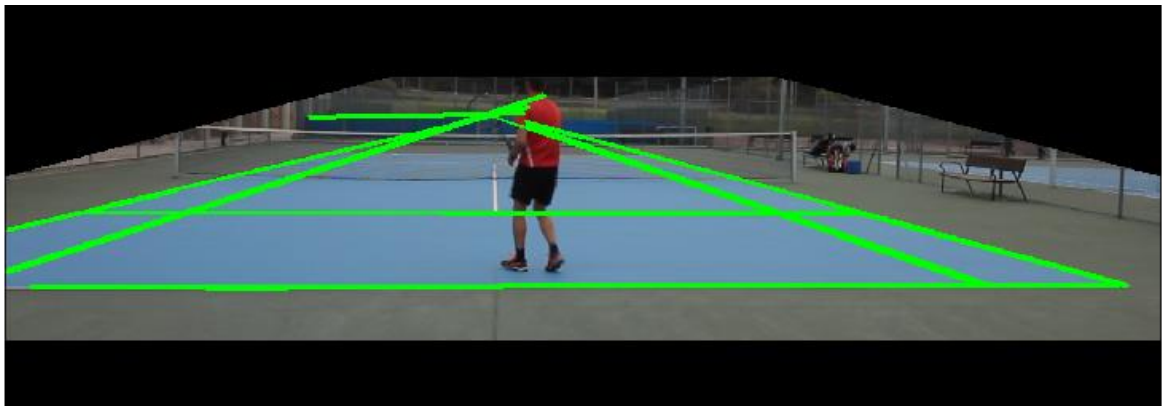

Imagen 19. Detección Líneas, Resultado Sprint 3

4.2.2.4 Sprint 4: Modelado de la pista

Visto que a pesar de tantas pruebas no se consigue replicabilidad de unos buenos resultados en todos los entornos, lo que se hace es, partiendo de las líneas que se consiguen en el sprint anterior, buscar las líneas de fondo, de saque y las laterales, que se identifican bien en todos los casos.

Partiendo de estas, se genera un modelo, porque al conocer la posición de la cámara, se sabe que la primera línea horizontal grande que haya será la línea de fondo, y la siguiente la de saque. En cuanto a las líneas laterales se identifican según la forma de la pista; empezando de izquierda a derecha, o viceversa, son línea de dobles, individuales, individuales y dobles nuevamente.

Previo al modelado, se realiza un tratamiento de las líneas, ya que algunas de ellas aparecen discontinuas, o varias líneas están superpuestas. Esta parte se utiliza trigonometría sencilla para ayudarnos en distintas tareas. Los pasos son los siguientes:

1. Ordenar las líneas en función de que sean horizontales, verticales, de izquierda y de derecha. Las líneas laterales, por efecto de la proyección del fondo, aparecen diagonales. La forma de clasificarlas es mediante el cálculo de la pendiente de la recta, y según su valor van cayendo en un saco u otro:

   ```
   def get_slope (line):
     return (l[3]-l[1])/(l[2]-l[0])
   ```

2. Seguidamente se obtienen de las líneas horizontales las que están más cerca de la cámara con un margen de 12 píxeles entre ellas. Estas líneas se agrupan y se calcula la mediana del valor y los valores máximos y mínimos de la x. La recta de estos valores, da la línea de fondo (baseline).

3. Se aplica el mismo algoritmo a las líneas horizontales restantes, y se obtiene la línea de saque.
4. Las líneas clasificadas como derecha e izquierda se suman y se aplica un k-means con 4 grupos, y produce una perfecta segmentación de las líneas de dobles e individuales de cada lado de la pista.
5. Para delimitar la y de las líneas laterales se cruza con la línea de fondo, para cortar las líneas y que dibujen exactamente la pista.
6. Lo mismo se realiza con la línea de saque, esta vez cruzando con las líneas laterales de individuales.
7. El resto de líneas horizontales, la línea imaginaria que separa los dos campos, y las líneas contrarias de saque y fondo, además de la línea vertical de saque, se generan a partir de las anteriores. El proceso es básicamente aplicar un factor de reducción de los metros que representa cada pixel según la profundidad. Partiendo de un valor, que depende de la posición de la cámara (entre 6 y 8 metros de la línea de fondo), el ángulo del enfoque y el tipo de lente. Este parámetro habrá que ajustarlo en cada caso, salvo que la colocación de la cámara sea idéntica en todos los casos, algo no demasiado realista. Conocer la distancia exacta de la línea de fondo con la de saque y de esta a la red ayuda en esta tarea. A continuación, el código de cálculo de la mitad de la pista, idéntico en el caso de las contrarias de saque y fondo:

   ```
   pixel_meter_baseline = (img.shape[0] - baseline[3]) / camera_distance
   pixel_meter_serveline = (baseline[3]-serveline[3]) / baseline_serviceline
   pixel_reduction = pixel_meter_serveline / pixel_meter_baseline
   baseline_single = baseline.copy()
   get_intersection_segment(baseline_single,side_lines[1],side_lines[2])

   slope = get_slope(serveline)
   current_pixel_meter = pixel_meter_serveline * (pixel_reduction * 1) # net
   y_aux = serveline[1] - int((current_pixel_meter * serviceline_net))
   y_aux2 = int(get_y(serveline[0],y_aux,serveline[2],slope))

   netline = [serveline[0],y_aux,serveline[2],y_aux2]
   get_intersection_segment(netline,side_lines[1],side_lines[2])
   ```

8. Las líneas laterales se intersectan con la línea de fondo contraria, para ajustarlas, ya ajustándose a las líneas reales.
9. Solo falta la línea vertical de saque, que se obtiene teniendo en cuenta la relación de las otras líneas. (Nota: si la cámara no está centrada o un poco ladeada, esta línea aparece un poco desplazada de su sitio real):

   ```
   vertical_serveline = [(netline[0] + netline[2]) // 2,
                         serveline[1],(netline[0] + netline[2]) // 2, serveline2[1]]
   ```

El conjunto de líneas obtenido se pintan sobre la imagen 20, quedando un modelo muy ajustado a la realidad.

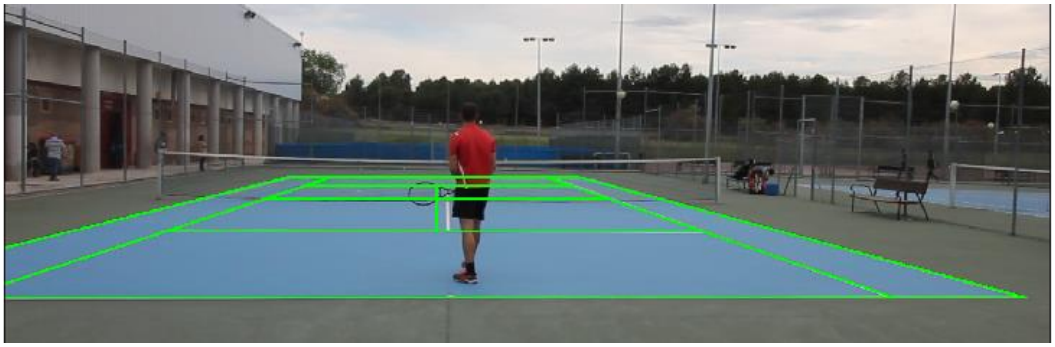

Imagen 20. Detección Líneas, Resultado Final

4.2.3 Seguir la pelota

Esta fase corresponde al tercer objetivo específico planteado. Una vez conseguido localizar al jugador y las líneas de la pista, lo que falta para poder empezar a extraer más información relevante. El código inicial utilizado es el de detección de movimiento del jugador, el cual en ese caso funciona correctamente, pero luego se ve la necesidad de modificarlo. La velocidad de la pelota es muy superior a la del jugador, solo comparable al movimiento de la raqueta durante un golpe. Esta alta velocidad, supera en ocasiones los 200 kilómetros por hora en profesionales y semiprofesionales, lo que genera que en muchas ocasiones la bola deje una traza muy alargada, o aparezca como dos objetos distintos en un frame. Esto se produce, además de por la velocidad, porque el número de frames por segundo de la cámara utilizada no es suficiente para grabaciones de este tipo y se necesitaría una cámara de las llamadas superlentas. El enfoque de la cámara también provoca que la pelota cuando se va alejando hacia el campo contrario aparece tan pequeña que incluso mirado detenidamente el frame en ocasiones cuesta encontrarla. Otro efecto encontrado es que el color de la pelota, amarillo chillón de las bolas nuevas, no se mantiene con bolas viejas como las usadas en los videos (con una duración previa de una hora). Además, según se aleja, y en función de las situaciones de luz, el amarillo se torna gris y se puede confundir con una sombra de un pájaro lejano o una mota de polvo en el objetivo.

Estas dificultades añadidas han forzado a utilizar una técnica ligeramente distinta a la empleada en la detección del jugador. Técnica basada en el artículo (Rozumnyi, Kotera, Sroubek, Novotny, & Matas, 2016) y una sencilla implementación de este artículo publicada en la web (Massenzio, 2017) se prueba y funciona mejor para este objetivo que la anterior.

4.2.3.1 Sprint1: Continuidad a la detección del jugador

El punto de partida es el sprint 3 de objetivo 1. El caso del jugador era más fácil encontrar al jugador puesto que es el contorno más grande encontrado, pero eso para la pelota no es suficiente, por lo que a este código le realizan ajustes y uso de otras técnicas.

Lo primero es introducir, como en los sprints iniciales de la detección de líneas, una máscara de color, amarilla obviamente en este caso, para filtrar junto con la máscara generada por la diferencia de imágenes con delta 3, donde los valores en HSV del rango de amarillos que se prueban son varios, según si son imágenes de día o de noche:

```
# Valores para por la noche
# yellowLower = (50, 10, 175)
# yellowUpper = (100, 25, 190)

# Rango amplio por la noche
# yellowLower = (20, 0, 40)
# yellowUpper = (100, 100, 200)

# buenos de día
# yellowLower = (50, 60, 60)
# yellowUpper = (64, 255, 255)
```

Esta máscara ayuda en algunos casos, aunque no será suficiente, porque el amarillo de la camiseta de uno de los jugadores también se detecta, pero también el movimiento de hojas de los árboles, e incluso partes de la pista de tenis. Estos últimos se debe a que la parte exterior de la cancha es verde, y según la luz se acerca al rango de los amarillos ampliada utilizados. Los puntos de la pista no deberían aparecer en la máscara de movimiento, pero cambios en la luz, y sobre todo cuando el jugador se va moviendo, los píxeles en los que estaba el jugador en la frame anterior sí que aparece como que ha habido movimiento.

Otro cambio que se realiza al código del objetivo 1, es eliminar los suavizados de imagen, dado que en muchas ocasiones termina por hacer invisible la pelota, y es lo contrario a lo que a conseguir.

También se realiza un filtrado del tamaño de los objetos encontrados, pero esta vez en lugar de escoger el mayor, de deja “pasar” a todos los que no sean demasiado pequeños ni más grandes del tamaño de una pelota. Esta no tiene un tamaño mayor a 30 píxeles en las imágenes en las que aparece muy alargada.

Las máscaras que se fueron generando se presentan en la imagen 21. En la captura de la izquierda se ve la imagen original, mientras que en la derecha se ve el resultado de aplicar la máscara de color amarillo:

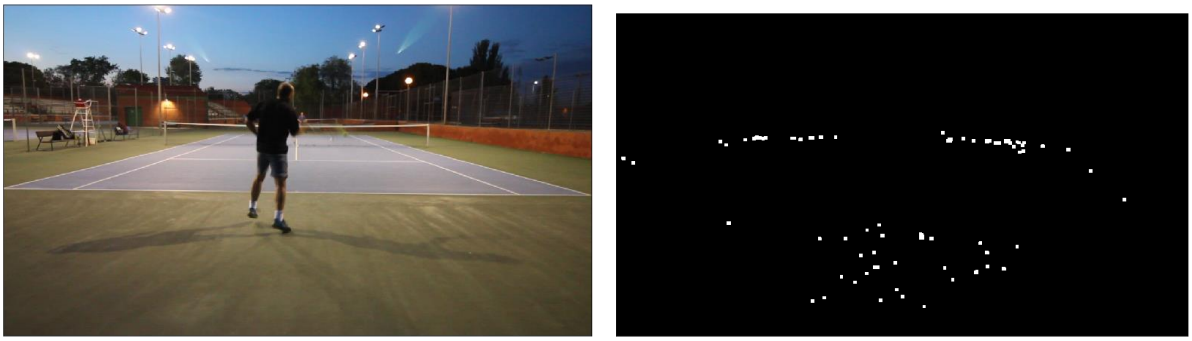

Imagen 21. Detección Pelota, Izq. Original; Dcha. Máscara Color

Ahora, la imagen 22, muestra el resultado de la máscara de detección de movimientos, siendo la captura de la derecha la versión binarizada de la captura de la izquierda.

Imagen 22. Detección Pelota, Izq. diferencia con delta; Dcha.: binarización anterior

En la imagen 23, en la captura de la izquierda se ve la dilatación de la máscara anterior binarizada. Esta operación de aumentar la detección de objetos (en blanco) se realiza para que al juntarlo con la máscara de color detecte en más ocasiones la pelota, puesto que la máscara de color suelen ser pequeños puntos. El resultado de unir la máscara de color y la dilatada de movimientos se ve en la captura de la derecha.

Imagen 23. Detección Pelota, Dilatación Binarización; Dcha. Color And Binarización

El resultado de la detección de contornos sobre la imagen original se ve en la imagen 24. Se detecta la bola, aunque no exactamente en el punto real, pero también muchos falsos positivos en los pies del jugador. Estos se producen por los pies del jugador y las sombras, que al moverse marca como zona de movimiento zonas verdes de la pista, similar a la pelota.

Imagen 24. Detección Pelota, Resultado Sprint 1

4.2.3.2 Sprint 2: Eliminación de puntos no deseados

El objetivo ahora se fija en eliminar los puntos no deseados que aparecían en bastantes sitios de las imágenes.

Lo primero que se hace es volver a jugar con las máscaras HSV de amarillo y que sea común en diferentes circunstancias de luz, quedando finalmente los valores del rango en:

yellowLower = (50, 50, 100)
yellowUpper = (90, 100, 200)

En este punto surge el siguiente planteamiento: ¿los puntos amarillos que no se muevan del sitio, se pueden descartarlos como pelota? Incluso una pelota en una esquina de la pista no interesa. En este punto es va almacenando los contornos de las n frames anteriores (el valor final queda fijado de 48 frames o 2 segundos). Entonces, en cada frame, antes de dar por válido un contorno se mira en nuestra lista de entornos previos y se chequea dentro de un pequeño margen si está dentro del radio de estos. En el caso de que no haya ningún contorno cercano, se da por válido.

Este pequeño algoritmo consigue reducir muchos de los puntos falsos, aunque no lo hace completamente. Además, la máscara de color, aunque mejor que en el anterior sprint, funciona mejor en unas circunstancias que en otras. Aplicados estos cambios, la capacidad de seguir a la pelota es razona, aunque con muchos falsos positivos, lo que en la práctica hace imposible hacer un seguimiento efectivo de la pelota y generar información fiable. Las siguientes imágenes contienen varios ejemplos.

Este primer ejemplo, a la izquierda siempre la original, se ve a la derecha que se detecta como pelota una parte de la raqueta, por lo tanto es un caso de falso positivo:

Imagen 25. Este primer caso es un falso positivo (detecta la raqueta).

En este caso, a la derecha, se ha detectado correctamente la pelota, a pesar que es necesario ampliar la original para ver donde se encuentra la bola, que además está algo movida:

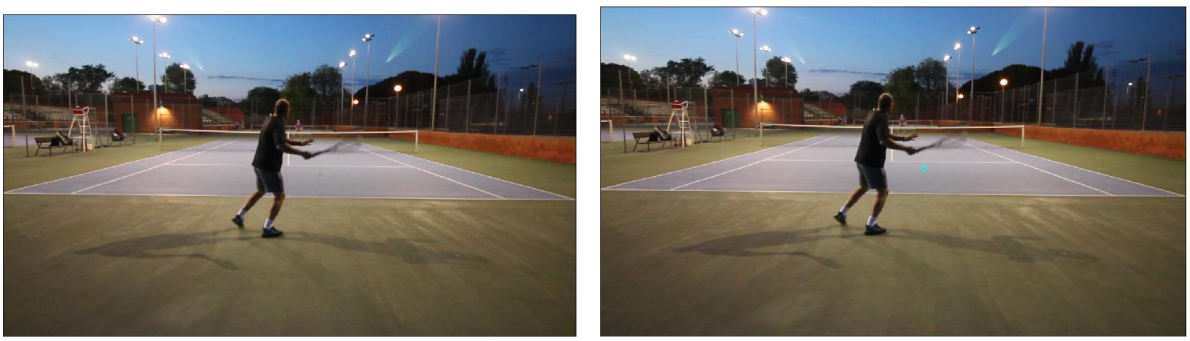

Imagen 26. Detección Pelota, A la derecha se ve lo que detecta la pelota correctamente.

En este tercer ejemplo, se detecta la pelota está a pesar de estar muy difuminada y parcialmente tapada por la raqueta (ampliar original bastante para apreciarlo), aunque también detecta un punto en el suelo. Por tanto, aquí se ha detectado tanto la pelota correcta como un falso positivo:

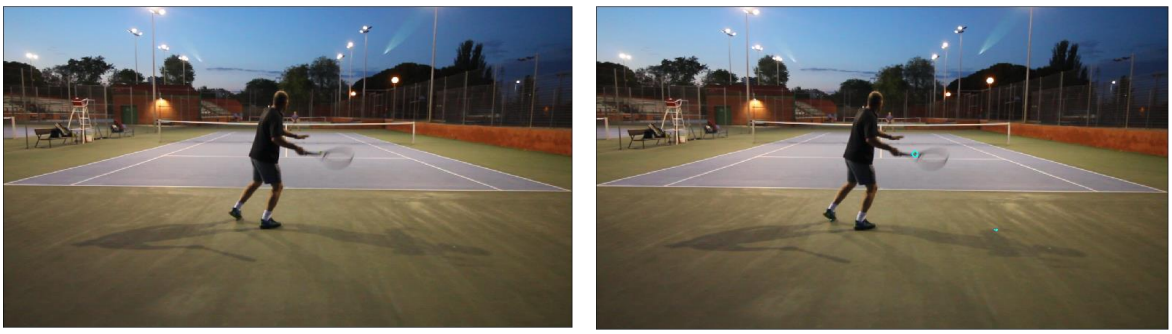

Imagen 27. Detección Pelota, Se detecta pelota y falso positivo

Usando la misma máscara y procedimiento, también funciona en imágenes en otro entorno y condiciones de día. En la imagen de la derecha se aprecia la pelota cuando se dirige hacia del jugador contrario:

Imagen 28. Detección Pelota, ejemplo luz de día detección correcta

4.2.3.3 Spring 3: Creación de trayectorias posibles

Visto que en el estado actual es imposible saber qué punto de los detectados es la pelota se debe idear otras estrategias para eliminar todos los falsos puntos. En este caso, no se puede eliminar una gran zona de la imagen como se hacía para las líneas, puesto que, en ciertos golpes como un globo, la pelota incluso se sale de enfoque por arriba. Solo se podrían quitar algunos puntos muy cercanos a la cámara, por debajo de la línea de fondo, pero es algo que por ahora no implementamos.

Lo primero es que buscando en varios foros de Internet aparecen un par de algoritmos para saber si un color RGB es amarillo o no. Y sorprendente, una de estas funciones cumple su función sorprendentemente bien:

```
def rgb2yellow(R,G,B):

  u=math.atan2( (R-G)/math.sqrt(2)  ,  (R+G-2 * B)/math.sqrt(6) )
  V=max(R,G,B)
  v=min(R,G,B)
  S= 2 *(V-v)/(1+ abs(V-0.5)+ abs(v-0.5))
  yelowness =  S* math.cos(u)

  return (yelowness + 4) / 6
```

Además, aplicando las técnicas del artículo sobre detección de FOM (Fast Object Movement) se cambia la manera de detectar los objetos en movimiento. Lo que ataca esta técnica es que la pelota se desplaza a una velocidad mayor que el resto de elementos en los vídeos. Aplicando esta nueva forma de procesar las diferencias entre imágenes ofrece mucho mejor resultados en el caso de la pelota. Este método en lugar de tener un delta de 3, lo que hace es crear 3 máscaras de diferencias:

1. Entre la imagen actual y la anterior (minus)
2. Entra la imagen actual y la posterior (plus)
3. Entre la imagen anterior y la posterior (delta)

Se calcula la desviación estándar de estas máscaras y se aplica a la binarización Threshold. Luego se combinan las máscaras minus y plus, para volver a combinarlas estaba con la máscara delta invertida. Este proceso arroja algunos puntos de un pixel, pero pocos contornos mayores, facilitando los siguientes pasos muchísimo.

Solucionado el problema de la máscara, que ya funciona muy bien, siguen apareciendo unos pocos falsos positivos. Aunque solo sean 2 o 3 hay que implementar alguna solución para saber cuál de ellos es la pelota. La solución que implementamos, aunque algo

complicada, es en lugar de guardar todos los puntos que se detecta para eliminar los que no se han desplazado del sitio, como sí hace la pelota, es guardar todos los contornos. En este momento los contornos detectados son muchos menos gracias a la nueva máscara, lo que se hace es almacenarlos durante unas pocas frames, en la versión final 6. Ahora sobre esto puntos lo que se hace es un pequeño algoritmo que recorre desde los contornos encontrados más antiguos en adelante, si en la siguiente frame hay un contorno cercado dad un margen y el radio del contorno encontrado. Si es así, se forma con ambos puntos una recta que se guarda en posibles trayectorias. Este paso se repite para los 6 niveles de antigüedad; al final lo que se tiene es una serie de puntos más o menos alineados, que marcan posibles trayectorias de la pelota. Se Seleccionan para mostrar por pantalla los que tengan al menos 3 puntos, lo que en la mayoría de los casos es la pelota, aunque todavía hay algún falso positivo con la raqueta y los pies del jugador.

Esta iteración fue muy fructífera, introduciendo una nueva máscara de color, de movimiento y el cálculo de posibles trayectorias. Al pintarlas sobre las imágenes, y sobre todo cuando se visionan en video, ofrece muy buen comportamiento ya que se pueden ver las curvas que realiza la bola en el aire, pudiendo apreciar la altura que alcanza la bola y el tipo de golpeo. Los golpes más planos tienen trayectorias más rectas que los liftados que suben a más altura para caer bruscamente, casi de manera vertical en los últimos momentos.

Aun así, no se consigue aislar suficientemente la trayectoria como para procesar los datos fehacientemente de la posición de la pelota en todas las frames. En este punto se podría considerar un nivel de cumplimento aceptable pero no suficiente para la extracción de información relevante, más allá que visualmente sí ofrece un buen resultado de seguimiento de la pelota.

La siguiente iteración de este objetivo se solapa con el siguiente objetivo de reconocimiento de golpes, por lo que se contará en dicho apartado. Las imágenes que se muestra son de una etapa son de los pasos previos al sprint 4 del objetivo 4, en donde añado la posición de la línea de fondo y de la red para ayudar a la detección de golpes. En cualquier caso, se puede ver en la secuencia como se sigue la trayectoria de la bola con bastante acierto. Las imágenes de la derecha están ampliadas para poder ver mejor la trayectoria.

En la imagen de la derecha, se ve la pelota sale del jugador contrario, por encima del hombro del jugador:

Imagen 29. Detección Pelota, Izq. Original; Dcha. Inicio trayectoria pelota

Como se ha detectado varios puntos cercanos, se pinta una trayectoria porque ya se da por seguro que el punto detectado es la pelota:

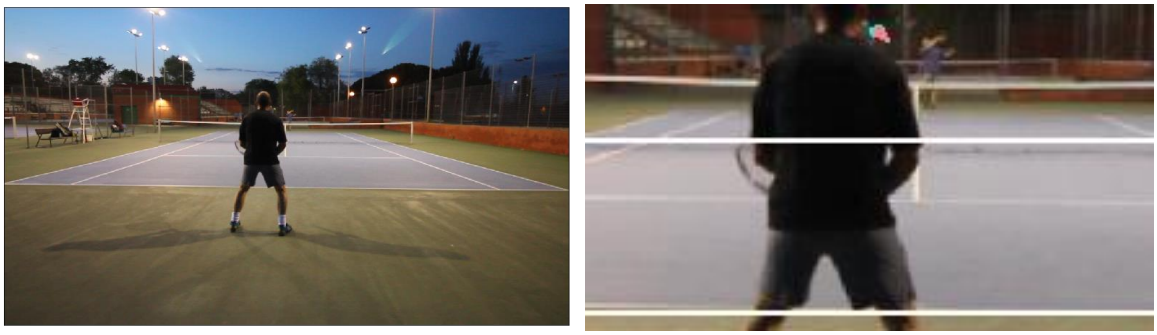

Imagen 30. Detección Pelota, Izq. Original; Dcha. continuación trayectoria pelota

La imagen 31, continua la trayectoria anterior, confirmado que lo detectado es la bola:

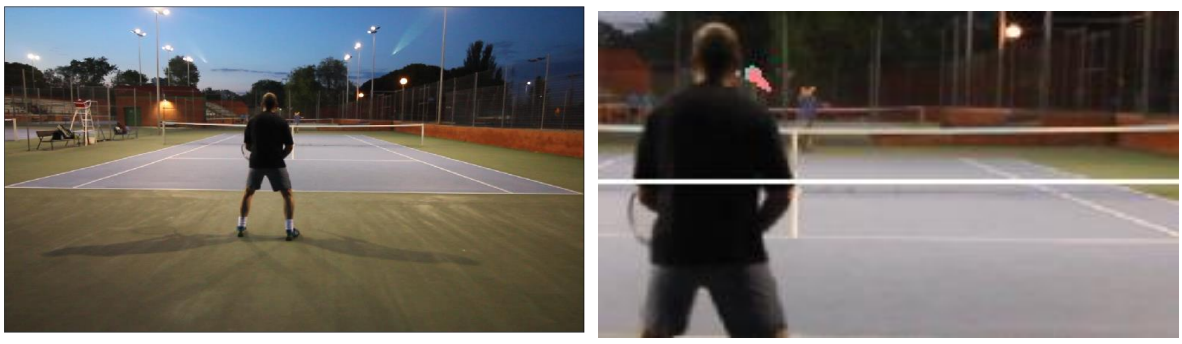

Imagen 31. Detección Pelota, Izq. Original; Dcha. continuación trayectoria pelota 2

Varias frames después, se sigue perfectamente a la pelota, pintando en este caso la traza de color azul:

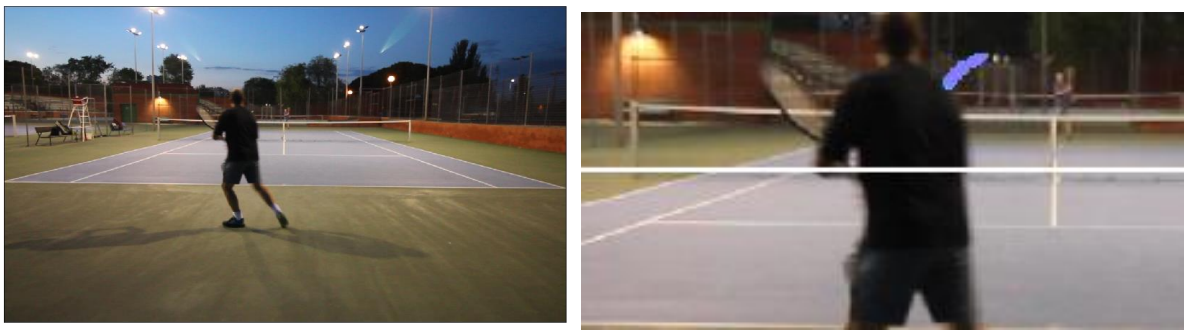

Imagen 32. Detección Pelota, Izq. Original; Dcha. continuación trayectoria pelota 3

El color de las trazas es rojo si la bola está ascendiendo o en azul si desciende, según disminuya la coordenada Y de los contornos detectados en la trayectoria. En este caso solo se pinta (se da por válida) una trayectoria, si tiene más de 2 puntos, para evitar falsos positivos, aunque, aun así, alguno persiste puntualmente.

4.2.4 Clasificar el tipo de golpe

Los objetivos anteriores se han abordado desde el tratamiento de imágenes digitales, utilizando una gran variedad de técnicas utilizadas en esa área. Sin embargo, para este target se quiere emplear las técnicas actuales de redes neuronales que en su versión de redes convolucionales han tenido mucho éxito en el reconocimiento de imágenes. Aquí hay tres grandes retos para conseguir alcanzar nuestra meta:

1. La creación de una base de datos con suficientes ejemplos para el entrenamiento de la red. Estas redes normalmente necesitan una gran cantidad de información para entrenar correctamente.
2. La dificultad de encontrar la configuración adecuada de la red. Existen ilimitadas configuraciones posibles, y dar con la más adecuada requiere conocimiento teórico sobre las redes y, sin duda, experiencia previa para dar con alguna configuración válida.
3. Los entrenamientos de estas redes requieren mucha capacidad de cómputo, y por tanto tiempo. Aunque se usa el entorno gratuito de Google (Colab), en algunos casos el entrenamiento tarda varias horas, lo que complica la prueba de cada configuración enormemente.

En los dos primeros sprints se han utilizado estas técnicas para el reconocimiento del tipo de golpe, pero en tercero se optó por un enfoque desde los procedimientos de visión artificial, ya que, aunque se consiguen cierto nivel de predicción en los test, en los casos reales caía a

aproximadamente la misma distribución de nuestros datos de ejemplos, lo cual indica que el modelo no entrena correctamente.

4.2.4.1 Sprint 1: Primer prototipo de red neuronal para la clasificación de golpes

El primer paso fue tratar adecuadamente los datos de los videos. Lo que se hizo en un primer momento, fue grabar unos pocos cientos de videos de 2/3 segundos con diferentes golpes: saques, derechas y reveses. Los jugadores disponibles para los vídeos solo fueron 2 y se grabó en un frontón, con la idea que la red solo se fijara en el jugador ya que el fondo era bastante homogéneo. Estos videos se clasifican en función del golpe y el tipo de efecto que se daba a la bola, liftado o cortado. También se incluye el revés a dos manos.

Todos esos videos se clasifican en distintas carpetas y se pasan a frames, formato con el que trabaja la red neuronal. El objetivo ideal era que la red incluso detecte el efecto que se ha dado a la bola. Desde el principio se intuye que puede haber un problema con los datos, ya que el entorno no es igual al real y algunas de las frames de los vídeos ni alguien que conoce de tenis sabría decir con certeza que golpe es. Aun así, se confía que el poder de reconocer patrones de las redes convolucionales sea suficiente para obtener un nivel de acierto razonable. Un porcentaje de acierto por encima del 80% si la tasa de errores no tiene mucho sesgo hacia un tipo de golpe puede ser suficiente para extraer información valiosa de los vídeos.

Además, conociendo que la necesidad de más imágenes para el entrenamiento se utiliza la técnica de data augmentation. Esta consiste en ir realizando pequeñas modificaciones en los datos de entrenamientos, que no cambian demasiado la imagen, pero si son suficientemente notables para multiplicar varias veces el tamaño de nuestro dataset. Se utiliza una clase de la librería de keras que se utiliza para las operaciones con la red neuronal, y con una sola línea, se le indica a la clase que tipo y con qué intensidad debe aplicar estas transformaciones a las imágenes. El siguiente código accede a las imágenes originales, y prepara los dataset de entrenamiento y validación con las imágenes transformadas, listas para ser utilizadas por nuestra red neuronal.

```
path = '/content/gdrive/My Drive/Universidad/Proyecto/videos/stroke_frames'

image_rescale = 0.35
image_width = round(500 * image_rescale)
image_height = round(281 * image_rescale)

datagen = ImageDataGenerator(rescale = 1./255 ,height_shift_range=0.2,
width_shift_range=0.4, horizontal_flip=False, validation_split=0.2,
```

```
rotation_range=30,brightness_range=[0.2,1.0],zoom_range=[0.8,2.0],preprocessing_function
= convertRGB2BGR)

train_generator = datagen.flow_from_directory(path, subset='training',
target_size=(image_height,image_width),class_mode='categorical',
  batch_size=batch_size)

val_generator =
datagen.flow_from_directory(path,subset='validation',target_size=(image_height,
image_width),class_mode='categorical', batch_size=batch_size)
```

Una vez realizada preparación del dataset, se implementa un primer modelo como los vistos durante las prácticas del máster. Después de varias arquitecturas distintas se ve que el problema no se va a resolver como inicialmente plantea de manera inocente. La definición de la red es la siguiente:

```
def cnn_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), padding = 'same', input_shape = (image_height, image_width,
3), activation='relu'))
  model.add(Conv2D(32, (3, 3), activation = 'relu'))
  model.add(MaxPooling2D(pool_size = (2, 2)))
  model.add(Dropout(0.2))

  model.add(Conv2D(64, (3, 3), padding = 'same', activation = 'relu'))
  model.add(Conv2D(64, (3, 3), activation = 'relu'))
  model.add(MaxPooling2D(pool_size = (2, 2)))
  model.add(Dropout(0.2))

  model.add(Conv2D(128, (3, 3), padding = 'same', activation = 'relu'))
  model.add(Conv2D(128, (3, 3), activation = 'relu'))
  model.add(MaxPooling2D(pool_size = (2, 2)))
  model.add(Dropout(0.2))

  model.add(Flatten())
  model.add(Dense(512, activation = 'relu'))
  model.add(Dropout(0.3))
  model.add(Dense(num_classes, activation = 'softmax'))

  return model
```

Se trata de una CNN no demasiado profunda comparada con los modelos actuales pero que sirve de piedra de toque para probar el procedimiento que se quiere instaurar para clasificar los golpes. Ahora solo queda entrenar la red y ver cómo se comporta:

```
model = cnn_model()
```

```
# Definición el optimizador SGD configurando momentum para evitar los pequeños mínimos locales
sgd = SGD(lr=lr, decay = 1e-6, momentum = 0.9, nesterov = True)
model.compile(loss = 'categorical_crossentropy',
        optimizer = sgd,
        metrics = ['accuracy'])

model.fit_generator(train_generator,validation_data=
val_generator,steps_per_epoch=steps_size/4,
validation_steps=validation_steps_size/8,epochs = epochs,
            callbacks=[LearningRateScheduler(lr_schedule),
            ModelCheckpoint('/content/gdrive/My
Drive/Universidad/Proyecto/modelos/model_testTomas.h5', save_best_only = True)])
```

Después de procesar el modelo, se guarda en gDrive, ya que como el entrenamiento requiere cierto tiempo, merece la pena guardar el modelo para las pruebas posteriores. Si bien el acierto o accuracy de la red durante el entrenamiento en el conjunto de validación es razonable, falla miserablemente con los datos reales, como inconsciente se esperaba.

·El siguiente sprint, debe mejorar nuestro conjunto de datos con imágenes reales y probar arquitecturas más avanzadas de Deep learning.

4.2.4.2 Sprint 2: Más datos, profundidad de la red y modelos pre entrenados

El primer requisito para mejorar los datos es incluir fragmentos de partidos o peloteos reales. Este proceso requiere mucho tiempo ya que hay que ir casi frame a frame para realizar el corte del vídeo de un peloteo completo para ir guardando solo la parte de los golpes. Además, en el tenis existe una predominancia del golpe de derecha, ya que la mayoría de jugadores, las bolas que van alrededor de la pista se colocan para realizar la derecha, “cubriéndose” el revés, golpe más débil habitualmente. Esto implica una gran cantidad de tiempo para con los videos disponibles conseguir un dataset suficiente. A la postre se verá que ni siquiera con este esfuerzo ha sido suficiente para conseguir datos suficientes o un equilibrio necesario para la obtención de buenos resultados.

Una vez preparado nuestro dataset mejorado, se para a utilizar modelos muy potentes entrenados previamente, a los que se les quita la capa de clasificación. Al hacer esto, se aprovecha la capacidad de extraer información de las imágenes y aprovecharla para introducir la imágenes recopiladas para este TFM, permitiendo que las capas densas de neuronas clasifiquen los golpes. La configuración de estos modelos requiere un estudio previo de cómo funcionan cada una de ellas, y de preparación específica de las imágenes requieren para un correcto entrenamiento. Este último punto es muy importante, dado que si no se hace el

preproceso específico en las imágenes específico para ese modelo, la capacidad predictiva será casi nula. Se prueba tanto VGG16 como el modelo más profundo RestNet, del que se incluye el código básico, donde gracias a un procedimiento auxiliar permite fácilmente ir probar diferentes arquitecturas en las capas de clasificación, solo tocando en una línea de configuración:

```
def build_finetune_model(base_model, dropout, fc_layers, num_classes):
    for layer in base_model.layers:
        layer.trainable = False

    x = base_model.output
    x = Flatten()(x)
    x = Dropout(dropout/2)(x)

    for fc in fc_layers:
        # New FC layer, random init
        x = Dense(fc, activation='relu')(x)
        x = Dropout(dropout)(x)

    # New softmax layer
    predictions = Dense(num_classes, activation='softmax')(x)
    finetune_model = Model(inputs=base_model.input, outputs=predictions)

    return finetune_model

def cnn_model():

  base_model = ResNet50(weights='imagenet', include_top=False,
input_shape = (image_height, image_width, 3))

  FC_LAYERS = [512,512,512,512]
  dropout = 0.5

  finetune_model = build_finetune_model(base_model,
                                        dropout=dropout,
                                        fc_layers=FC_LAYERS,
                                        num_classes=num_classes)
  return finetune_model
```

La parte del código del entrenamiento, indicar que es igual que en el anterior caso, salvo que se hacen pruebas con diferentes funciones reguladoras como:

```
LReduceLROnPlateau(patience=2)
```

También se prueba distintos tamaños de batch size para ver cómo afecta al over y under fiffing (el cual no se produce casi en ningún caso).

Los resultados en entrenamiento podrían ser aceptables, por encima del 75%, porque en este casi sí que incluyen imágenes reales el conjunto de validación, pero al procesar un video completo lo que no se consigue superar al porcentaje de imágenes disponibles de cada tipo. Es decir, si nuestra distribución de imágenes de ejemplo de derechas, reveses y nada (fotogramas en los que no hay un golpe) es de 40%, 20%, 40%, nuestra red neuronal tiene un acierto razonable con las derechas y los momentos en los que no hay golpeo, pero falla dramáticamente en el reconocimiento de reveses, ya que la red aprende que es una minoría y nunca lo da como una opción probable.

En este punto se llega al reconocimiento que por esta vía no se va a conseguir el objetivo, no al menos en un tiempo suficiente para la consecución de este TFM, ya que este objetivo por sí mismo, podría haber sido objeto de un TFM completo. Aquí en el apartado de próximos pasos se dan indicaciones de por donde se cree que habría que continuar para lograr el objetivo.

4.2.4.3 Sprint 3: Vuelta a métodos de visión artificial

Este sprint se podría dividir en dos partes; la primera es que se une en un solo script la detección del jugador, siendo la segunda el estudio detenido de cómo conociendo parte de la trayectoria de la bola, pero sin conocer la profundidad, y la posición x del jugador de manera exacta, implementar alguna lógica que permita conocer cuando se golpea una bola, y si es una derecha o un revés.

Durante la integración de los objetivos 1 y 3, también se refina algo el método de detectar la pelota. Primero se abre el filtro para aceptar contornos con radios desde 0.1 a 15, anteriormente estaba de 1 a 8. Esto permite detectar más trayectorias, aunque primero hay que solucionar otro problema generado al ampliar el rango, la aparición de más contornos. Lo que se hace es agrupar los contornos cercanos en píxeles, entendiendo que si están cerca pertenecen al mismo objeto, y se crea un contorno con la suma de los radios de ambos. Esta operación permite detectar la pelota en más ocasiones, obteniendo más trayectorias posibles.

Ahora, el primer paso es detectar cuando hay un golpe o se va a producir en breve. Lo que se hace es que cuando en una frame, se consigue el radio del contorno detectado como pelota (el más cercano a la trayectoria). Si este radio es mayor de 3.8, valor depurado según los vídeos de muestra, se considera que la bola está en el campo del jugador principal. Normalmente lo que ocurre, es que, cuando se detecta la primera bola grande es que se dirige hacia el jugador, y a partir de ahí se cuentan 24 frames (1 segundo), tiempo habitual que tarda la pelota en llegar y que el jugador la responda. Pasados esos 24 frames, se deja en

“barbecho” otros 24 frames, tiempo suficiente para que la pelota pase al campo contrario o no la detecta 2 veces.

Una vez conseguido restringir cuando se realiza un golpe, lo que resta, después de bastantes pruebas con diferentes casuísticas, es detectar el tipo de golpe, derecha o revés. Para ello se marca una zona central imaginaria en la imagen, aproximadamente el tercio central de la pista. Si se detecta una bola, y la posición media del jugador está a la derecha, fuera de la zona central, se entiende que el jugador va a realizar una derecha. Si, por el contrario, la media de posiciones x del jugador está a la izquierda, fuera de la zona central, se cuenta como un revés. Esto es una simplificación, pero que funciona razonablemente por la lógica del propio juego, pero además es que es en los extremos dónde es más habitual que el jugador oculte la pelota con más frecuencia, dificultando obtener suficientes trazas para conseguir la posición de la pelota con fiabilidad.

En la zona central, donde es más habitual que un jugador pueda golpear una derecha en el lado del revés, se opta por otra técnica. Esta ocasión se elige la posición de la pelota detectada con mayor radio. Una vez conseguida la posición x, y la que ya se tenía del jugador, se asume simplemente, que, si la pelota está a la derecha del jugador, el golpe será de derecha y viceversa.

Esta forma de obtener qué tipo de golpe se está realizado es bastante artesana, pero funciona con cierta precisión. Un par de ejemplo de detección de golpes, donde las líneas blancas verticales son los límites de la zona central. Las líneas horizontales son mitad del campo y la línea de fondo. Estas líneas no se han obtenido con el código del objetivo 2 por simplificar el código del piloto, aunque se podrían haber obtenido sin problema.

A la izquierda se marca correctamente una derecha. El color naranja indica en este caso que está en la parte derecha de la pista. La siguiente imagen es también una derecha (forehand), pero que en este caso está más centrado en la pista. Como se puede apreciar la pelota se ha dirigido hacia el lado del revés, aunque el jugador ha evitado golpear de revés, más débil habitualmente, realizando un golpe que se conoce como derecha invertida:

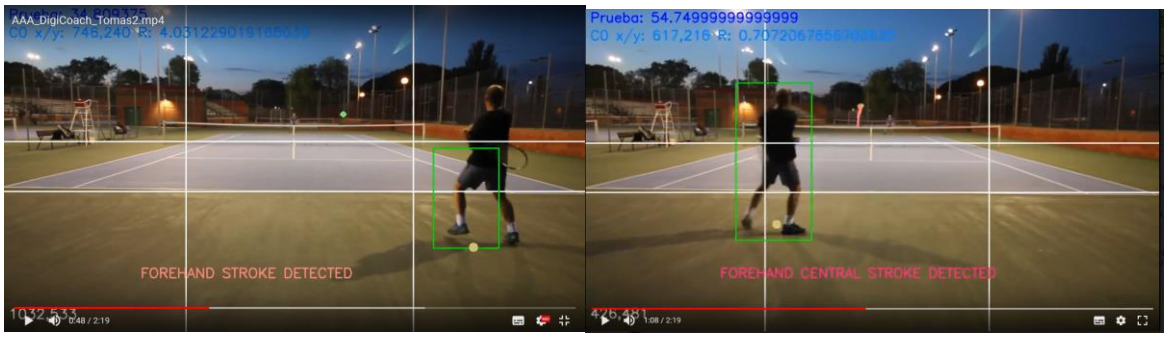

Imagen 33. Dos ejemplos de derechas detectadas

El ejemplo de la izquierda se detecta un revés (backhand) en la zona central de la pista. La imagen de la derecha se detecta un revés, pero en este caso el golpe era una derecha. El procedimiento de detección tiende a fallar cuando la bola viene muy cruzada, es decir, la trayectoria en la imagen es muy diagonal más que vertical:

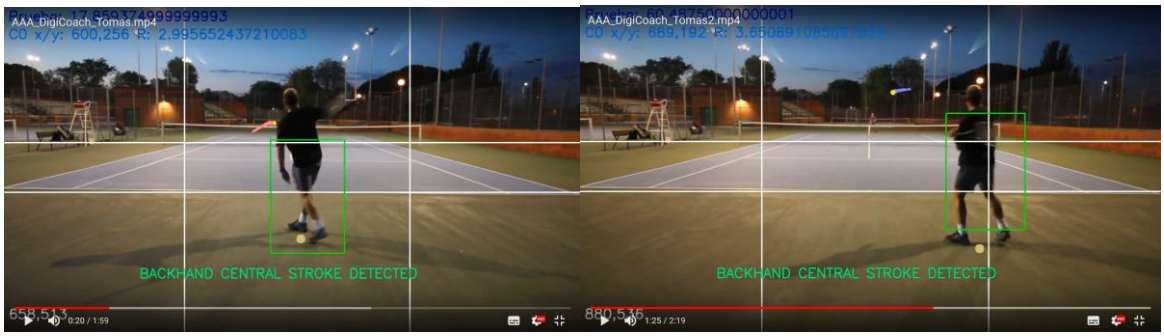

Imagen 34. Dos ejemplos de reveses detectados, a la derecha erróneamente

4.3 Resultados

4.3.1 Detección del Jugador

Al finalizar esta parte del piloto, el proceso es capaz de situar muy fielmente a la realidad la posición x del jugador y aproximada de la y. El proceso posterior explicado en el apartado 4.2.1.3 permite obtener cuantos metros se desplaza el jugador por la línea de fondo. Además, el procedimiento es bastante robusto y funciona igual de bien en diferentes entornos, de día y noche, distintos jugadores o pista.

Para comparar el desplazamiento calculado por este piloto y la real no se ha contado con ninguna herramienta 100% fiable. El uso de un GPS no es válido porque el propio margen de error del sistema, al menos en su versión comercial, no es suficiente para detectar los desplazamientos típicos del tenis, cortos y rápidos. La comparación del resultado se ha

realizado a mano, estudiando los vídeos, y conociendo las medidas de la pista ir sumando los desplazamientos.

Por tanto, no se puede ofrecer un porcentaje exacto de acierto en este parámetro al no contar una fuente precisa, pero sí que los valores son muy buenos para poder utilizar en posteriores análisis al partido, y sobre todo de manera comparativa entre diferentes encuentros. Permite saber si en un partido el jugador ha tenido que desplazarse más metros o menos, y contrastarlo con los resultados.

La secuencia mostrada a continuación son solo una parte de 100 frames (4 segundos) de un vídeo, que comienza cuando el jugador está terminando un revés. Se han seleccionado las más representativas. Cuando está estático se observa que no suma metros, pero a medida que durante varios frames se desplaza en una misma dirección, comienza a medir distancias. Esta secuencia la suma que da el script es 1,75 metros; aunque no dé tiene la medida precisa, se puede estimar. La línea de individuales está centro de la pista a 4.115 metros. El jugador en estas frames se mueve desde la mitad de la parte izquierda de la pista (2 metros del centro) hacia la derecha sin llegar al centro (1 metro aprox.) y luego vuelve hacia la izquierda (un metro), por lo que la cuenta que se procesa parece ajustada a la realidad.

La imagen de la izquierda, el jugador acaba de dar un revés; la del medio, varias frames después, está esperando el siguiente golpe y a la derecha comienza a moverse:

Imagen 35: Suma Metros Jugador: inicio secuencia

Aunque sutil, el jugador mueve el pie derecho hacia el centro de la pista. El desplazamiento se ve mejor comparando con la primera imagen; se ha movido del pixel x 133 al 194:

Imagen 36: Suma Metros Jugador: secuencia intermedia

A la izquierda está el jugador en espera, al medio (varias frames después) está preparando una derecha. Si se observa, el jugador claramente se ha desplazado hacia la izquierda, pero sin embargo el contador de metros solo ha avanzado 0,02 metros. Esto se debe que para solo se da por bueno cuando durante 7 frames el jugador se desplaza en la misma dirección. Esto evita las pequeñas imprecisiones al calcular la posición del jugador, 7 frames es menos de 1/3 de segundo, donde es imposible que un jugador pueda desplazarse significativamente a izquierda y derecha por rápido que sea, por lo que si hay cambios de posición se toman como imprecisiones del sistema:

Imagen 37. Suma Metros Jugador: secuencia final

4.3.2 Referenciar medidas de la pista de tenis

Los resultados sobre los casos de ejemplo utilizados en este TFM son buenos, y más que suficientes teniendo en cuenta el objetivo de este paso. No se busca segmentar perfectamente, aunque se consigue de forma muy aproximada, si no que sirva de referencia para poder situar al jugador o la pelota sobre la pista. Por tanto, el objetivo se ha conseguido satisfactoriamente.

Como ocurre en el objetivo anterior, y en como ocurrirá en el siguiente, la comprobación de los resultados se tiene que realizar a mano, pero en este caso la comprobación visual de la imagen es sencilla y rápida. El siguiente ejemplo es en una imagen nocturna, en la que incluso se ajusta más que a la mostrada en el apartado 4.2.2 sprint 4, donde la línea vertical de saque está desviada en el modelo debido a la colocación un poco ladeada de la imagen:

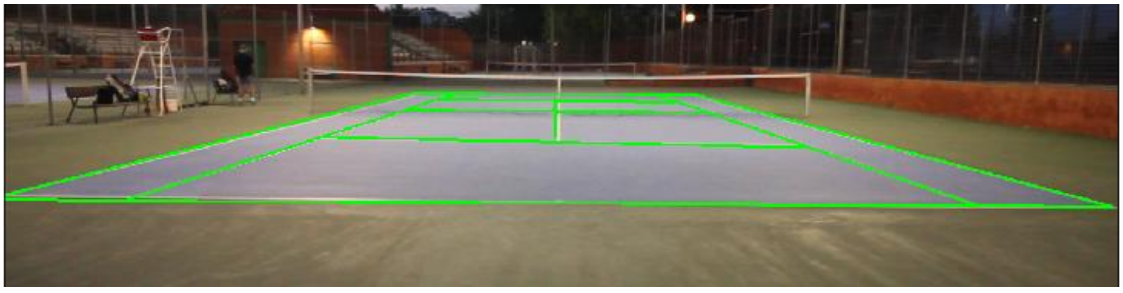

Imagen 38. Detección Líneas, Resultado final noche

Las líneas se ven a trazos, porque no se aplica anti-aliasing que suavice las líneas.

4.3.3 Seguir la pelota

Cada iteración realizada ha ido reconociendo la posición de la pelota con mejor precisión, incluso procesando la trayectoria, el cual se puede utilizar posteriormente para inferir el tipo de efecto que se ha imprimido a la pelota, un dato que en análisis técnicos puede ser muy útil para conocer cómo afecta al jugador y al contrario los diferentes modos de golpeos.

Igualmente, la velocidad de la pelota se podría llegar a estimar, ya que se conocen las medidas de la pista en la realidad, la posición de la pelota y las líneas de la pista en la imagen. Para completar el cálculo hace falta conocer la profundidad de la pelota con exactitud, porque cuanto más lejos está de cámara más complicado es. Esto se puede calcular, aunque no de una manera demasiado precisa, por el tamaño del radio del contorno detectado de la pelota, aunque siempre será una medida aproximada. Como se comentará en el apartado correspondiente, quedará como mejora de este TFM conseguir información más detallada sobre la pelota y que mejoraría notablemente la calidad de los reportes que se extraen de los vídeos.

Cuando se ha reorientado la forma de abordar el objetivo 4, y se ha juntado la posición de la pelota con la del jugador para distinguir los golpes, se ha refinado consiguiendo reconocer las frames del golpeo del jugador, adivinado el tipo de golpe que está realizando.

Se escoge del video grabado por la noche, una secuencia de 85 frames que comprenden un ciclo completo desde que llega la pelota al rival, la golpea, llega al jugador principal, la devuelve y la pelota llega de nuevo al contrario. Esta secuencia se encuentra a partir de la frame 2630 de dicho video. Se ha contado frame a frame, realizando la anotación manual, y el resultado es el siguiente.

La tabla 1 muestra el total de frames en los que se puede ver la pelota o por el contrario está oculta. Se puede ver que más de un tercio de las imágenes donde imposible ver la pelota, bien porque el jugador principal está delante, o porque está demasiado lejos o difuminada. Aquí cabe comentar, que dentro de las bolas no visibles, se reconoció un contorno erróneo, y 3 contornos válidos, que a simple vista no es posible ver vestigio alguno de la pelota, aunque he podido confirmar que son correctos al ver las siguientes frames porque se ve que siguen la trayectoria :

Tabla 1. Frames Visibles en Vídeo

Total Frames	**Número Frames**	**Porcentaje Sobre Total**
Bolas Visibles	56	65,88%
Bolas **No** Visibles	29	34,12%
TOTAL	85	100 %

La tabla 2 muestra de las frames en las que la pelota se puede ver, la mayoría de las veces hará falta hacer zoom en la imagen. Se distinguen si no se ha detectado nada (no cuenta), se detecta bien o mal, y un apartado final cuando se detecta la bola, pero también contornos erróneos. Curiosamente, el número de frames en el que se detecta la bola y en el que no se igual, un 46%. Aunque hay muchas frames en las que no se detecta nada, lo importante es que el porcentaje de errores, sumando los caso que encuentra la bola, pero también otro contorno es del 7%, lo cual es una tasa bastante baja:

Tabla 2. Clasificación frames detección pelota

Total Frames Visibles	Número Frames	Porcentaje Sobre Total
No se cuentan	26	46,43%
Se Cuenta Mal	3	5,36%
Se Cuenta Bien	26	46,43%
Se Cuenta Bien y Mal	1	1,79%
TOTAL	56	100 %

Finalmente, la tabla 3 es el número de trayectos de bola detectados. Esto es, cuando se detectan contornos en frames consecutivas en posiciones cercanas. Contiene la información del total de frames en los que se puede ver la pelota, en casi un tercio es capaces de conocer la trayectoria. Lo más importante es que no hay ninguna trayectoria detectada que sea errónea:

Tabla 3. Trayectorias detectadas en video

Total Trayectorias	Número Frames	Porcentaje Sobre Total
Trayectorias Buenas	18	32,14%
Trayectorias Malas	0	0%
No hay trayectoria	38	67,86%
TOTAL	56	100 %

La secuencia que se ha analizado está elegida al azar. Sí se han encontrado trayectorias erróneas en otras observadas, aunque en ningún caso es un porcentaje algo. Algunos casos detectan la raqueta como trayectoria, porque también es un elemento que se mueve a gran velocidad.

No se ha intentado conseguir la profundidad a la que está la pelota en campo contrario, aunque se piensa que siguiendo el trabajo por la línea marcada se puede obtener resultados bastante aceptables, teniendo en cuenta que falta el uso de al menos 2 cámaras, para con el cruce entre ellas calcularlo con mayor precisión la posición de la pelota en 3D. Para los objetivos marcados en este TFM con detectar acertadamente la pelota en campo propio es suficiente para lograr la información deseada.

Este objetivo se da como conseguido porque se localiza en la imagen la posición de la pelota, y junto con la información de los objetivos 1 y 2 ayudara resolver el objetivo 4.

4.3.4 Clasificar el tipo de golpe

A continuación, se incluyen algunas estadísticas del acierto de la clasificación de golpes del video con luz anocheciendo. La tabla 4 contiene el número de golpes detectados satisfactoriamente. Como se aprecia, la mayor parte de los golpes se detecta adecuadamente; realmente se detectan todos, y los errores viene dados porque se cuentan dos veces. Esto ocurre en bolas muy altas o esquinadas, que la cuenta de seguridad de un segundo sin que detecte un nuevo golpe no funciona en estas situaciones:

Tabla 4. Golpes detectados correctamente

Golpes Detectados	Número Golpes	Porcentaje Sobre Total
Golpes Buenos	64	91,43%
Golpes Malos	6	8,57%
TOTAL	70	100 %

La tabla 5 muestra los aciertos sobre el tipo de golpe. El acierto es del 70%, siendo un 8,57% golpes que se cuentan dos veces:

Tabla 5. Tipos de golpes detectados correctamente

Tipo de Golpe		Porcentaje Total
Acierto	49	70,00%
Error	15	21,43%
No hay golpe	6	8,57%
TOTAL	70	100 %

Si se separa los aciertos por cada tipo de golpe, se ve que predominan las derechas, y además sobre estas hay un mayor porcentaje de acierto:

Tabla 6. Aciertos por tipo de golpe concreto

Tipo de Golpe	Derechas	Porcentaje Total	Reveses	Porcentaje Total
Acierto	39	75,00%	10	55,56%
Error	8	15,38%	7	38,89%
No hay golpe	5	9,62%	1	5,56%
TOTAL	52	100 %	18	100 %

4.4 Problemas encontrados

La dificultad que entraña obtener información valiosa, con imágenes en las que muchas veces el jugador tapa la pelota, imágenes borrosas o distintas situaciones de luz es grande. A

continuación, se comentan los mayores encontrados y los que han resultado más difíciles de resolver.

4.4.1 Detección del Jugador

Este objetivo ha sido relativamente más simple que el resto, aun así no ha estado exento de retos:

- Múltiples entornos reconocidos: Se reconocen varios contornos que indican movimiento, por lo que se ha decidido quedarnos con el mayor.
- Contornos erróneos o parciales: En algunas ocasiones, el contorno obtenido no encuadra al jugador enteramente. Esto se debe normalmente porque en esas frames el jugador está esperando inmóvil o durante un golpe a veces se reconoce solo la parte del brazo y raqueta que es la que realiza más movimiento.
- Por los problemas anteriores, para procesar la posición x (ancho imagen) se ha podido solventar haciendo la mediana de las posiciones anteriores, pero esto no servía con precisión para coordenada y, ya que esta presentaba saltos entre frames mayores.

4.4.2 Referenciar medidas de la pista de tenis:

- Variabilidad de los colores: Los colores de las líneas se vio que no eran fáciles de capturar y que funcionara en diferentes condiciones. Esto se debe en parte a las propias imágenes, que en ocasiones “parecen” coger algo del color de la pista por lo que tienden al verde o azul. Esto hace que sea muy complejo una segmentación por color genérica.
- La luz: Los cambios de luz y las sombras tienen igualmente el efecto de cambiar el color en las imágenes. Esto provoca que si se amplía mucho el rango de colores se cogen partes de la pista.
- La diversidad de espacios de colores y posibles operaciones para resolver el problema hace que lleve mucho tiempo dar con una solución válida.
- Para modelar las líneas que no se aprecian en la imagen, las más lejanas, se tiene que tener en cuenta la profundidad de imagen, lo que obliga a introducir un valor equivalente a la longitud y ángulo de la cámara a la línea de fondo. Gracias a este parámetro, y el conocimiento de las medidas de la pista, se puede estimar las líneas de la pista contraria con buen resultado.

4.4.3 Seguir la pelota

- Al igual que con las líneas, la segmentación del color es complicada, pero se encuentra una función que solución muy razonablemente el problema.
- La dimensión de la profundidad ha sido muy complicada de obtener, ya que no todas las veces cuando está la pelota cerca de la cámara se detecta un contorno más grande. Depende mucho de la velocidad que lleva, porque en ocasiones casi no se desplaza en la imagen (si va muy recta la bola) por lo que casi no se detecta movimiento. Otras veces es problema de la máscara de color, que igualmente solo segmenta una pequeña parte de la pelota.
- Como ha sido imposible reducir a cero los falsos positivos, la solución de crear trayectorias ha sido muy satisfactoria, aunque algo compleja de llevar a cabo.

En definitiva, aunque a nivel de imagen se sigue bien la pelota, el mayor problema es conocer a ciencia cierta su posición cerca o lejos de la cámara.

4.4.4 Clasificar el tipo de golpe

Este objetivo ha sido la fuente de los problemas más complejos, hasta el punto de obligar a replantear la forma de abordarle debido a que en sí mismo probablemente podría haber sido un TFM completo. Los problemas encontrados más relevantes han sido:

- Recopilación y preparación de los datos: He recopilado más de 100 videos de 2/3 segundos, y 4 videos de varios minutos grabados por mí. El enfoque de los vídeos cortos era entrenar la red neuronal, pero de los vídeos largos era para ser usado de manera general. Se ha demostrado que al no ser el mismo entorno los videos cortos que los largos, la red no generalizaba bien. La corrección de esta situación sería manualmente recorrer los vídeos largos e ir cortando frame a frame. Este trabajo se hizo hasta contar con más de 20 vídeos adicionales, aunque con un claro sesgo hacia los golpes de derecha. Este trabajo de recopilación ha sido costoso y aun así creo que no es suficiente.
- Arquitectura CNN: Las arquitecturas de redes de reconocimiento son muy complejas, con muchas capas, por lo que necesitan muchos datos para entrenarlas. La alternativa es utilizar modelos pre entrenados y modificar solo las capas últimas donde se realiza la clasificación. He probado VGG16 y RestNet, y aunque en este último se conseguía resultados en validación del 70%, básicamente ofrecía porcentajes cercanos a la distribución de datos, solo modificándose un poco los resultados entre las clases más

numerosas, el golpe de derecha y los momentos en los que no se hacía nada. En estas clases la predicción podría hasta ser razonable, pero insuficiente a nivel global.

- Entrenamiento: No cuento con un ordenador potente, por lo que he utilizado Colab. Funciona bien, pero tiene limitaciones de tiempo, que a veces compartes con más usuarios, necesidad de subir los ficheros en Zip,… Los entrenamientos se alargaban una eternidad, por lo que ciertas pruebas, después de horas se cortaba, quizás antes de que empezara a mejorar. Este hecho impedía realizar pruebas rápidas, por ejemplo, entrenando una red como RestNet desde cero.
- El propio entorno: Algunos artículos construyen reconocedores de acciones deportivas; unos cuentan con buenos datasets, otros con sensores adicionales como Kinect, algo con lo que no contamos. Además, al estar de espaldas, había frames en las que para un humano también era complicado saber de qué golpe se trabaja
- Enfoque visión Artificial: Conseguido buen acierto al conocer las frames en las que hay un golpe. También la posición horizontal del jugador y pelota, pero cuando esta viene con trayectorias anguladas y con velocidad (no se detecta bien) es complicado estimar si viene por la derecha o por el revés del jugador. Este problema viene, por la comentada dificultad de conocer la profundidad a la que está la bola.

5 CONCLUSIONES Y TRABAJO FUTURO

5.1 Conclusiones

Una vez repasados cada uno de los objetivos específicos, es hora de hacer puesta en común de lo conseguido, y se ha podido contestar positivamente a la pregunta que quiere contestar este TFM. ¿Cómo crear un sistema de anotación automática en tenis, con videos de baja calidad? Se puede contestar que por lo experimentado en este piloto que las técnicas empleadas sirven para crear un sistema de anotación, aunque aún hay que vencer algunos retos. Recopilando se ha conseguido:

1. Conocer la distancia recorrida por el jugador cuando está en el fondo.
2. Se conoce la posición de la pelota en la imagen, y saber si está en el campo del jugador principal, pero con poca precisión sobre la profundidad exacta.
3. Se sigue la trayectoria de la pelota, pudiendo saber cuándo asciende o desciende, importante para conocer si el golpe lleva más o menos efecto.
4. Se conoce la situación de la pista, por lo que se pueden montar informes de tipo zona caliente de por donde se mueve el jugador, o si se consigue un poco más de precisión, de donde va cada golpe.
5. Se detecta cuando se produce un golpe 91,43%, y si es una derecha o un revés con un 70% de acierto.

Conviene también comentar que se han empleado diferentes técnicas bajo el paraguas de la IA, lo cual tiene la complejidad inicial de aprender a utilizarlas, pero una vez conocidas son herramientas muy útiles:

- Técnicas de visión artificial: máscaras de color, diferencia de imágenes, filtros gaussianos y de Canny, operaciones and/not.
- Aprendizaje automático no supervisado: se ha usado k-means, técnica de clustering, para segmentar el background de las imágenes.
- Deep Learning: aunque no se consiguió el objetivo, se ha utilizado data augmentation, transfer learning, redes convolucionales, así como diferentes técnicas de regulación para evitar overfitting en los entrenamientos, como batch normalization y decay learning rates.

El objetivo general de este TFM se ha cumplido al haber sido posible a partir de diferentes vídeos extraer información relevante para un análisis posterior al partido, e incluso refinando el código y hardware suficiente, se podría utilizar durante el partido.

El punto de partida del TFM son videos complicados para tratar, y se ha conseguido la extracción información valiosa, pero sí que hay que comentar la dificultad de cada una de las partes, que podrían ser mejoradas o refinadas para conseguir una aplicación que generara informes más fiables.

Este trabajo aporta un ejemplo de herramienta que se puede llevar a cabo en la realidad, y, además, al necesitar de pocos recursos, ser utilizada por cualquier escuela o deportista de cualquier nivel socio-económico.

5.2 Trabajo futuro

Mejoras directamente aplicables al piloto

- Mejorar la precisión de la posición del jugador en el eje y.
- Conseguir seguir la pelota al menos en las imágenes que a simple vista se puede distinguir, lo que mejoraría el cálculo de las trayectorias.
- Ajustar mejor la línea vertical del cuadro de saque, ya que habitualmente desaparecía de las máscaras, y la estimación que se realiza no se encaja perfectamente si la cámara no está recta con respecto a la pista.
- Optimizar los cálculos, ya que el cálculo de por cada segundo, de los objetivos 1, 3, 4 simultáneamente tarda unos 10 segundos por cada segundo (x10) en Colab.

Además, otras mejoras sensibles:

- Implementar la detección de golpes con redes neuronales, restringiendo las búsquedas a las frames en las que se detecta un golpe con alto porcentaje de fiabilidad. Existen artículos dónde se recomienda utilizar redes recurrentes para la detección de acciones (Mora & Knottenbelt, 2017).
- Convertir el código para que se pueda ejecutar desde una app o web que directamente se pueda subir los videos desde un móvil.
- Preparar y presentar informes con los datos obtenidos, pudiendo almacenarlos en algún tipo de base de datos, para futuras consultas
- Detección de frames en los que no hay juego, lo que aliviaría los cálculos y evitaría errores.
- Se podría combinar con otros sensores o cámaras para mayor precisión, aunque perdería el enfoque inicial de este TFM.

6 REFERENCIAS

Aprimatic. (2019). Lector de Matrículas - Aprimatic. Retrieved September 3, 2019, from aprimatic.es website: https://www.aprimatic.es/tienda/barreras-automaticas-3/lector-de-matriculas-barreras-automaticas-3/lector-de-matriculas-apripark-view/

Ayn de, J. (2019). Computer Vision Applications - Shopping, Driving and More. Retrieved June 1, 2019, from Emerj Artifiicial Intelligent Research website: https://emerj.com/ai-sector-overviews/computer-vision-applications-shopping-driving-and-more/

Barrionuevo, E. (2009). Operaciones Morfologicas. Retrieved September 14, 2019, from slideshare.net website: https://www.slideshare.net/balfier/operaciones-morfologicas

Boden, M. A. (2016). *AI : its nature and future*. Retrieved from http://www.informationr.net/ir/reviews/revs568.html

Brownlee, J. (2016). Overfitting and Underfitting With Machine Learning Algorithms. Retrieved September 14, 2019, from Machine Learning Mastery website: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Calvo, D. (2017). Red Neuronal Convolucional CNN - Diego Calvo. Retrieved September 14, 2019, from diegocalvo.es website: http://www.diegocalvo.es/red-neuronal-convolucional/

Chenlei Guo, Qi Ma, & Liming Zhang. (2008). Spatio-temporal Saliency detection using phase spectrum of quaternion fourier transform. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. https://doi.org/10.1109/CVPR.2008.4587715

Cohen, T. S., Geiger, M., Koehler, J., & Welling, M. (2018). *Spherical CNNs*. Retrieved from http://arxiv.org/abs/1801.10130

El Khattabi, Z., Tabii, Y., & Benkaddour, A. (2016). Video Shot Boundary Detection in Sport Video Using the Scale Invariant Feature Transform. *Applied Mechanics and Materials*, *850*, 152–158. https://doi.org/10.4028/www.scientific.net/AMM.850.152

Elsayed, G. F., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I., & Sohl-Dickstein, J. (2018). *Adversarial Examples that Fool both Computer Vision and Time-Limited Humans*. Retrieved from http://arxiv.org/abs/1802.08195

Erol, V. (2017). *Line Extraction and Player Tracking in Tennis Videos*. https://doi.org/10.20944/PREPRINTS201706.0013.V1

Fang, Z., Cao, T., Yang, J., & Xing, Y. (2019). Dense dilation network for saliency detection. In H. Yu, Y. Pu, C. Li, & Z. Pan (Eds.), *Tenth International Conference on Graphics and Image Processing (ICGIP 2018)* (p. 51). https://doi.org/10.1117/12.2524200

Foxtenn. (2019). Weblet Importer. Retrieved September 3, 2019, from Fostenn.com website: http://www.foxtenn.com/

Hawkins, A. J. (2018). Riding in Waymo One, the Google spinoff's first self-driving taxi service - The Verge. Retrieved June 22, 2019, from theverge.com website: https://www.theverge.com/2018/12/5/18126103/waymo-one-self-driving-taxi-service-ride-safety-alphabet-cost-app

Hettena, D. (2018). 5 Breakthrough Computer Vision Startups to Watch in 2019. Retrieved June 2, 2019, from rocketspace.com website: https://www.rocketspace.com/corporate-innovation/5-breakthrough-computer-vision-startups-to-watch-in-2019

Ho, Daniel; Liang, Eric; Liaw, R. (2019). 1000x Faster Data Augmentation – The Berkeley Artificial Intelligence Research Blog. Retrieved September 14, 2019, from Berkeley Artificial Intelligence Research website: https://bair.berkeley.edu/blog/2019/06/07/data_aug/

Hulfish, G. (2017). Stop Fighting Over Line Calls with the In/Out Tennis Sensor | Digital Trends. Retrieved September 3, 2019, from digitaltrends.com website: https://www.digitaltrends.com/outdoors/in-out-tennis/

IBM. (2019). El modelo de redes neuronales. Retrieved September 14, 2019, from ibm.com website: https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/neuralnet/neuralnet_model.html

ITusa. (2019). Home | iTUSA Tennis. Retrieved June 30, 2019, from itusatennis.com website: http://www.itusatennis.com/es

Jacobs, H. (2018). China's "Big Brother" surveillance technology isn't nearly as all-seeing as the government wants you to think | Business Insider España. Retrieved June 22, 2019, from Business Insider website: https://www.businessinsider.es/china-facial-recognition-limitations-2018-7?r=US&IR=T

Johnson, D. (2016). The battle of the tennis sensors - Gadgets & Wearables. Retrieved May 1, 2019, from gadgetsandwearebles.com website: https://gadgetsandwearables.com/2016/04/16/the-battle-of-the-tennis-sensors-who-is-winning/

Johnson, K. (2019). Google's lung cancer detection AI outperforms 6 human radiologists | VentureBeat. Retrieved June 22, 2019, from venturebeat.com website: https://venturebeat.com/2019/05/20/googles-lung-cancer-detection-ai-outperforms-6-human-radiologists/

Johnson, K. (2019). Google's SpecAugment achieves state-of-the-art speech recognition without a language model. Retrieved September 3, 2019, from venturebeat.com website: https://venturebeat.com/2019/04/22/googles-specaugment-achieves-state-of-the-art-speech-recognition-without-a-language-model/

Kurose, R., Hayashi, M., Ishii, T., & Aoki, Y. (2018). Player pose analysis in tennis video based on pose estimation. *2018 International Workshop on Advanced Image Technology (IWAIT)*, 1–4. https://doi.org/10.1109/IWAIT.2018.8369762

LongoMatch | La herramienta de videoanálisis para entrenadores, analistas deportivos y jugadores. (2019). Retrieved June 30, 2019, from longomatch.com website: https://longomatch.com/es/

López Briega, R. E. (2017). Introducción al Deep Learning. Retrieved September 14, 2019, from relopezbriega.github.io website: https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

Lyu, C., Liu, Y., Li, B., & Chen, H. (2015). Multi-feature based high-speed ball shape target tracking. *2015 IEEE International Conference on Information and Automation*, 67–72.

https://doi.org/10.1109/ICInfA.2015.7279260

Marti-Vargas, J. R., Ferri, F. J., & Yepes, V. (2013). Prediction of the transfer length of prestressing strands with neural networks. *Computers and Concrete*, *12*(2), 187–209. https://doi.org/10.12989/cac.2013.12.2.187

Martínez-Gallego, R. (2015). (PDF) El análisis de la táctica en el tenis. Retrieved June 30, 2019, from E-Coach - Revista Electrónica del Técnico de Tenis website: https://www.researchgate.net/publication/291814049_El_analisis_de_la_tactica_en_el_tenis

Massenzio, M. (2017). Detection of Fast-Moving Objects (FOM) using OpenCV – Code Trips & Tips. Retrieved September 3, 2019, from codetrips.com website: https://codetrips.com/2017/10/29/detection-of-fast-moving-objects-fom-using-opencv/

Molins renter, A. (2018). China estrena su 'Gran Hermano.' Retrieved January 26, 2019, from lavanguardia.com website: https://www.lavanguardia.com/internacional/20180503/443196686690/china-puntuacion-ciudadanos-delitos-sociales.html

Mora, S. V., & Knottenbelt, W. J. (2017). Deep Learning for Domain-Specific Action Recognition in Tennis. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 170–178. https://doi.org/10.1109/CVPRW.2017.27

Nacsport | Software de Vídeo Análisis para todos los Deportes. (2019). Retrieved June 30, 2019, from Nacsport website: https://nacsport.com/index.php?lc=es-es

Nakhuda, D. (2019). 9 Computer Vision Predictions for 2019 | Mighty AI. Retrieved June 2, 2019, from https://mighty.ai/blog/9-computer-vision-predictions-for-2019/ website: https://mighty.ai/blog/9-computer-vision-predictions-for-2019/

Ning, M. (2019). SIFT(Scale-invariant feature transform) - Towards Data Science. Retrieved September 14, 2019, from towardsdatascience.com website: https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37

Orduz, Ó. (2017). La entropía y su aplicación en los sistemas de información. Retrieved September 14, 2019, from Medium.com website: https://medium.com/@deimian5/la-entropía-y-su-aplicación-en-los-sistemas-de-información-a8d5e5c2379c

Pastor, J. (2017). La empresa china que se lleva su fábrica a Estados Unidos para ahorrar costes de fabricación. Retrieved September 3, 2019, from xataka.com website: https://www.xataka.com/empresas-y-economia/la-empresa-china-que-se-lleva-su-fabrica-a-estados-unidos-para-ahorrar-costes-de-fabricacion

raona. (2017). Los 10 Algoritmos esenciales en Machine Learning - Raona. Retrieved September 14, 2019, from raona.com website: https://www.raona.com/los-10-algoritmos-esenciales-machine-learning/

Rosebrock, A. (2015). Ball Tracking with OpenCV - PyImageSearch. Retrieved June 23, 2019, from pyimagesearch.com website: https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

Rosebrock, A. (2016). Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox - PyImageSearch. Retrieved May 2, 2019, from pyimagesearch.com website: https://www.pyimagesearch.com/2015/06/01/home-

surveillance-and-motion-detection-with-the-raspberry-pi-python-and-opencv/#comment-515704

Rosebrock, A. (2018). OpenCV Saliency Detection - PyImageSearch. Retrieved May 2, 2019, from pyimagesearch.com website: https://www.pyimagesearch.com/2018/07/16/opencv-saliency-detection/

Rouse, M. (2017). ¿Qué es Aprendizaje automático (machine learning)? - Definición en WhatIs.com. Retrieved September 14, 2019, from searchdatacenter.techtarget.com website: https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-automatico-machine-learning

Rozumnyi, D., Kotera, J., Sroubek, F., Novotny, L., & Matas, J. (2016). *The World of Fast Moving Objects*. https://doi.org/10.1109/CVPR.2017.514

Teachabarikiti, K., Chalidabhongse, T. H., & Thammano, A. (2010). Players tracking and ball detection for an automatic tennis video annotation. *2010 11th International Conference on Control Automation Robotics & Vision*, 2461–2494. https://doi.org/10.1109/ICARCV.2010.5707906

Top 20 Facebook Statistics - Updated July 2019. (n.d.). Retrieved September 3, 2019, from https://zephoria.com/top-15-valuable-facebook-statistics/

Tsang, S.-H. (2018). Review: AlexNet, CaffeNet — Winner of ILSVRC 2012 (Image Classification). Retrieved September 14, 2019, from Medium.com website: https://medium.com/coinmonks/paper-review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160

Vincent, J. (2018). Chinese police are using facial recognition sunglasses to track citizens - The Verge. Retrieved June 22, 2019, from www.theverge.com website: https://www.theverge.com/2018/2/8/16990030/china-facial-recognition-sunglasses-surveillance

Wang, W., Shen, J., Guo, F., Cheng, M.-M., & Borji, A. (2018). *Revisiting Video Saliency: A Large-scale Benchmark and a New Model*. Retrieved from http://arxiv.org/abs/1801.07424

Wu, Y., & He, K. (2018). *Group Normalization*. Retrieved from http://arxiv.org/abs/1803.08494

Zamir, A., Sax, A., Shen, W., Guibas, L., Malik, J., & Savarese, S. (2018). *Taskonomy: Disentangling Task Transfer Learning*. Retrieved from http://arxiv.org/abs/1804.08328

Zbeide Rivera, Y. (2018). Propuesta de análisis notacional para el registro - informe táctico como entrenador – analista en tenis = Proposal of notational analysis for registration - tactical report as trainer – tennis analyst. *Buleria.Unileon.Es*. Retrieved from http://buleria.unileon.es/xmlui/handle/10612/8750

Zhou, X., Xie, L., Huang, Q., Cox, S. J., & Zhang, Y. (2015). Tennis Ball Tracking Using a Two-Layered Data Association Approach. *IEEE Transactions on Multimedia*, *17*(2), 145–156. https://doi.org/10.1109/TMM.2014.2380914

7 ANEXOS

El siguiente proyecto GitHub contiene dos notebooks en Python, uno con la detección de líneas y el segundo une la detección del jugador, pelota y los golpes, como se comenta en el capítulo 4.

https://github.com/fordfarlin/Tennis-Annotation

Además, se entrega junto con este TFM un artículo resumen del trabajo realizado:

Anexo_Articulo_Gonzalo_Molina_TFM.docx