Universidad de Valladolid

E.T.S Ingeniería Informática

6th Jul 2019

Master in Computer Engineering

Master's thesis

Understanding Badminton with computer vision

Autor: Diego González Serrador
Tutor: Javier Finat Codes

Agradecimientos

En primer lugar, me gustaría agradecer la ayuda a mi tutor Javier Finat, porque sin haber sido yo el mejor alumno del mundo Javier ha comprendido mi situación personal y se ha adaptado a ella. Javier, rápidamente acepto la tutorización de este trabajo y cada conversación con él ha sido un auténtico placer.

En segundo lugar, a Francisco Delgado del Hoyo que durante su etapa de doctorando ha aportado muchísimo a este trabajo.

En tercer lugar, a mis compañeros del equipo de inteligencia artificial de Nielsen Connect, dirigido por Antonio Hurtado, que con sus consejos durante todo el trabajo han ayudado enormemente al resultado de este.

Finalmente, a mi familia, padres y novia por toda la paciencia que han tenido conmigo durante mi etapa como estudiante en la Universidad de Valladolid y en especial durante el transcurso de este trabajo.

A todos ellos, gracias de corazón.

Abstract

Automatic broadcast video analysis using computer vision is a very interesting area with many real applications such as rival study, sport improvement or computer game simulation.

In this paper, an easily extendable mechanism or *pipeline* is proposed for the automatic extraction of information from badminton matches using computational vision and convolutional neural networks (*CNNs*). The implementations and their corresponding evaluation are provided for some of the steps of this *pipeline*, such as the detection of court lines, poles detection, shutlecook detection and tracking, and players detection. In addition, *datasets* are presented for the training and validation for those stages.

For the implementation of those steps some of the state of the art deep learning architectures for image classification and object detection like ResNet, Faster R-CNN or Mask R-CNN are used.

Finally, required next steps for high scale and production deployment for the proposed system are explained.

Resumen

El análisis automático de retransmisiones deportivas usando visión computacional es un área muy interesante con multitud de aplicaciones cómo el estudio del rival, la mejora deportiva o la simulación de partidos por ordenador.

En este trabajo se propone un mecanismo o *pipeline* fácilmente extensible para la extracción automática de información de partidos de bádminton usando visión computacional y redes neuronales convolucionales (*CNNs*). Se aportan implementaciones y su correspondiente evaluación para algunos de los pasos de ese *pipeline*, como son la detección de las líneas la pista, la detección de los postes, la detección y el seguimiento del volante, y la detección de los jugadores. Además se presentan *datasets* para el entrenamiento y validación de cada una de esas etapas.

Para la implementación de esos pasos se utilizan algunas de las arquitecturas *deep learning* que son o han sido estado del arte para la clasificación de imágenes y para la detección de objetos en imágenes como son ResNet, Faster R-CNN o Mask R-CNN.

Finalmente, se enumeran los siguientes pasos para desplegar un sistema como el propuesto en producción a gran escala.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

El análisis de secuencias de vídeo para la práctica de un deporte plantea problemas relativos al contexto (región en la que tiene lugar el juego) y el seguimiento de los agentes móviles (jugadores y pelota). Cada uno de estos problemas es específico y requiere una adaptación de herramientas de Visión Computacional relativas a la segmentación estática y móvil.

De una forma intuitiva la segmentación afecta a una descomposición espacial (centrada en el terreno de juego) y temporal (centrada en los movimientos y la interacción de jugadores). La segunda se superpone a la primera como una capa superpuesta a la representación (eventualmente variable) de la escena. La segunda presenta una mayor complejidad, pues afecta al seguimiento y, en fases más avanzadas, la identificación de gestos (secuencias temporales de posturas) en relación con los modelos asociados al movimiento de los jugadores. Con ello, se pretende no sólo evaluar características cinemáticas del juego, sino evaluar el rendimiento, identificar estrategias y corregir posibles defectos observables en las secuencias de vídeo.

En el deporte profesional, hasta el más mínimo detalle es importante. Cualquier entrenador y la mayor parte de los jugadores saben que para alcanzar el éxito, la preparación física, la preparación mental, la estrategia y el estudio del rival son tareas imprescindibles, aunque la mayoría de las personas no nos paremos a pensar en ello. Puede restar un poco de emoción al deporte, pero los deportistas profesionales emplean muchísimas horas estudiándose a sí mismos y estudiando al rival. El análisis de los momentos donde fallaron, y aquellos donde acertaron les permite mejorar día a día.

El bádminton es un deporte de raqueta donde la resistencia física, la capacidad mental y la estrategia son cruciales para ganar. El estudio del rival y el de uno mismo, mediante vídeos y otros recursos, es una parte fundamental del entrenamiento de cualquier jugador. Todos los jugadores profesionales de bádminton analizan a sus rivales antes de un enfrentamiento. Un sencillo ejemplo: si supiéramos que un jugador de bádminton falla más después de un punto largo o hace más puntos en una zona concreta de la pista, podríamos utilizar esa información para definir nuestra estrategia.

Está claro que los datos en el deporte profesional pueden marcar la diferencia entre ganar o perder. Sin embargo, ¿cómo obtenemos estos datos?, ¿cuánto tiempo puede dedicar un entrenador o un jugador a ver vídeos de sus rivales?, ¿cuánto dinero podemos gastar para obtener estos datos?. Las respuestas a estas preguntas son cruciales, pues ningún jugador o entrenador puede ver todos los partidos de todos sus rivales o no todos los deportistas pueden pagar cantidades elevadas de dinero para conseguir este tipo de información. Al final, se necesitan herramientas que ayuden a extraer la información relevante de la manera más eficiente posible.

Para obtener la información relevante es necesario transformar los datos en bruto en eventos, en estadísticas y en indicadores que permitan identificar o caracterizar modelos básicos de movimiento, o situaciones relevantes para su estudio. En nuestro caso de uso, los datos en bruto proceden de retransmisiones televisivas de partidos de bádminton profesional en forma de vídeos. Los indicadores están asociados a diferentes tipos de eventos como el golpe donde más falla, la distancia recorrida en un partido o el tipo de golpe más efectivo. Finalmente, las estadísticas que resultan de este análisis pueden ser utilizadas por entrenadores y jugadores de una manera sencilla para alcanzar los objetivos mencionados previamente.

La extracción manual de información útil de un partido de bádminton es una tarea tediosa, pues una persona experta debería revisar los 45 minutos que un partido profesional dura de media anotando las acciones transcurridas. Por ejemplo, deberíamos hacer anotaciones del tipo: en el segundo 5 el jugador A ha hecho un golpe de fondo desde la posición (x, y, z) o el jugador B ha ganado el punto 15 en el segundo 625. Revisando este tipo de anotaciones podemos observar dos problemas, el primero es el tiempo que una persona necesita para anotar por completo todos los detalles de un partido de bádminton; el segundo es la dificultad para realizar estas anotaciones, pues hay detalles que no son evidentes a simple vista, como por ejemplo, la posición exacta de un jugador en un determinado instante. Si se amplían esta tarea a todos los partidos de todos los jugadores profesionales durante un año, la tarea se vuelve inabordable o extremadamente costosa.

Finalmente, un punto importante y que rara vez se menciona es que la tasa de error humana no es cero. En muchos problemas las máquinas han demostrado ser más precisas que los seres humanos, tal y como demostró Andrej Karpanthy [6]. Además, en tareas altamente repetitivas, como las que conciernen a este trabajo, estos errores son más propensos.

1.2. Objetivos

El *objetivo fundamental* es la aplicación de modelos y herramientas de Visión Computacional para la extracción automática y análisis supervisado de los actores y objetos implicados en secuencias de bádminton procedentes de retransmisiones televisivas; así como establecer la secuencia de acciones necesarias para poder construir un sistema experto capaz de extraer la máxima información de estas secuencias.

Este objetivo tiene asociados varios *objetivos secundarios* tales como

- Caracterizar los *modelos* más apropiados para la detección y extracción de características móviles.
- Identificar las *herramientas estadísticas* que proporcionan el agrupamiento de datos más eficientes para la detección de gestos.
- Adaptar los *algoritmos* que presenten mejor rendimiento de acuerdo con las características de los casos de uso.

El análisis automático de vídeos deportivos es un área muy interesante y motivante dentro de la visión computacional [2]; la utilización de estas herramientas haría posible una oleada de nuevas aplicaciones. La aplicación de modelos, datos y algoritmos para el análisis de jugadas y jugadores ofrece posibilidades de extrapolación a partidos más complejos (cuatro jugadores, p.e.) o bien a otro tipo de prácticas deportivas. Asimismo, podría usarse para enriquecer la información relativa a retransmisiones deportivas, complementar las noticias de los periódicos deportivos, ampliar el mundo de las apuestas, realizar simulaciones por ordenador de los partidos y para la extracción automática de las escenas más importantes de una retransmisión deportiva.

Para el análisis de estos vídeos, como ya hemos comentado, usaremos mayoritariamente modelos y herramientas de visión computacional, vinculadas por un lado al Procesamiento y Análisis de Secuencias de Vídeo y a técnicas más novedosas en relación con Sistemas Expertos. Entre estas últimas, prestamos especial atención a las basadas en aprendizaje profundo (*o deep learning*), que han revolucionado la inteligencia artificial en los últimos años. Dentro del aprendizaje profundo, una de las herramientas más utilizadas en este trabajo serán las redes neuronales convolucionales *(Convolutional Neural Networks o CNNs)*.

1.3. Hipótesis

La hipótesis de partida es la siguiente: es posible extraer estadísticas de un partido de bádminton a partir de las secuencias de vídeo grabadas de una forma robusta, con bajo coste y una precisión aceptable. Para validar nuestra hipótesis, utilizamos una combinación de modelos y herramientas basados en visión por computador e inteligencia artificial. Más concretamente, aplicaremos técnicas de *deep learning* para resolver el problema de reconocimiento y detección de objetos y jugadores en las secuencias tratadas.

Para acotar el problema, nos restringimos a vistas frontales y sagitales de la escena con cámaras fijas que siempre están enfocadas al lugar en el que se desarrolla el partido. De este modo, se puede referenciar las secuencias de vídeo analizadas a una estructura fija dada por las líneas marcadas en la pista y en la red. En particular, evitamos vistas oblicuas (que pueden dar lugar a mayores errores de paralaje) y no se lleva a cabo un seguimiento del "volante" por parte de la cámara; ello facilita un tratamiento más eficiente de la trayectoria del volante, incluso bajo condiciones de incertidumbre (cuando no es posible detectarlo en todos los *frames* de la secuencia de vídeo).

Los vídeos proceden de retransmisiones en abierto publicadas a través de internet con resoluciones de 720p (1280 x 720) a 1080p (1920 x 1080), con una tasa de 25 a 30 imá-

genes por segundo (fps). Estas secuencias, proceden siempre de partidos profesionales de bádminton de torneos de primer nivel, donde la iluminación de la secuencias es generalmente aceptable y permite visualizar el desarrollo de las acciones. El funcionamiento de los procedimientos aquí expuestos podría no funcionar en secuencias grabadas en diferentes condiciones, o en partidos *amateur* o torneos regionales.

Generalmente, en relación con los agentes móviles, nos restringimos al caso de dos jugadores en total, es decir, un jugador por cada equipo. Ello facilita la identificación de posturas y el seguimiento de gestos (como concatenación de posturas), minimizando las auto-oclusiones de jugadores del mismo equipo. Asimismo, la identificación de las líneas de la pista y de la red facilita la localización de los jugadores en relación con la escena

A la vista de las limitaciones actuales en las herramientas (modelos, datos, algoritmos) disponibles actualmente y para acotar el alcance del trabajo, es necesario describir los aspectos que no se abordan en este documento. Los más relevantes son:

- Detección y seguimiento de puntos de control, justificado por la ausencia de marcas o sensores *wearables*.
- Seguimiento de formas en movimiento, debido a la elevada variabilidad en la apariencia de las siluetas.
- Caracterización de gestos como composición temporal de posturas a lo largo de la secuencia de vídeo.
- Estimación de las características cinemáticas (localización, velocidad, aceleración) para elementos significativos (features).
- Análisis automático en tiempo real de las jugadas, como consecuencia de la caracterización de gestos y el coste computacional de las soluciones que presentaremos más adelante.
- Predicción del resultado en función de las acciones llevadas a cabo por un jugador.

1.4. Estructura del documento

En el capítulo 2 se introduce el estado del arte de la visión computacional en relación con los objetivos planteados, los últimos avances y las técnicas de interés para el caso de uso planteado, así como revisando soluciones relacionadas existentes en el mercado.

El capítulo 3 explica la aproximación desarrllada para resolver el problema planteado. En primer lugar, se propone un esquema general para extraer información de manera automática o semi-automática de un partido de bádminton explicando los pasos que el procedimiento planteado debería realizar para ello. De este esquema nos centraremos en las etapas de detección de los objetos y jugadores presentes en partidos de bádminton.

En el capítulo 4 de esta memoria, antes de explicar los resultados obtenidos, se introducen los *datasets* utilizados para evaluar las etapas abordadas en este trabajo, así como la metodología utilizada para evaluar cada uno de los experimentos realizados.

En el capítulo 5 se explican los experimentos realizados y los resultados obtenidos para cada una de las etapas abordadas. Cada experimento, en la medida de lo posible se evalúa de forma numérica, aportando resultados objetivos y una visualización.

En el capítulo 6, se exponen las conclusiones del trabajo así como el trabajo futuro necesario para alcanzar el objetivo propuesto. Este trabajo futuro, trata de introducir los siguientes pasos tanto en el ámbito científico como en uno más ingenieril, como p.e. que sería necesario hacer o revisar para desplegar un sistema propuesto en un entorno de producción o qué aspectos habría que revisar para poder ofrecer la solución propuesta a terceras partes.

Finalmente, en los anexos se enumeran los materiales que complementan este documento y se enumeran otros desarrollos secundarios realizados durante este trabajo.

Capítulo 2

Estado del arte

Este capítulo revisa el estado del arte de los modelos, tecnologías y herramientas que se han utilizado o que son referentes para entender este trabajo. Además, se incluye un breve estudio de las soluciones existentes en el mercado con un objetivo similar a los planteados en este trabajo.

2.1. Visión por computador

La visión por computador [34] es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica que pueda ser tratada por un computador; o de forma más sencilla, la visión por computador es el estudio de datos visuales por parte de un computador.

Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión por computador trata de reproducir el mismo comportamiento para que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación. Esta disciplina es interdisciplinar e involucra diferentes áreas científicas, como la geometría, la estadística, la física y diferentes tecnologías, como bases de datos, programación, inteligencia artificial, p.e. (véase la Figura 2.1). La adquisición de los datos se consigue usando diferentes dispositivos y formatos como secuencias de imágenes, vistas procedentes de varias cámaras de vídeo o datos multidimensionales de otros tipos de sensores.

La visión por computador incluye múltiples áreas. En este trabajo utilizaremos métodos de áreas como el procesamiento de imagen, la detección de objetos, la clasificación de imágenes y el seguimiento de objetos en movimiento.

Una *imagen digital* es un mapa de bits, es decir, una función escalar (intensidad en la escala de grises, p.e.) o vectorial (terna de funciones correspondiente a una representación del color, p.e.) definida sobre un dominio asociado al soporte físico de la imagen. Supondremos que el tipo de imagen está fijado y habitualmente tomaremos la representación vectorial RGB para el color. En lo sucesivo, supondremos que todas las imágenes son digitales.

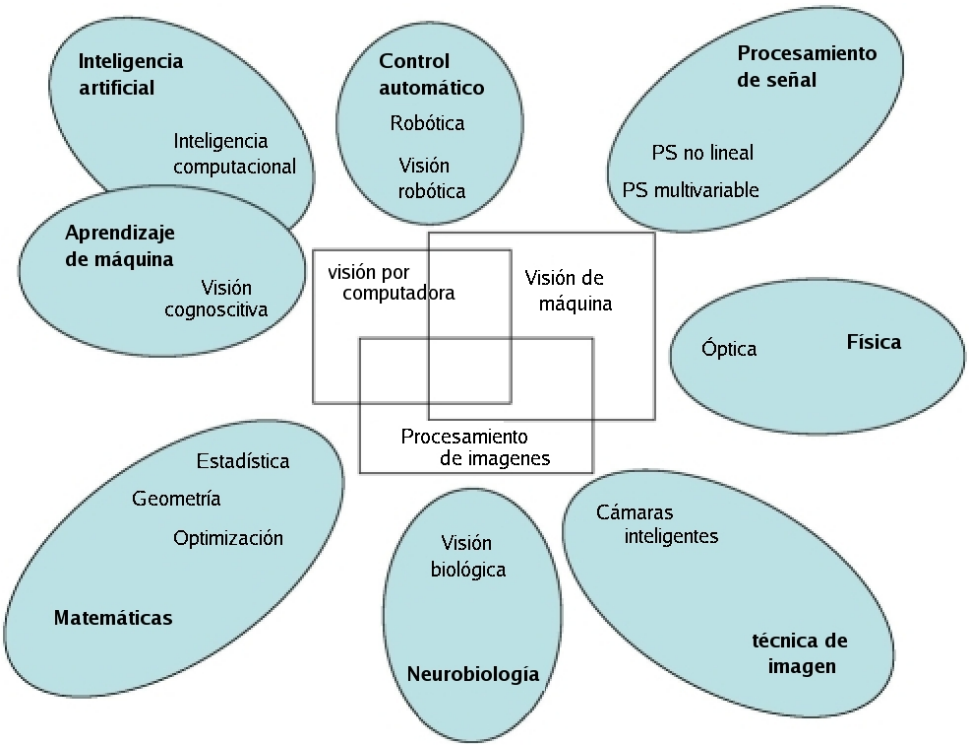

Figura 2.1: Relaciones entre la visión por computadora y otras áreas afines

Las funciones definidas sobre la imagen toman valores discretos con un número k de bits, lo cual da lugar a 2^k posibles valores que se almacenan como valores enteros del intervalo $[0, 2^k - 1]$. Frecuentemente, se toma $k = 8$, lo cual da lugar a una representación para los valores enteros del intervalo $[0, 255]$. Para la función de intensidad en escala de grises se asigna el valor 0 al negro puro y el valor 255 al blanco puro. En el caso de la escala RGB la representación del color para una imagen de 8 bits se lleva a cabo sobre el cubo $[0, 255]^3$.

El objetivo de la *segmentación de imagen* es la descomposición de una imagen en una unión disjunta de regiones r_α con borde $b_\alpha := \partial r_\alpha$ (donde ∂ denota el operador borde) para facilitar el análisis y la interpretación de cada imagen por separado. Para ello, se utilizan técnicas de procesamiento y análisis de imagen que se comentan en la primera subsección.

Una *secuencia de vídeo digital* $S(t)$ es una colección consecutiva de imágenes digitales ó *frames*. Inicialmente, se supone que la secuencia tiene un número comprendido entre 20 y 30 frames por segundo (fps).

El estudio de las propiedades de una secuencia de vídeo se lleva a cabo aplicando de forma secuencial las técnicas de procesamiento y análisis de imagen para cada frame de la secuencia o bien para un muestreo (regular o aleatorio) de la secuencia. En este caso, tanto las regiones $r_\alpha(t)$ como sus bordes $b_\alpha(t) := \partial r_\alpha(t)$ tienen una etiqueta temporal para facilitar su localización a lo largo de la secuencia.

El objetivo de la *segmentación de vídeo* es la descomposición de una secuencia en una colección finita de subsecuencias que tienen un "contenido similar". Para ello, es necesario

extraer características básicas de las imágenes (mediante histogramas) y fijar un umbral de variación para las mismas. Estas cuestiones se abordan en la segunda subsección.

2.1.1. Procesamiento y análisis de imagen

El *objetivo del Procesamiento de Imagen* es la detección y extracción de características ó "hechos significativos" (*features*) de una imagen digital que sean relevantes para el problema. Estas características pueden ser relativas a regiones o bien relativas a los bordes de las regiones:

- Las características relativas a regiones r_α están asociadas a variaciones radiométricas (relativas a la luz, es decir, intensidad en escala de grises o color) por debajo de un umbral. Por ello, se basad en la continuidad (modulo un umbral) de las funciones definidas sobre el soporte discreto de la imagen digital.

- Las características relativas a los segmentos como componentes del borde $b_\alpha := \partial r_\alpha$ están asociadas a la supresión de las discontinuidades que afectan a pequeñas variaciones geométricas (agrupamiento de segmentos en poligonales, típicamente) de acuerdo con restricciones de proximidad o de adyacencia. Por ello, su extracción se basa en la discontinuidad (variaciones por encima de un umbral) de las funciones definidas sobre el soporte discreto de la imagen digital.

El *objetivo del Análisis de Imagen* es el agrupamiento de datos relativos a regiones casi-homogéneas y sus bordes en objetos con características radiométricas o geométricas predeterminadas. De este modo, se facilita la interpretación de los elementos estáticos en relación con cuestiones de reconocimiento. Para ello, se extienden las técnicas comentadas más arriba:

- Las regiones deben ser "depuradas" mediante técnicas de restauración de imagen que, al nivel más bajo, utilizan diferentes modelos de supresión del ruido o de propagación (en presencia de agujeros o deterioro del soporte en retransmisiones, p.e.). De este modo se garantiza un carácter casi-homogéneo (variabilidad por debajo de un umbral) para las regiones a extraer.

- Los bordes deben ser completados para delimitar el contorno de las regiones planares (siluetas), añadiendo los elementos que fueran necesarios. De este modo, se garantiza la continuidad en los bordes de las regiones extraídas.

La combinación de ambas herramientas genera elementos robustos que se almacenan en diferentes tipos de estructuras de datos.

El primer paso de cualquier tubería en visión por computador pasa por el procesamiento y análisis de imagen, que puede ser utilizado también como parte del procesamiento de video.

2.1.2. Procesamiento y análisis de vídeo

El *objetivo* del procesamiento de vídeo es la detección y extracción de elementos correspondientes al fondo (background, BG) y al primer plano (foreground, FG). Para una cámara fija la substracción de imágenes consecutivas permite identificar los elementos móviles correspondientes a las dos imágenes comparadas.

Si la cámara fuera móvil, como todos los objetos tienen un movimiento aparente, es necesario identificar los elementos que presentan un movimiento inercial, es decir, asociado al movimiento de la cámara. Para resolver este problema se adopta una estrategia de tipo incremental en la que, inicialmente, se supone que la velocidad de la cámara es constante (en fases posteriores se supone que la aceleración es constante o bien sigue un patrón lineal ó cuadrático). De acuerdo con las restricciones de la escena y la aplicación a desarrollar, supondremos que la cámara está fija.

Una vez detectados los elementos contenidos en FG, es necesario etiquetarles como regiones asociadas a jugadores en movimiento u otros elementos móviles en la escena. Estos últimos se suprimen mediante filtrado asociado al color o al tamaño. El output de esta fase de procesamiento de vídeo es una colección de regiones asociadas a cada jugador que deben ser "pegadas" de acuerdo con restricciones radiométricas (color de la ropa, p.e.) y geométricas (compatibilidad de bordes con las regiones detectadas). Asimismo, es necesario suprimir mediante filtros específicos los efectos de emborronamiento (blurr) que degradan la calidad de las figuras detectadas.

El *objetivo del Análisis de Vídeo* es la complección de las figuras (supresión de agujeros y continuidad en los bordes asociados a siluetas móviles) y seguimiento de los objetos planares móviles $b^{\beta}(t)$ correspondientes a los jugadores contenidos en el FG.

Un análisis más avanzado utiliza modelos geométricos deformables (basados en snakes, típicamente) para facilitar el seguimiento de las siluetas móviles $s^{\beta}(t) := \partial b^{\beta}(t)$ (borde de objetos móviles). Para evitar una excesiva complejidad y facilitar el ajuste utilizando un número razonable (a lo sumo 5 por trozo, p.e.) de puntos de control, habitualmente se toman snakes dadas por trozos de curvas racionales "pesadas" de grado ≤ 4. Como los jugadores no llevan marcas ni sensores, estas curvas se pueden describir en términos de puntos significativos ó segmentos tangentes que se aproximan mediante secantes al "verdadero" contorno.

La generación automática de contornos basados en snakes y el seguimiento predictivo (basado en fuerzas normales y tangenciales a las siluetas) presenta una elevada complejidad computacional. Por ello, en este trabajo se ha adoptado una estrategia heurística que considera las siluetas (borde de los objetos) móviles a nivel icónico, es decir, sin asociar ningún tipo de trozos de snakes a las siluetas detectadas.

2.1.3. Segmentación espacio-temporal

La *segmentación espacial* es la correspondiente a cada imagen por separado; se ha descrito en la primera subsección.

La *segmentación temporal* es una descomposición de la secuencia de vídeo en una co-

lección finita de mini-secuencias que presentan un contenido similar. Por ello, el primer objetivo es la detección de *shots* (disparos) que "separan" mini-secuencias con contenido similar. La falta de semejanza se detecta a partir de variaciones en los estadísticos asociados al histograma (en la escala de grises, inicialmente) o a los histogramas para la escala de color (pudiendo dar peso a los colores que resultan más significativos).

La *segmentación espacio-temporal* se representa en términos de una superficie espacio-temporal asociada al seguimiento de los objetos de interés en cada una de las mini-secuencias de vídeo. Para ello, se introduce un modelo $2D + 1d$ en el que los datos $2D$ están asociados a cada imagen y $1d$ representa la línea del tiempo. En este caso, cada silueta móvil $s^\beta(t)$ está contenida en la sección temporal (slice) asociada al instante t.

El enlazado entre cada par de siluetas móviles consecutivas da lugar a una *superficie espacio-temporal* S^β que acota el objeto sólido B^β en movimiento a lo largo de la (mini)secuencia de vídeo, es decir, $S^\beta := \partial B^\beta$. La restauración superficial asociada al borde del objeto descripción proporciona un soporte continuo que facilita el seguimiento de los objetos.

En un contexto más avanzado, es necesario especificar los modelos de propagación que permiten "restaurar" el soporte de las superficies. Para ello, se toman modelos parabólicos de propagación que permiten incorporar de forma natural fenómenos de difusión a lo largo del tiempo sobre las superficies espacio-temporales $S^\beta(t)$ asociadas a los objetos móviles. Estas superficies se modelan en términos de B-splines dadas como producto de curvas racionales pesadas (splines) de grado bajo. Debido a la elevada complejidad computacional asociada al cálculo del soporte (superficies B-splines) y de los fenómenos de propagación correspondientes (difusión-reacción), en este trabajo no se han desarrollado dichos modelos. Además, aún no se dispone de herramientas eficientes para su reconocimiento automático en términos de alguna variante de redes neuronales. Por ello, este tipo de desarrollos podrían ser incluidos en desarrollos posteriores del modelo presentado.

2.2. Deep Learning (CNNs)

El aprendizaje profundo, más conocido como *Deep Learning* es una subrama del aprendizaje máquina o *Machine Learning* enfocado principalmente al uso de redes neuronales profundas, es decir, que usan un número mayor o igual que 4 de capas ocultas o intermedias *(hidden layers)*.

Los algoritmos basados en *Deep Learning* han ganado mucha importancia en los últimos años en campos como la visión por computador, el reconocimiento del habla o la traducción automática, superando con creces a las implementaciones previas. La figura 2.2 muestra una estimación de la diferencia de rendimiento cuando tenemos datos suficientes de las técnicas basadas en *Deep Learning* respecto a los algoritmos tradicionales de ML.

El concepto de *Deep Learning*, abarca muchos tipos de algoritmos pero en este trabajo nos centraremos principalmente en uno en concreto, las redes neuronales convolucionales *(Convolutional Neural Networks o CNNs)*.

Una red neuronal convolucional simple es una secuencia de capas donde cada capa

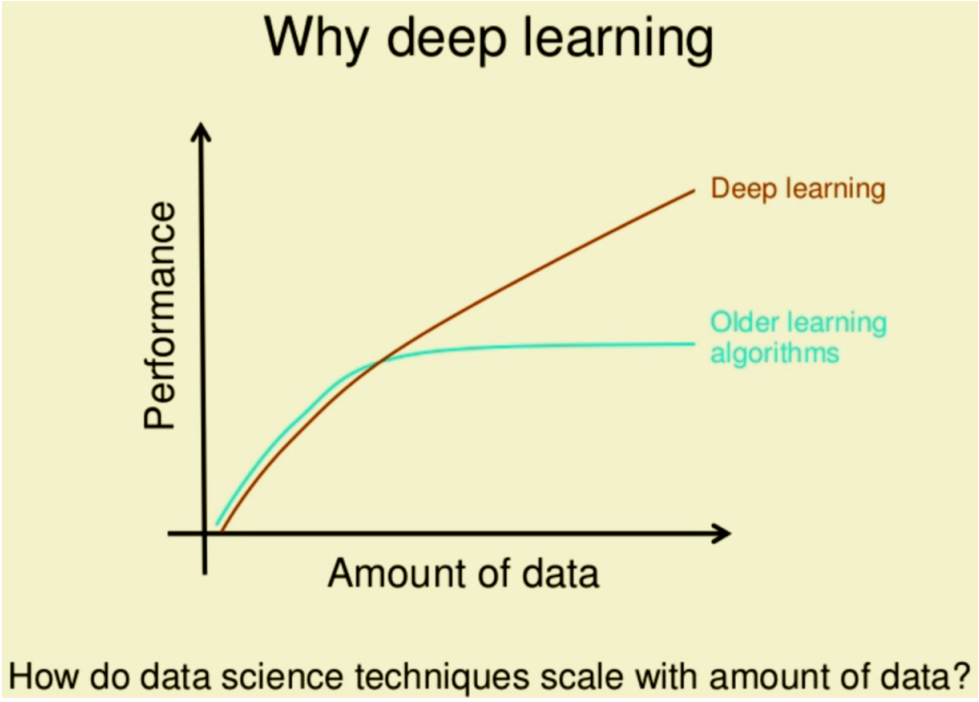

Figura 2.2: Rendimiento del *Deep Learning* y del *Machine Learning* en función de los datos

transforma un volumen de activaciones en otro a través de una función que inicialmente suponemos diferenciable. Para construir la arquitectura de una red convolucional principalmente se usan tres tipos de capas: las capas convolucionales *Convolutional Layer*, capas de reducción *Pooling Layer* y las capas totalmente conectadas *Fully Connected layer*. Esta último tipo de capa es exactamente igual al que usaríamos en una red neuronal artificial tradicional o en un perceptrón multicapa. Apilando estos tipos de capas construiremos la arquitectura de nuestra red convolucional.

Convolutional Layer: Las capas convolucionales son la clave de las redes convolucionales. Los parámetros iniciales de las capas convolucionales consisten en datos relativos a la conexividad y los pesos en las células básicas. Sobre esta arquitectura actúa un conjunto de filtros que se pueden aprender. Intuitivamente, la red aprenderá los filtros que se activan cuando ven algún tipo de característica visual, como un borde en alguna orientación, una mancha de algún color en la primera capa, o eventualmente patrones complejos en forma de rueda o panal en las capas internas de la red.

De este modo, tendremos un conjunto finito de filtros en cada capa convolucional (por ejemplo, 12 filtros), y cada uno de ellos producirá un mapa de activación bidimensional por separado. Apilando estos mapas de activación a lo largo de la dimensión correspondiente a la profundidad de la red se genera el volumen de salida. Este tipo de capas hacen a las redes convolucionales ideales para los problemas de Visión Computacional.

Pooling Layer: Con el objetivo de reducir el número de parámetros, es común incluir una capa de reducción (*Pooling Layer*) entre capas convolucionales consecutivas. Su objetivo es reducir sucesivamente la cantidad de parámetros para agilizar el cálculo de la red y por lo tanto también para reducir el sobreajuste *Overfitting*

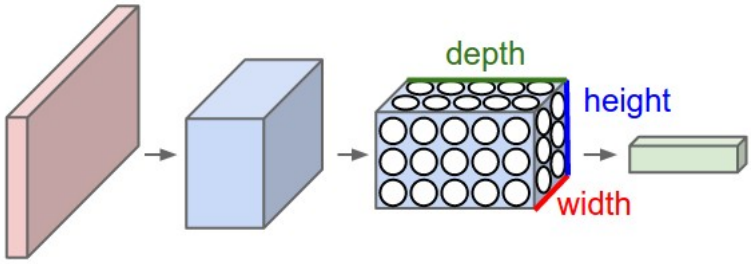

Figura 2.3: Componentes de una Red Convolucional
Una Red Convolucional organiza sus neuronas en tres dimensiones (ancho, alto, profundidad). Cada capa de una Red Convolucional transforma el tensor de entrada en un tensor de salida de activaciones neuronales. En esta figura, la capa de entrada roja contiene la imagen, por lo que su ancho y alto serían las dimensiones de la imagen, y la profundidad sería 3 (canales rojo, verde, azul).

Fully Connected layer: El ultimo tipo de capas que solemos utilizar en las redes convolucionaes son las capas totalmente conectadas donde todas las neuronas tienen conexiones con las activaciones producidas en las anteriores capas, este tipo de capa es similar al que podemos encontrar en las redes neuronales tradicionales.

Las primeras implementaciones prácticas de las redes neuronales convolucionales datan de los años 90, como por ejemplo la red que Yann LeCun y sus colaboradores construyeron para detectar digitos [1]; pero fue a partir del año 2012 cuando este tipo de tecnologías empezaron a estar omnipresentes en la visión por computador. En el año 2012, una red convolucional (Alexnet) [7] consiguió ganar el reto de ImageNet[4] y desde entonces, año tras año, los ganadores de este reto fueron redes convolucionales cada vez más profundas, con más capas y obteniendo mejores resultados.

Aunque las redes convolucionales fueron inventadas en los años 90, no fue hasta 2012 cuando realmente se empezaron a popularizar. Este cambio de paradigma se produjo principalmente debido a dos innovaciones:

- La primera de estas innovaciones fue el aumento de la capacidad de computación tanto en CPUs tradicionales como en GPUs. Estas últimas fueron la clave para implementar CNNs ya que permitieron entrenar modelos más profundos dentro de un tiempo razonable.
- La segunda de estas innovaciones fue la cantidad de datos disponibles. Estos algoritmos para funcionar bien y generalizar necesitan una gran cantidad de imágenes etiquetadas y en los años 90 simplemente no existía esta cantidad de datos disponibles, ni la capacidad para transportarlos y almacenarlos.

En el universo de las redes convolucionales podemos encontrar diferentes tipos de arquitecturas pero en este trabajo nos centraremos, principalmente, en dos arquitecturas:

- ResNet [10]
- Faster R-CNN [9, 11].

2.2.1. ResNet: *Deep Residual Networks for Image Recognition*

Las redes neuronales residuales (*Residual Networks*) son un tipo de redes neuronales que facilitan la construcción y el entrenamiento de redes profundas (e.g. 101 o 152 capas) introduciendo conexiones residuales con capas previas.

Resnet[10] fue introducida por Microsoft en el año 2015, ganando la competición Imagenet[4] de ese año. Actualmente es la implementación de referencia de este tipo de redes para el reconocimiento de imagen. La figura 2.4 muestra un ejemplo de ResNet con 34 capas.

2.2.2. Faster R-CNN

En la detección de objetos el objetivo es detectar la región de interés en la imagen mediante una *bounding box*. Suele hacerse en dos pasos: 1) proposición de regiones candidatas; 2) clasificación de estas regiones en fondo u objeto, y en este caso en la correspondiente categoría del objeto.

El principal problema de este enfoque simple (generación de candidatos + clasificación) es que los objetos de interés pueden tener diferentes ubicaciones, diferentes tamaños y diferentes *aspect ratios*. Por lo tanto, tendríamos que seleccionar un inmenso número de regiones como propuestas para poder detectar los objetos con buena precisión lo que computacionalmente se hace rápidamente inabordable o, en caso contrario, nos obliga a perder precisión. Para evitar este problema algoritmos como R-CNN, Yolo, o el propio Faster R-CNN han emergido durante los últimos años.

Faster R-CNN es una arquitectura para la detección de objetos, que incluye en una única arquitectura tanto la generación de propuestas de regiones como la clasificación de estas regiones (véase figura 2.5). En Faster R-CNN la generación de las propuestas corre a cargo de la RPN (*Region Proposal Network*) mientas que la clasificación del objeto la realiza la *Detection Network*. La RPN es implementada usando una CNN, que suele ser ResNet, pero que podría estar implementada por otras arquitecturas como VGG, Inception o RexNet. La salida de esta CNN es compartida con la parte de detección (*detection network*).

2.3. Visión por computador y Big Data

Dada la importancia que en el presente Máster se ha dado al concepto de Big Data, hemos querido reseñar la relación entre la visión por computador y el Big Data.

Los algoritmos más relevantes de Reconocimiento en Visión Computacional utilizan técnicas de *Machine Learning* y necesitan ser entrenados a partir de un número elevado de muestras. El entrenamiento puede ser supervisado o no. La disponibilidad de una colección de muestras correctamente etiquetadas facilita el proceso de aprendizaje supervisado. En particular y como hemos visto en secciones anteriores, los algoritmos basados en *deep learning* necesitan ser entrenados con muchos datos y durante un tiempo considerable

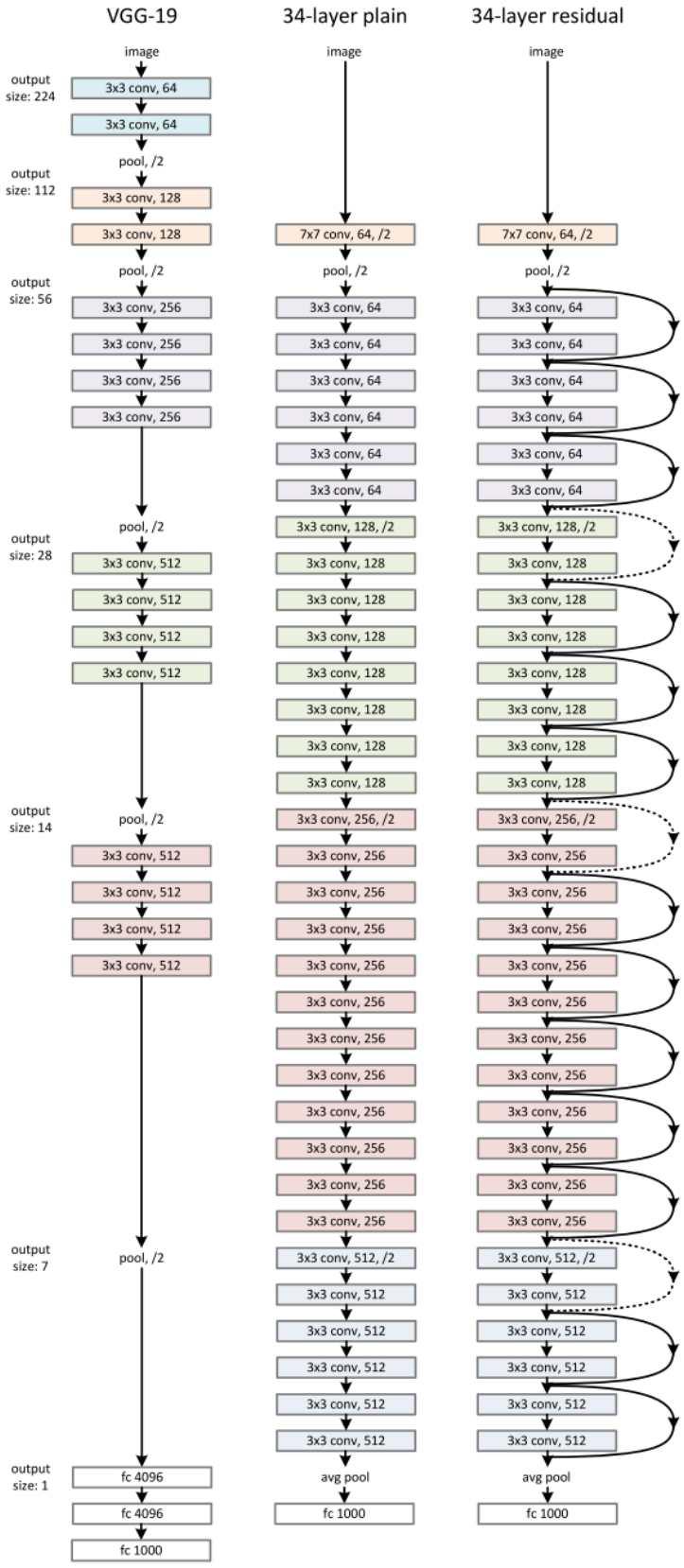

Figura 2.4: Arquitectura ResNet 34 capas

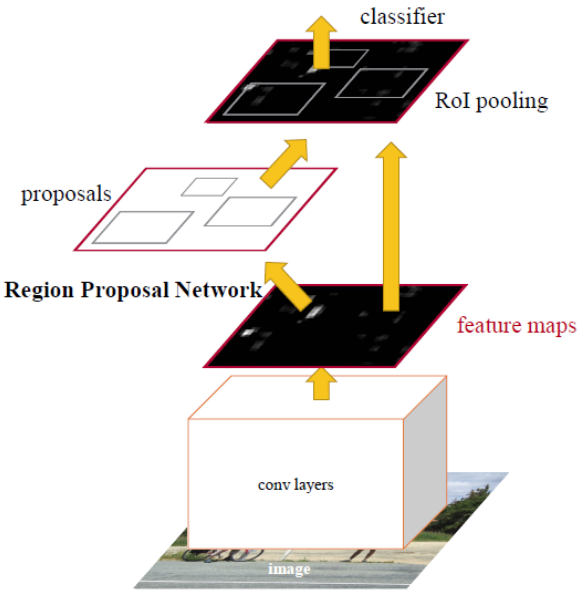

Figura 2.5: Arquitectura Faster R-CNN

(desde varias horas a varios días) para conseguir el rendimiento esperado en el subconjunto de validación.

En el aprendizaje máquina es común dividir nuestro *dataset* en tres fases: entrenamiento, validación y test.

1. el conjunto de entrenamiento se utiliza para ajustar de manera automática e iterativa los parámetros de nuestro modelo.

2. el conjunto de validación se utiliza para seleccionar los mejor hiperparámetros, variables que debemos establecer antes de aplicar un algoritmo de aprendizaje a un conjunto de datos, para nuestros datos.

3. el conjunto de test lo utilizamos, una vez entrenado nuestro mejor modelo, para dar una estimación del rendimiento real del modelo ante datos nuevos (con los que nunca hemos entrenado ni validado).

Para entrenar estos algoritmos necesitamos utilizar grandes volúmenes de datos, preprocesarlos, almacenarlos y por último utilizarlos en relación con jerarquías o restricciones, eventualmente estructurales. En estos momentos algunos de los datasets públicos disponibles a día de hoy ya cuentan con varios millones de imágenes. Distribuir los entrenamientos entre diferentes computadoras y GPUS se ha convertido en esencial para alcanzar resultados razonables en unos pocos días. Lamentablemente, la mayor parte de estos datos no están estructurados y se confía ciegamente en una convergencia que sería mucho más eficiente si se incorporaran restricciones estructurales asociadas al movimiento.

2.4. Soluciones comerciales existentes en el mercado

Aunque es complicado encontrar soluciones que se centren exactamente en el mismo problema que este trabajo, existen algunas empresas y herramientas enfocadas en campos cercanos. En términos funcionales, podemos agruparlas en los siguientes conjuntos:

- Estadísticas deportivas.
- Ayudantes de entrenador con análisis de vídeo.
- Herramientas de visualización de estadísticas deportivas.
- Apoyo al arbitraje
- Extracción automática de datos

A continuación, explicaremos qué ofrecen cada una de ellas, mencionando además algunas de las herramientas existentes.

2.4.1. Estadísticas deportivas

El primer conjunto de herramientas que comentaremos son las que se centran en vender estadísticas deportivas o en vender el servicio de etiquetado a partir de vídeos para diferentes deportes. La idea de estas herramientas es muy similar a la presentada en este trabajo. Por lo general, ninguna de las herramientas revisadas menciona el uso de herramientas automáticas para extraer información.

Un ejemplo de este tipo de servicios es Hudl Assist[24], ofrecido por la empresa Hudl para etiquetado de vídeos deportivos a la carta. El cliente envía un vídeo y la empresa se encarga de extraer la información relevante del mismo.

La mayor pega que podríamos encontrar a este tipo de servicios es su dificultad para ofrecer información a gran escala con un coste bajo; ya que parece que los flujos usados para extraer información son bastante manuales. Otra debilidad que cabe destacar en este tipo de soluciones podría ser la dificultad para ofrecer estadísticas en tiempo real. Sin duda, este tipo de servicios podrían verse complementados con soluciones como la que presentamos.

2.4.2. Ayudantes de entrenador con análisis de vídeo

El segundo conjunto de soluciones, relacionadas con la solución propuesta en este trabajo, son las herramientas de análisis de vídeo. Estas herramientas tienen dos funciones básicas: facilitar la labor de etiquetado en vídeos y ayudar en la edición de vídeos deportivos. Entre otras cosas, permiten anotar qué ha pasado en un determinado instante, dibujar sobre el vídeo, añadir comentarios y ofrecer capacidades de zoom y repetición a cámara lenta.

De este tipo de aplicaciones encontramos en el mercado varios ejemplos:

- NacSport [30]
- Sportscode [26]
- Hudl [25]
- Krossover [27]
- Coach's Eye [20]
- VidSwap [32]
- XOS Digital [23]
- Performa Sports [29]
- Dartfish[22] (utilizada por la Federación Española de Bádminton [21]).

A diferencia de la solución propuesta, este tipo de herramientas están mas centradas en ayudar a documentar y en revisar lo que pasó durante el juego, en lugar de extraer información de manera automática. Igual que para el grupo anterior, estas aplicaciones necesitan bastante trabajo manual lo que dificultaría realizar análisis a gran escala.

2.4.3. Herramientas de visualización de estadísticas deportivas

El siguiente tipo de aplicaciones que revisaremos están centradas en presentar la información con el objetivo de facilitar la obtención de conclusiones. Básicamente son herramientas de visualización de estadísticas centradas en el deporte. Estas presentan de forma útil estadísticas o datos tomados previamente.

Un ejemplo de este tipo de herramientas podría ser Dashboard[31], ofrecida por la empresa RII Sports Technology para visualizar datos y estadísticas relativas a datos como la posición de los jugadores, posesión y marcadores en partidos de fútbol americano de una forma útil.

2.4.4. Apoyo al arbitraje

Las herramientas de apoyo al arbitraje son, posiblemente, las más famosas dentro del mundo del deporte ya que se usan activamente en el mundo del tenis y también en el fútbol. La principal función es ayudar al árbitro a decidir sobre acciones dudosas del juego, como por ejemplo, la detección de goles fantasma o la decisión de si un golpe entró en la pista.

En el tenis, existen dos soluciones que usan visión computacional, el Ojo de Halcón[28] y Foxten In&Out[19], ambas reguladas por la Federación Internacional de Tenis. Sin lugar a dudas la más famosa de estas aplicaciones es el Ojo de Halcón, que se ha convertido en imprescindible en todos los torneos de alto nivel, hasta el punto de ser parte del espectáculo. Por otra parte Foxtenn In&Out, desarrollado por la empresa española Foxtenn,

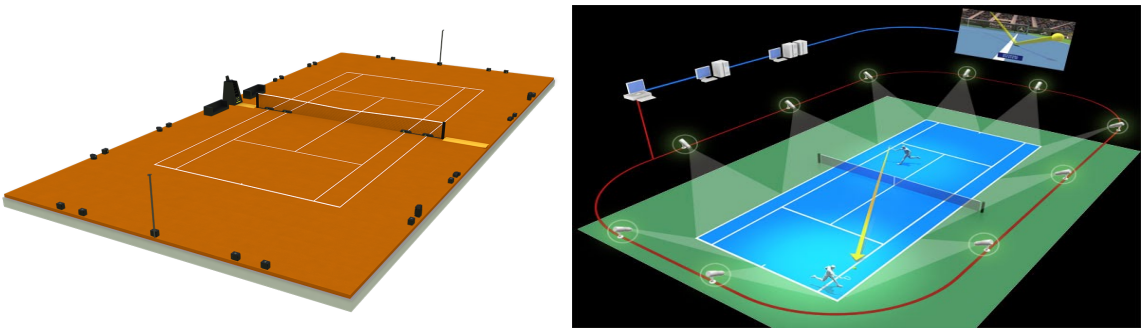

Figura 2.6: Foxtenn In&Out y Ojo de Halcón (Hawk-Eye)

promete más precisión y también se está empezando a implantar en torneos de máximo nivel.

El Ojo de Halcón, ver figura 2.6, está compuesto por 10 cámaras suspendidas a cierta altura sobre la pista de tenis. Estas diez cámaras son capaces de obtener unas 150 imágenes por segundo cada una. Combinando las imágenes de las 10 cámaras el sistema consigue estimar la trayectoria del golpeo con una precisión de 3mm. Por su parte, Foxten In&Out (véase Figura 2.6) utiliza 22 cámaras en partidos de individuales y 44 cámaras en partidos de dobles más 10 láseres de alta precisión. Estas cámaras, situadas directamente sobre la pista, recogen globalmente 150000 imágenes por segundo y consiguen obtener el bote real de la pelota; en lugar de una simulación como en el caso del Ojo de Halcón.

Estos sistemas son pioneros en su campo y tienen una efectividad demostrada que cumple su objetivo. Su mayor inconveniente es su coste, aunque se compensa con su elevada precisión.

2.4.5. Extracción automática de datos

Finalmente, el último grupo de aplicaciones que comentaremos tienen un objetivo muy parecido al propuesto en este trabajo. su objetivo es extraer información a partir de cámaras y otro tipo de sensores para generar indicadores, momentos claves del encuentro, puntos de golpeo, etc.

Un ejemplo, de este tipo de herramientas es Foxtenn Diamond Data System[18, 17] ofrecida por la empresa española Foxtenn para el tenis (véase Figura 2.6). Igual que para el caso de Foxten In&Out mencionado en la sección anterior, el sistema usa 22 o 44 cámaras y 10 láseres que consiguen extraer toda la información de un partido de Tenis. Durante el partido, el sistema permite usar los datos en tiempo real para sacar estadísticas del encuentro o para mejorar las retransmisiones televisivas. Además, después del partido, estos datos pueden ser usados para enviar un informe personalizado al jugador.

La solución planteada por Foxtenn es muy similar a la propuesta en este documento; la diferencia con respecto a la solución presentada en este trabajo es la versatilidad y el coste reducido. Foxtenn extrae datos útiles a partir de la información obtenida usando un sistema propio de cámaras, láseres y sensores que deben ser instaladas en cada pista. Instalar este sistema es algo inviable para torneos medianos o pequeños. Nuestra aproxi-

mación es menos precisa pero permite extraer la información a partir de los vídeos de las retransmisiones deportivas que ya están disponibles. Esto reduce el coste y hace la solución más escalable ya que permitiría tener una mayor cobertura de jugadores, partidos y torneos.

Capítulo 3

Desarrollos

A lo largo de este capítulo se introducirá la aproximación para conseguir el objetivo principal de este trabajo: extraer información a partir del procesamiento de vídeo-secuencias de bádminton.

3.1. Entendiendo un partido de bádminton

Extraer información de los eventos y las acciones transcurridos durante una secuencia de vídeo es una tarea aparentemente fácil para una persona pues se realiza casi de manera inconsciente. Sin embargo, realmente realizamos automáticamente muchas subtareas que es necesario especificar para un sistema. Estas subtareas son las que debemos modelar, diseñar e implementar mediante algoritmos que puedan ser incorporados a una aplicación como la que se propone en este trabajo.

El marco general para abordar este problema es el de cualquier proceso de reconocimiento. Los elementos básicos de esta estrategia incluyen la detección, descripción y clasificación de características cinemáticas asociadas a eventos en el vídeo. Debido a las limitaciones en el desarrollo actual para el análisis de este tipo de secuencias de vídeo, sólo se dispone de herramientas eficientes para la primera fase del proceso citado, es decir, para la detección. Por ello, nos hemos centrado en esta fase.

Esta sección presenta los pasos y actividades que el proceso de análisis debería realizar para extraer la información relevante de un partido de bádminton. Este proceso queda ilustrado en el diagrama de la Figura 3.1. Con el objetivo de acotar el trabajo, nos centraremos en escenas grabadas bajo las siguientes condiciones:

- **Cámara fija:** la posición de la cámara es fija durante todo el desarrollo del juego. Esta cámara encuadrará la pista completa dentro de su campo de vista en una posición sagital.

- **Resolución:** Los vídeos deberán tener una resolución aceptable, de al menos 1280*720 píxeles.

- **Iluminación:** La escena estará iluminada uniformemente, proyectando sombras solo en los jugadores.

El flujo presentado para el análisis está enfocado al Bádminton, pero, mediante algunas adaptaciones, podría ser aplicado a otros deportes de red como el Tenis o el Voleibol, pues las hipótesis hechas son válidas también para estos deportes. Obviamente, sería necesario identificar los eventos y re-entrenar los modelos para cada deporte; asimismo, habría que adaptar ciertos parámetros, como las medidas de la pista, la altura de la red y las reglas del juego. El diagrama de la Figura 3.1, inspirado en [3], muestra los principales tareas que un algoritmo como el introducido en este documento debería realizar. En verde se resaltan los pasos que se abordarán (completa o parcialmente) en este trabajo, mientras que en amarillo se representan las tareas no abordadas en este trabajo.

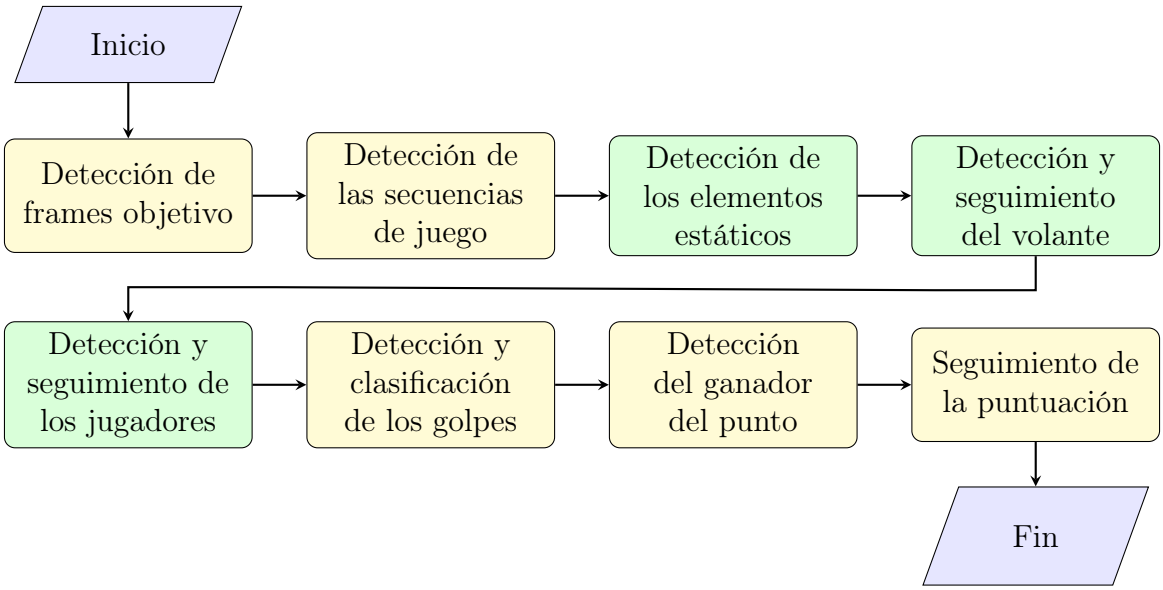

Figura 3.1: Diagrama propuesto para analizar un partido de bádminton

Cada una de estas etapas del flujo será introducida y explicada en posteriores subsecciones, aunque nos centraremos en las partes abordadas durante este trabajo. Para cada de ellas explicaremos los retos y dificultades, la utilidad, la entrada esperada y la salida esperada del paso en concreto. También comentaremos los experimentos realizados, cuyos resultados y conclusiones se comentan en los siguientes capítulos.

3.2. Procesamiento y análisis de vídeo

En esta sección comentaremos las tareas relacionadas con el procesamiento y el análisis del vídeo que se utiliza como fuente de información en nuestro sistema.

Figura 3.2: Ejemplos positivos y negativos de frames objetivo para el *pipeline* propuesto

3.2.1. Detección de frames objetivo

El primer paso de nuestro *pipeline* consiste en detectar los frames correspondientes al plano de cámara que nos interesa y que usaremos durante el resto del flujo diseñado.

Durante una retransmisión de un partido de Bádminton profesional es muy común que el realizador esté constantemente efectuando cambios de plano para enfocar al jugador después de un punto, para enfocar al árbitro después de una decisión dudosa o incluso durante el transcurso del punto para hacer más espectacular la retransmisión para los espectadores. Estos cambios de plano son realmente un problema para nuestro algoritmo, pues suponemos una cámara fija posicionada en el fondo de la pista.

La figura 3.2, muestra ejemplos negativos y positivos del tipo de frames que nos interesan para el resto de los pasos del flujo presentado. En la fila superior se muestran los frames buscados en cada vídeo, capturados por una cámara fija al fondo de la pista. En la fila inferior se muestran los frames que serán descartados.

Buscamos utilizar los frames en los que además de tener una cámara fija, podamos observar toda la pista. Descartaremos los frames que muestran al público, primeros planos de los entrenadores o primeros planos de los jugadores. En resumen, utilizamos los frames tomados desde el fondo de la pista porque:

1. Desde el fondo de la pista podemos apreciar todos los eventos que ocurren en la pista.
2. Podemos utilizar las líneas de la pista, la ubicación de los postes y la ubicación de la red como elementos de referencia y así calcular la ubicación 3D de los jugadores a partir de su posición en la imagen (aunque la parte de reconstrucción 3D queda fuera del alcance de este trabajo).
3. Este tipo de frames son los más comunes en las retransmisiones de Bádminton.

Esta etapa podría verse como un problema de clasificación de frames con dos categorías: frame objetivo o frame no objetivo. Este recibirá como entrada vídeos completos y devolverá, como resultado, el listado de frames correspondientes a la cámara fija en el fondo de la pista.

En este trabajo la selección de frames significativos se ha realizado de forma manual por lo que este paso forma parte de los desarrollos futuros.

3.2.2. Detección de las secuencias de juego

El segundo paso de nuestro *pipeline* consiste en detectar las secuencias de juego, es decir, el momento en el que un punto empieza y el momento en el que acaba. En bádminton un punto siempre empieza con el saque, normalmente con los jugadores en la zona central de la pista, y termina cuando uno de los jugadores o parejas (si es un partido de dobles) consigue el punto.

Este paso es necesario, ya que en las retransmisiones deportivas no todo el tiempo de vídeo se corresponde con juego real; tenemos momentos donde los jugadores están parados, celebrando haber conseguido un punto o esperando para empezar el siguiente punto. En este trabajo sólo nos interesan las secuencias de juego efectivo.

Por lo tanto, esta tarea recibirá como entrada un vídeo y devolverá como salida los diferentes puntos (secuencias reales de juego) descritos por el momento o frame que da inicio al punto y por el frame en el que el punto concluye. En este trabajo la detección de las secuencias de juego se ha realizado de forma manual y por lo tanto su implementación forma parte de los desarrollos futuros.

La detección automática de esta secuencia puede plantearse como una segmentación temporal, es decir, una descomposición de la secuencia de vídeo en minisecuencias separadas por eventos instantáneos. Cada evento instantáneo está marcado por el saque y la respuesta por parte de cada jugador. Por ello, cada evento separa el tiempo de vuelo del volante correspondiente a la jugada propiamente dicha y los tiempos de espera anteriores a la realización del juego.

Para llevar a cabo la segmentación de forma automática se utiliza una doble aproximación basada en propiedades radiométricas (discriminación basada en color de las camisetas de los jugadores, p.e.) y cinemáticas (asociadas a gestos una vez realizada la segmentación con respecto al fondo, es decir, evolución de posturas típicas, basadas en siluetas, correspondientes a las jugadas más relevantes). Ambos procesos deben ser entendidos de forma consecutiva y pueden incluir post-procesamiento debido al ruido (para las propiedades radiométricas) y al emborronamiento ó blurr (para las propiedades cinemáticas).

Un análisis más fino off-line puede requerir técnicas de restauración de imagen (para propiedades radiométricas y extracción de siluetas) ó de secuencias de vídeo en relación con la detección de bordes ∂r_β de las regiones de interés (ROI) r_β asociadas a siluetas $s_\beta(t) = \partial r_\beta(t)$ en movimiento. La detección de los bordes a bajo nivel se lleva a cabo en términos de envolventes visuales (visual hulls); como presentan discontinuidades, es necesario enlazarlas ateniendo a criterios de búsqueda radial en cada discontinuidad.

Existen modelos cinemáticos más sofisticados que utilizan snakes (trozos curvas racionales con pesos relativos) que son la extensión natural de los PL-modelos utilizando para las envolventes visuales mencionadas más arriba. Estos procedimientos son de gran utilidad para una reconstrucción de la cinemática, pero aún no se dispone de un ajuste automático suficientemente fino que tenga en cuenta la cinemática asociada a las carac-

terísticas de los movimientos de los jugadores. Por ello, en este trabajo hemos preferido desarrollar una aproximación más simple basada en envolventes visuales.

3.2.3. Detección de los elementos estáticos

El siguiente paso consiste en detectar los elementos estáticos de la pista de bádminton. Por elementos estáticos nos referimos a los elementos que no cambiarán de posición durante todo el partido: las líneas de la pista, los postes y la red. Dedicamos una sección a la primera y otra a los demás elementos.

La pista

El objetivo de la detección de la pista, es segmentar la imagen en términos de las líneas de la pista, es decir, buscamos extraer las líneas correspondientes a la pista de bádminton. En este paso queremos tener una buena precisión en cuanto a la ubicación de las líneas detectadas. Por ello, un simple rectángulo que incluya la pista no es suficiente, pues queremos la localización casi exacta de las líneas en la imagen; necesitaremos usar una aproximación que permita obtener de forma automática las líneas con precisión métrica.

Además, la detección de las líneas de la pista tiene varios **retos y dificultades** (vease la Figura 3.3):

- Diferentes colores de pista.
- Líneas borrosas o difuminadas.
- Distorsiones introducidas por las cámaras.
- Oclusiones parciales de las líneas como las provocadas por los jugadores durante el transcurso del punto.
- Diferentes relaciones de aspecto provocadas por la ubicación de la cámara.
- Líneas fantasmas que aparecen en la grada o en otros elementos de la composición.

Para detectar las líneas de la pista sacaremos partido de las propiedades radiométricas y geométricas del campo de bádminton en las imágenes. En eventos de alto nivel, por reglamento, las líneas son siempre blancas y la disposición geométrica de las líneas es conocida y es siempre la misma. Usaremos el color blanco de las líneas para binarizar la imagen y eliminar líneas o artefactos en otros colores para aumentar la precisión del algoritmo de detección de líneas. En una imagen en escala de grises un píxel puede tomar valores entre 0 y 255, donde el 0 es negro y el 255 representaría el color blanco; entonces, si sobre una imagen en escala de grises aplicamos un fuerte umbral en el que eliminamos (dejamos con valor 0) los píxeles menores que 200 conseguiremos los píxeles más blancos de la imagen (véase Figura 3.4).

Como solución a la detección de líneas se proponen tres alternativas. La primera utiliza algoritmos de procesamiento de imagen mientras que las dos siguientes están basadas en

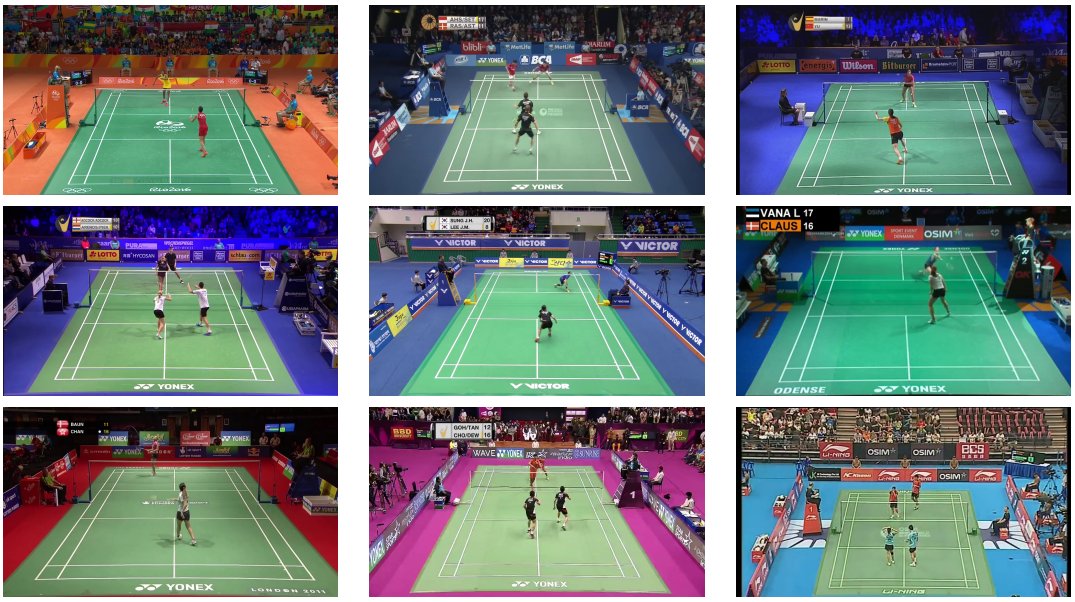

Figura 3.3: El conjunto de imágenes mostradas presenta varios de los retos implícitos en la detección de la pista como: las oclusiones, las diferentes perspectivas o las líneas borrosas o difuminadas.

Figura 3.4: Umbralizado binario (>200) para facilitar la detección de las líneas de la pista

deep learning. Todos los procedimientos detectarán la pista y sus líneas buscando sus 4 esquinas exteriores en el frame objetivo, la diferencia entre los procedimientos está en el modo de buscar estas cuatro esquinas. Dado que las dimensiones de una pista de bádminton son siempre las mismas, una vez detectadas las cuatro esquinas podemos calcular la homografía (transformación de perspectiva entre dos planos) entre las esquinas de la pista y las esquinas en un patrón de la pista (ver figura 3.5). Finalmente usando esta matriz de transformación, proyectaremos las líneas de la pista de bádminton sobre el frame que estamos analizando.

Este algoritmo podría funcionar tanto en situaciones de cámara fija como en situaciones de cámara móvil. Como en este trabajo suponemos que se tiene una cámara fija, podríamos ejecutar este algoritmo una única vez por partido o vídeo. Para el caso de cámara móvil deberíamos ejecutar este algoritmo en todos los frames o en un subconjunto de ellos en función de la localización (posición y orientación) cambiante de la cámara móvil.

La solución usando algoritmos de procesamiento de imagen está basada en el algoritmo propuesto en [2], sobre el que cual hemos realizado algunos cambios para mejorar su eficiencia. Se detalla en el siguiente fragmento en pseudocódigo:

```
1 img = leer_imagen()
```

Figura 3.5: En la imagen de la izquierda (patrón de la pista) tenemos los puntos A, B, C, D que utilizaremos junto a los puntos A', B', C', D' (de la imagen objetivo) para obtener la matriz $\boldsymbol{M}$ de transformación para la homografía buscada.

```
img_gray = transformar_imagen_a_escala_de_grises(img)

# Buscamos los píxeles en blanco, entre los que se incluyen las líneas
img_threshold = umbralizado_binario(img_gray)

# Buscamos líneas usando el algoritmo de Hough
lineas = hough_lineas(img_threshold)

# Buscamos intersecciones entre las líneas detectadas
intersecciones = intersecciones(lineas)

# Buscamos conjuntos de 4 candidatos para la homografía.
candidatos_h = combinaciones(intersecciones,4)

# Reducimos el conjunto de candidatos finales
candidatos_h = filtrar_candidatos(canditatos_h)

# Obtenemos los cuatro pares de puntos de la pista para la homografía.
pb = PistaBadminton()

# Buscamos las 4 esquinas de la pista en nuestra imagen
esquinas_reales_pista = pb.esquinas_exteriores()

# Para cada tupla de 4 puntos calculamos su homografía
# Cada homografía es evaluada con una distancia L2
mejor_cantidato = Null
mejor_cantidato_error = Null
for candidato_hgrf in candidatos_h:
  m = estimar_matriz_homografia(esquinas_reales_pista, candidato_hgrf)
  e = estimar_error_homografia(m, intersecciones, pb)
  if e < mejor_candidato_error:
    mejor_candidato_error = e
```

```
34      mejor_candidato
35  return mejor_candidato
```

Con respecto a este algoritmo, en el capítulo de resultados, además de mostrar algunos resultados analíticos, también mostramos ejemplos visuales para facilitar una mejor comprensión de los pasos seguidos.

La primera solución basada en deep learning consiste en entrenar un detector **Mask R-CNN** [16], una arquitectura que extiende Faster R-CNN para segmentar a nivel de píxel, además debe devolver los *bounding boxes* que envuelven los objetos. Se parte de la premisa de que una segmentación semántica a nivel de píxel conseguirá un ajuste de la pista mucho que mejor que un rectángulo ya que la perspectiva de la cámara deforma el aspecto de la pista.

La segunda solución basada en deep learning consiste en detectar directamente las esquinas de la pista. El planteamiento es similar: detectar las esquinas para después calcular la homografía y las líneas de la pista. Usando la arquitectura Faster R-CNN [9] detectaremos los vértices exteriores de la pista para posteriormente poder calcular la homografía. Faster R-CNN devuelve varios rectángulos entre los que presumiblemente estarán los 4 vértices de la pista. Para convertirlos en un sólo vértice seleccionaremos el punto medio de estos rectángulos detectados. Si Faster R-CNN encontrase más de 4 *bounding boxes* seleccionaremos los 4 resultados con mayor confianza.

Los postes y la red

Además de las líneas de pista, los postes y la red son elementos estáticos significativos para el análisis de secuencias de vídeo. Por ello, se les dedica una subsección específica.

Esta subsección describe los modelos y herramientas que permiten detectar los postes y la red en las secuencias de vídeo. Para la detección de los postes simplemente necesitaríamos el segmento vertical que parte desde el suelo y acaba en la red. En el caso de la red, detectar u obtener la línea horizontal que une los postes también sería suficiente. En este trabajo abordaremos el problema detectando los postes y obteniendo la línea de la red como el segmento que une ambos postes en su posición más alta.

La detección de los postes y la red también conlleva varios retos a los cuales es preciso dar una respuesta (véase la Figura 3.6):

- Diferentes tipos de poste, con diferentes formas y colores. Además no es raro que cada nuevo año o temporada los fabricantes introduzcan nuevos modelos. Por ello, nos restringimos a características geométricas ya estandarizadas.
- Distorsiones introducidas por las cámaras: Las más frecuentes corresponden a efectos de curvatura (tipo tonel o tipo cojín) sobre píxeles para los cuales debemos corregir la distorsión gracias a la calibración intrínseca de la cámara.
- Oclusiones parciales de las postes: Para resolverlos se aplica la información previa sobre la longitud de los postes, módulo la corrección métrica asociada a la localización de la cámara (corresponde a la calibración extrínseca de la cámara).

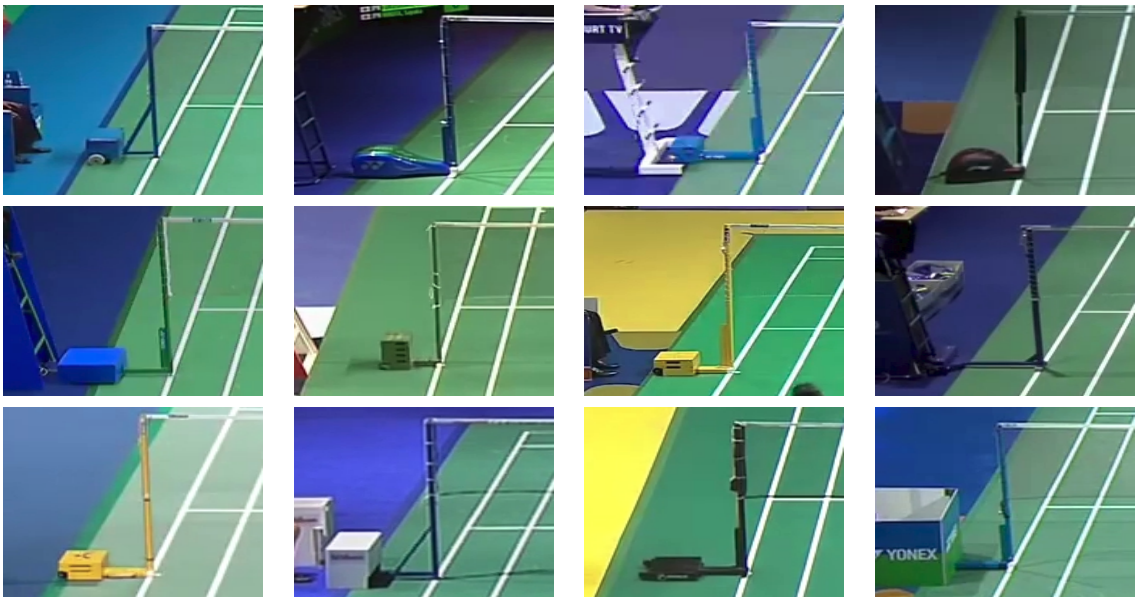

Figura 3.6: Los 12 postes de la ilustración muestran la variedad existente el mercado, con diferentes formas y colores.

- Diferentes relaciones de aspecto provocadas por la ubicación de la cámara. Este problema aparece asociado a localizaciones oblicuas de la cámara en relación con la escena; como habitualmente se utilizan cámaras con localización fronto-paralela o en un plano sagital, no consideramos localizaciones oblicuas (caso de haberlas, se resolverían mediante una homografía).

En este trabajo abordaremos la tarea de detectar los postes cómo un problema de deep learning en el que enseñaremos a un sistema a detectar los elementos de la pista. Más concretamente utilizaremos de nuevo *Faster R-CNN* [11].

Esta aproximación reduce sustancialmente el número de umbrales definidos manualmente, permite escalar la precisión de la detección en función de los datos y también es rápida. Además estos modelos tienen mayor precisión si en nuestro conjunto de entrenamiento contamos con suficientes datos, como veremos en el capítulo de resultados. Finalmente, la inferencia, ejecutar la red neuronal ya entrenada, puede llegar a hacerse en tiempo real utilizando una GPU.

Frente a esta aproximación, [2] propone resolver la detección de los postes y de la red utilizando técnicas de procesamiento de imagen convencionales. Propone afrontar la detección de postes después de detectar las líneas de la pista y, a partir de ellas, buscar segmentos verticales cercanos al centro de la pista aplicando un margen para acotar la región donde buscar.

Las principales ventajas de este método serían la velocidad, la sencillez y la ausencia de entrenamiento. Sin embargo, desde nuestro punto de vista, este procedimiento tiene tres inconvenientes, que resuelve nuestra aproximación:

1. Necesidad de identificar las líneas de la pista para localizar los segmentos verticales.

2. Dependencia de varios umbrales o *thresholds* establecidos manualmente, como por ejemplo el ángulo máximo para poder considerar un segmento como vertical o la región donde buscaremos segmentos verticales.

3. Uso intensivo de características radiométricas locales que son muy dependientes de la iluminación.

En la sección de resultados se mostrarán los resultados cuantitativos y cualitativos del procedimiento propuesto para la detección de los postes usando el conjunto de datos de evaluación.

3.2.4. Detección y seguimiento del volante

El siguiente paso de nuestro proceso será la detección y el seguimiento del volante de bádminton. En bádminton el volante es la "pelota" que los jugadores golpean de una pista a otra durante un punto.

Cómo en la mayoría de procedimientos de seguimiento de objetos, este paso podemos dividirlo a su vez en varias partes. Tendremos una etapa donde detectamos los objetos en un instante concreto, por ejemplo el frame $\boldsymbol{t}$; y, por otra parte, tendremos una o más etapas donde "enlazamos" esos objetos durante el tiempo. En estas últimas etapas, buscamos realizar un seguimiento a lo largo del tiempo de la trayectoria y posición de cada uno de los objetos, de manera inequívoca.

Detección

La primera tarea que desarrollaremos en esta etapa, es la detección del volante en cada uno de los frames de un punto. Buscaremos la ubicación del volante en base a sus propiedades radiométricas.

La detección del volante conlleva varios retos:

- Desaparición total del volante en algunos frames debido a la velocidad del movimiento o a oclusiones. En instantes donde la velocidad del volante es muy alta, como por ejemplo después de haber sido golpeado por un jugador, es habitual observar frames en los que es imposible para una cámara convencional (incluso para una persona) encontrar la posición del volante.

- Distorsiones introducidas por las cámaras, deformación de la forma del volante y emborronamiento (blurr). Cuando el volante está parado tiene una morfología conocida (triangulo invertido); pero cuando este es golpeado y por lo tanto se mueve a gran velocidad, es habitual, apreciar una mancha difuminada o emborronada que poco tiene que ver con la forma original del volante. Este emborronamiento también puede ser debido a discontinuidades en la iluminación o al ruido introducido por las propias cámaras.

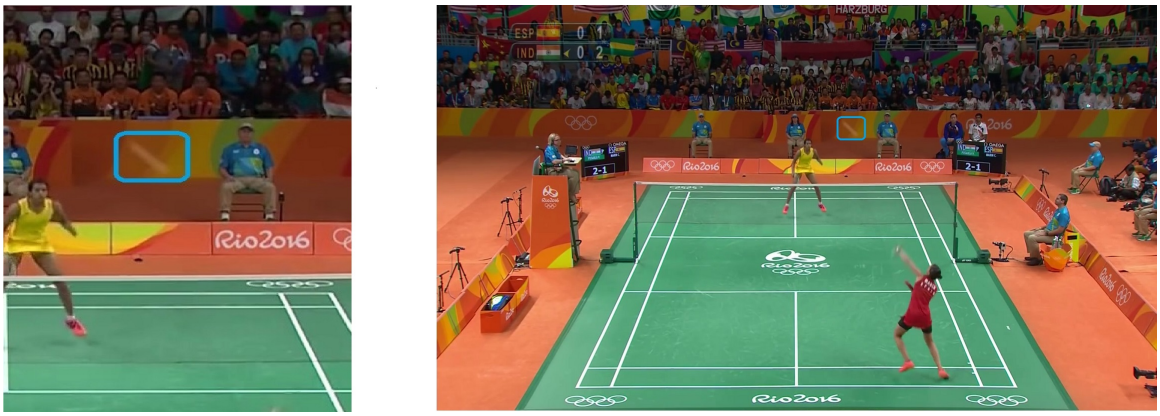

Figura 3.7: Después de un golpe, el volante se convierte en una estela alargada de color blanco. Este, es solo un ejemplo de los retos que conlleva la detección del volante.

En este trabajo abordaremos la tarea de detectar el volante, igual que en el caso de los postes, cómo un problema deep learning en el que enseñaremos a una red a detectar el volante. Para este enfoque, se desarrollarán tres alternativas o variantes.

La primera de estas variantes consistirá en usar *Faster R-CNN* para detectar el volante en frames aislados e independientes. Esta vía resulta la más fácil de desarrollar, ya no hace falta ningún trabajo de preprocesamiento, es decir, la entrada de la red es un frame aislado y la salida de la misma serán las detecciones del volante para la imagen dada. Por otra parte, también se platean dos variantes más aprovechando y utilizando el contexto temporal que aporta el vídeo.

Para entender estos dos nuevos planteamientos, partimos de una observación simple: para una persona es mucho más fácil encontrar el volante en el contexto de un vídeo que observando un frame o una imagen aislada ya que nuestro cerebro rápidamente detecta las cosas que se mueven entre diferentes instantes de un vídeo; sin embargo, si intentamos buscar el volante en imágenes aisladas, nos resultará mucho más difícil ya que este puede presentarse borroso, deformado y pequeño. El objetivo es, aprovechando que estamos trabajando con vídeo, intentar introducir este tipo de información temporal para ayudar a la red a aprender más rápidamente, con menos datos y con mejor rendimiento al detectar el volante en movimiento.

En relación con este último enfoque, se plantean dos experimentos en los que trataremos de detectar el volante en una nueva imagen artificial que construiremos concatenando la imagen en el instante actual con la imagen correspondiente a la resta del frame actual (t) con el frame anterior $(t-1)$, a la que llamaremos imagen diferencia. En el primero de estos experimentos concatenaremos la imagen diferencia en escala de grises (i.e. sólo un canal) y en el segundo experimento, concatenaremos la imagen actual con la imagen diferencia en RGB (i.e. 3 canales); es decir, generaremos imágenes artificiales de 4 y 6 canales respectivamente. Estas ideas tienen su origen en otras publicaciones en las que el autor de este documento ha participado [37] y [36].

En resumen, se realizarán tres experimentos:

1. Detección del volante en imágenes aisladas con *Faster R-CNN*

Figura 3.8: En la imagen se muestran ejemplos de las composiciones de cuatro y seis canales que usaremos para entrenar la detección del volante. Como puede apreciarse, en la imagen correspondiente a la resta de frames consecutivos es más fácil detectar la ubicación del volante.

2. Detección del volante en una imagen artificial generada mediante la concatenación del frame el instante t, con la imagen diferencia de los frames en los instantes t y $t-1$ en escala de grises (4 canales) con *Faster R-CNN* (veáse Figura 3.8).

3. Detección del volante en la imagen artificial generada mediante la concatenación del frame el instante t, con la imagen diferencia de los frames en los instantes t y $t-1$ en RGB (6 canales) con *Faster R-CNN* (véase Figura 3.8).

Dadas las propiedades radiométricas del volante podríamos utilizar métodos más sencillos ya que el volante de bádminton siempre es blanco. Por ello, haciendo una búsqueda por máximos de intensidad, también podríamos detectar el volante. En este trabajo, nos centramos en el enfoque basado en *deep learning* y dejamos la evaluación de esta alternativa para futuros desarrollos.

Enlazado de segmentos

La siguiente tarea para el seguimiento, será relacionar las regiones detectadas en imágenes consecutivas a lo largo del tiempo. En nuestro caso, estas regiones a conectar proceden del paso de detección anterior. Para cada frame $\boldsymbol{t}$ de un punto, primero ejecutaremos el procedimiento de detección y posteriormente relacionaremos esta detección, con la ocurrida en el frame $\boldsymbol{t-1}$ consiguiendo de esta manera estimar la trayectoria del volante.

En este trabajo la relación de las detecciones a lo largo del tiempo la realizaremos, mediante un simple enlazado de segmentos entre detecciones consecutivas del volante.

Los principales retos de esta parte son los siguientes:

- Detección de discontinuidades: por las razones comentadas anteriormente, en algunos instantes no habrá detecciones del volante; por lo tanto será necesario definir una estrategia para afrontar estos casos. En este trabajo si en el frame $\boldsymbol{t}$ no hemos detectado el volante, simplemente, pasaremos a enlazar el siguiente frame $\boldsymbol{t+1}$ directamente con el frame $\boldsymbol{t-1}$.
- Falsos positivos en la detección: es probable que en algunos instantes nuestro procedimiento de detección devuelva más de un resultado. En bádminton solo tenemos un volante al mismo tiempo en la pista y por lo tanto no puede estar a la vez en dos ubicaciones simultáneas; en este trabajo resolveremos este problema seleccionando la detección con mayor confianza. En desarrollos futuros se podría tratar de resolver este conflicto utilizando la región que mejor encaje con una trayectoria estimada.
- Detecciones muy grandes: debido a la velocidad, en algunas ocasiones, la región detectada corresponde con un rectángulo relativamente grande. En este trabajo seleccionaremos el punto medio de esta región.

Como trabajo futuro sería recomendable evaluar la utilización de redes neuronales recurrentes (RNN) para mejorar la detección y el seguimiento; aunque en este trabajo hemos decidido no incluirlas. Las RNN se pueden utilizar para incorporar la información temporal en relación con seguimiento y predicción. Por ello, presumiblemente son una buena alternativa para mejorar resultados en relación con el seguimiento del volante.

Estimación tosca de características cinemáticas

Se refieren a velocidad y acelaración (lineales y angulares) del volante. No se analizan debido a las irregularidades, a la falta de información sobre modelos cinemáticos subyacentes y a la elevada variabilidad dinámica en relación con las fases de saque, vuelo e impacto. Por ello, nos centremos en las dos primeras fases, dejando la tercera como posible ampliación de este trabajo.

3.2.5. Detección y seguimiento de los jugadores

El siguiente paso de nuestro proceso, y además el último que afrontaremos en este trabajo, será la detección y el seguimiento (*tracking*) de los jugadores de bádminton.

En el caso del seguimiento de jugadores podemos tener dos o cuatro jugadores simultáneamente; si el partido es individual o de dobles respectivamente; lo que hace este problema considerablemente más difícil que el seguimiento del volante donde sólo había un objeto en la escena. Comparando este problema con la detección y el seguimiento del volante, la dificultad reside en los idferentes patrones usados para la detección y seguimiento de los jugadores, tareas que requieren un mayor número de puntos de control para modelos más complejos.

El seguimiento de objetos móviles utiliza estrategias de refinamiento selectivo (coarse-to-fine) basadas en la utilización de propiedades radiométricas para la segmentación de imagen (bajo nivel), inicialización geométrica de los jugadores (medio nivel asociado a

posturas típicas) y seguimiento basado en características cinemáticas (alto nivel asociado a "gestos" que se modelan como una secuencia temporal de posturas).

Igual que para el seguimiento del volante, dividiremos esta etapa en varias partes. La primera centrada en la detección y las siguientes en el seguimiento a lo largo del tiempo de cada uno de los jugadores. En los apartados siguientes se aborda cada una de estas fases por separado.

Detección

A partir de las características radiométricas y geométricas (bordes, colores y patrones) realizaremos la detección de las regiones, expresadas como rectángulos, que representarán a cada uno de los jugadores.

Para esta etapa usaremos, de manera similar a lo realizado en los casos anteriores, Faster R-CNN + Resnet. Además, debido a que la gran mayoría de *datasets* públicos como COCO (Common Objects in Context)[8] incluyen personas como clases, podemos esperar que nuestro modelo aprenda rápido a detectar los jugadores. De hecho, en este trabajo, entrenamos nuestros modelos partiendo de modelos pre-entrenados con COCO, es decir, usaremos modelos pre-entrenados para hacer *finetuning* de nuestros modelos.

Seguimiento

El seguimiento de los jugadores quedará fuera de nuestro trabajo, por lo que formará parte del trabajo futuro. El objetivo de etapa es ubicar la posición de un mismo jugador a lo largo de un punto o incluso un partido entero, partiremos de n detecciones realizadas con los algoritmos descritos en la anterior subsección y deberemos relacionar cada detección con el jugador adecuado. Por ejemplo, en un partido de individuales un buen detector de jugadores será capaz de encontrar la ubicación de dos jugadores en cada uno de los frames de nuestras secuencias, la tarea del algoritmo de seguimiento será relacionar correctamente cada una de esas ubicaciones, un *bounding box*, con el jugador adecuado.

Algunos de los problemas que deberemos resolver en esta etapa serán la similar apariencia entre jugadores parecidos (misma ropa, misma apariencia física, etc), errores del detector de jugadores debido a las oclusiones parciales o totales durante ciertos momentos del partido y finalmente, multitud de cruces de los jugadores que dificultarán la tarea de seguimiento.

3.2.6. Siguientes etapas

En esta última sección, introducimos brevemente el resto de las etapas de nuestro *pipeline* para analizar un partido de bádminton.

Estas etapas son una tubería de hechos y desarrollos encadenados, por lo que entendemos que la salida de una etapa podrá ser usada para la realización de las siguientes. Por una parte, esto nos permite resolver nuestro problema de forma incremental, desde hechos a bajo nivel a eventos complejos. Un hecho (feature) de bajo nivel podría ser la

ubicación del volante; mientras que un evento complejo sería detectar que un punto ha terminado o que un jugador ha ganado un punto. Sin embargo, esta metodología hace que los errores se vayan encadenando etapa tras etapa; por ello, es necesario que al menos las etapas iniciales tengan una precisión muy alta. En caso contrario podría ser necesaria la revisión manual en alguno de los puntos del *pipeline*.

Detección y clasificación de los golpes

El siguiente paso de nuestra tubería, ver figura 3.1, será la detección y la clasificación de los golpes.

Esta etapa estaría formada a su vez por dos tareas, la primera de ellas sería la detección del momento del golpe, mientras que la segunda de ellas sería la clasificación del tipo de golpe realizado por el jugador.

En bádminton hay muchos tipos de golpes, cada uno con diversas variantes. Siguiendo la clasificación realizada en [15], podemos realizar una división de los golpes en las siguientes categorías.

- **Golpes de saque**: El saque es el golpeo que inicia el transcurso de un punto, a su vez tiene varias variantes cómo el saque de revés o el saque de derechas.
- **Golpes en la parte delantera de la pista**: Desde la parte delantera de la pista se pueden realizar diversos golpes como las dejadas, los remates en red (*kill*) y los *lobs*.
- **Golpes en la parte media de la pista**: En la parte media se realizan golpes como bloqueos o tensos.
- **Golpes en la parte posterior de la pista**: Finalmente en la parte posterior de la pista incluiríamos golpes como las dejadas desde el fondo de la pista o *drops*), los golpes de fondo a fondo o *clears* y los remates desde el fondo de la pista.

Además, la mayoría de estos golpeos pueden tener una variante defensiva u ofensiva en función de, la posición del rival, la altura del golpeo o la velocidad del volante, y pueden ser realizados de derechas o de revés; por ello, las posibles combinaciones son muchas. Finalmente, un aspecto importante a considerar es que los profesionales, con el objetivo de despistar al rival, son capaces de ocultar el golpe hasta casi el final del movimiento; lo que hace muy difícil identificar un golpe observando sólo el momento de la ejecución. Habitualmente, debemos evaluar la trayectoria del volante antes de poder clasificar un golpe.

Como mencionábamos algunos párrafos atrás, hemos dividido esta etapa en dos posibles tareas, la detección del momento del golpe y la clasificación del tipo de golpe. Para resolver la detección del momento del golpe, se proponen dos posibles vías. La primera idea podría ser usar el cambio de trayectorias del volante para marcar un instante como momento en el que se ha realizado un golpeo. La segunda idea propuesta, sería afrontar el problema como una clasificación de imágenes en dos clases ("golpeo" vs "no golpeo",

por ejemplo). O por supuesto, podríamos combinar de alguna forma ambas estrategias o incluso usar más información, como la posición de los jugadores o la posición del volante para entrenar un modelo capaz de discernir entre los momentos de golpe con momentos de transición.

Para la clasificación del tipo de golpe, creemos que posiblemente, la mejor alternativa pase por combinar información de alto nivel como la posición de los jugadores en el momento del golpeo y la trayectoria del volante después del golpeo, con información de bajo nivel asociada a los gestos del jugador durante el golpeo.

En conclusion, probablemente, para abordar esta parte de la secuencia propuesta habría que detectar el momento del golpeo y posteriormente clasificar el tipo de golpe. Para clasificar el tipo de golpe, primero habría que realizar un categorización lo más clara posible de los tipos de golpes en bádminton, que puede ser tan compleja como la descrita anteriormente, o quizás, una simplifación de ella.

Detección del ganador del punto

Detectar el ganador de un punto es el siguiente paso de nuestro *pipeline*. Realizar esta tarea usando eventos del juego no es trivial, pues prácticamente sería realizar la tarea del árbitro y linieres de un encuentro; lo que partiendo de vídeos de retransmisiones deportivas, con una única cámara en el fondo de la pista, es prácticamente imposible. Para poder construir un sistema así una única cámara no es suficiente y necesitaríamos disposiciones de cámara similares a las usadas por Foxtenn o El Ojo de Halcón.

En su lugar, se plantea este paso de una forma similar a cómo lo haría un espectador desde su casa. Desde nuestras casas, los espectadores sabemos quien ha ganado, con seguridad, un punto por dos principales hechos. El primero, cuando el marcador se incrementa a favor de uno de los jugadores o una de las parejas. El segundo, cuando un jugador saca para comenzar el siguiente punto; en bádminton, con el reglamento actual, siempre saca el jugador que ha ganado el punto previo, excepto en el inicio del encuentro, claro.

En conclusión, intuitivamente podríamos pensar que lo que aquí estamos proponiendo se parece más a un espectador inteligente, que iría tomando notas de cada evento a lo largo de un partido, que a un árbitro. Es decir, nuestro sistema se fiaría de las decisiones que el árbitro en pista ha tomado.

Seguimiento de la puntuación

Hasta este momento, todas las etapas explicadas, han involucrado un único punto aislado de observación. Pero un partido de bádminton se compone de varios sets y a su vez de varios puntos; por lo tanto, con el objetivo de entender un partido de forma global, en esta última etapa de nuestro *pipeline*, deberemos combinar la información de cada punto.

A grandes rasgos, esta etapa debería llevar la cuenta de los puntos y sets conseguidos por cada jugador, así como, entender cosas diferentes, como p.e. reconocer un intervalo de descanso entre puntos o entre sets.

En resumen, a lo largo de este capítulo hemos esbozado un proceso por el que podríamos, de forma automática, llegar a comprender los eventos de un partido de bádminton como si el sistema de un espectador se tratase. Este *pipeline* ha empezado detectando los elementos estáticos del juego hasta, por último, unir todos los eventos del partido para ofrecer una comprensión global del mismo.

Capítulo 4

Datasets y metodología

A lo largo de este capítulo se detallan los conjuntos de datos *datasets* que se utilizarán durante este trabajo. Estos serán utilizados para entrenar los modelos y para, posteriormente, evaluarlos.

Por otra parte en este capítulo también se explica la metodología de evaluación que utilizaremos y cuál es su relación con los datasets.

4.1. Datasets

Una parte muy importante de este trabajo son los conjuntos de datos o *datasets* que utilizaremos para entrenar y evaluar cada parte de del procedimiento presentado en el capítulo anterior.

De este procedimiento, tal y como se comentó en el capítulo anterior, nos centraremos en la detección de las líneas de la pista, la detección de los elementos estáticos también de la pista, la detección y el seguimiento del volante y, finalmente, la detección y el seguimiento de los jugadores; para cada una de estas tareas se usará un *dataset* manualmente etiquetado, que presentamos en las siguientes subsecciones.

Cada uno de estos *datasets* ha sido etiquetado manualmente por el autor de este trabajo debido a que no existen *datasets* públicos de bádminton. Para etiquetar estas imágenes se han utilizado herramientas de código abierto como [13] y [33], véase 4.1. Además en algunos casos se ha optimizado el proceso de etiquetado usando técnicas de *boostraping*.

El *boostraping* consiste en entrenar o desarrollar un modelo o procedimiento que funcione regular pero que ahorre una parte del tiempo de etiquetado manual. Primero etiquetamos manualmente un subconjunto pequeño de datos, con estos datos entrenamos un modelo, que por supuesto no funcionará demasiado bien. A continuación, usamos el modelo entrenado sobre el resto de los datos sin etiquetar consiguiendo de esta manera propuestas. Finalmente, de manera manual, debemos corregir las propuestas dadas por el modelo entrenado con pocos datos. Si el modelo que hemos entrenado con pocos datos acierta, por ejemplo, el 20 % de las veces, conseguiremos ahorrar el 20 % del trabajo

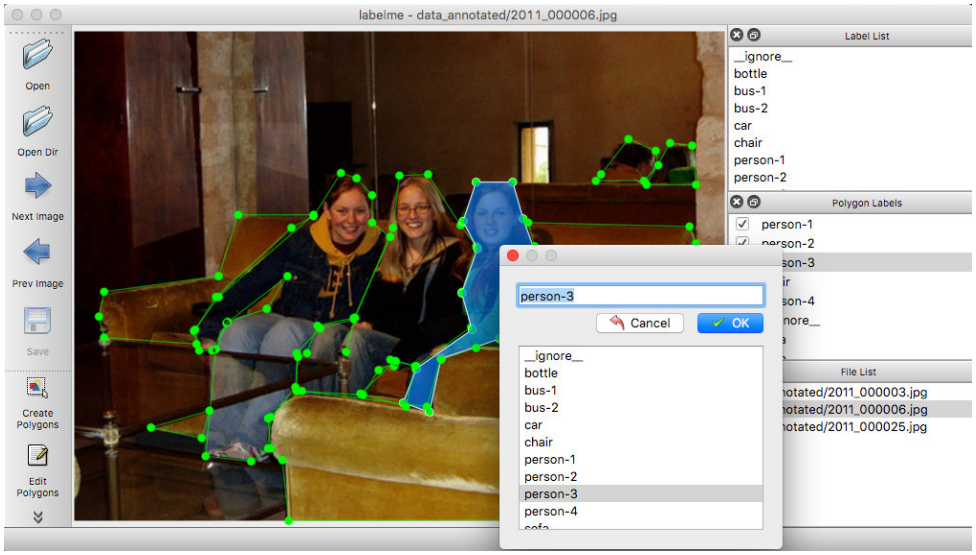

Figura 4.1: Herramienta de etiquetado Labelme

manual.

Un apunte de cara a siguientes desarrollos se refiere al tamaño de estos datasets, si bien la cantidad de datos que hemos etiquetado y utilizado es suficiente para un trabajo cómo este. Para un desarrollo aplicable en producción deberíamos ampliar el tamaño de estos *datasets* a varias miles o decenas de miles de muestras.

En cuanto al origen de los datos, cada una de las imágenes que forman parte de los *datasets* han sido extraídas de vídeos públicos en internet de competiciones profesionales de bádminton. Todos los *datasets* se incluyen en los entregables de este trabajo.

En las siguientes subsecciones revisamos cada uno de los conjuntos de datos que utilizaremos a lo largo de este trabajo.

4.1.1. Pista de Bádminton

El primer *dataset* está formado por imágenes que contienen la pista de Bádminton, donde cada una de las imágenes lleva asociada las anotaciones o etiquetas del polígono exterior de la pista, es decir, en este *dataset* sólo se etiquetan las líneas exteriores de la pista, ya que a partir de estas se podrían obtener el resto de líneas, hacemos esto ya que es más eficiente etiquetar sólo cuatro líneas (un polígono) que todas las líneas de la pista.

Como se puede ver en la Tabla 4.1.1, hemos dividido el *dataset* de 130 imágenes en dos conjuntos diferentes: entrenamiento y evaluación. Usaremos el conjunto de entrenamiento para entrenar nuestros modelos y usaremos el conjunto de evaluación para evaluar nuestros modelos. Nótese que, aunque es habitual tener una partición de validación, en este trabajo la hemos descartado dada la escasez de datos disponible. Para el resto de los *datasets* tendremos también estas dos particiones.

Es muy importante comentar cómo hemos realizado la partición del dataset teniendo en cuenta que en los problemas de *machine learning* se busca evaluar la precisión del

Conjunto	Frames
Entrenamiento	102
Evaluación	30

Cuadro 4.1: Tamaño del dataset de la pista de Bádminton

Conjunto	Frames
Entrenamiento	300
Evaluación	100

Cuadro 4.2: Tamaño del dataset de postes de Bádminton

modelo en datos desconocidos (i.e. con los que no se ha entrenado el modelo). Se trata de medir la capacidad de nuestro modelo de generalizar y realizar predicciones razonables ante datos que no ha aprendido de antemano. Por lo tanto, los conjuntos de entrenamiento y evaluación deben contener datos diferentes: en nuestro caso utilizaremos imágenes correspondientes a diferentes torneos. Por ejemplo si tenemos 3 imágenes del torne T, estas imágenes estarán en el conjunto de entrenamiento o en el conjunto de evaluación, pero nunca en ambos. Es decir, no utilizaremos una de estas imágenes en el conjunto de entrenamiento y las dos restantes en el conjunto evaluación. Esta misma idea la aplicamos en todos los *datasets* de este trabajo.

4.1.2. Postes de Bádminton

El segundo *dataset* está formado por imágenes de la pista de Bádminton, donde aparecen etiquetadas los postes con un rectángulo descrito por dos puntos (superior izquierda e inferior derecha).

4.1.3. Seguimiento del volante

El tercer *dataset* está formado por secuencias de vídeo compuestas por imágenes de fragmentos de juego, donde el volante aparece etiquetado en cada una de ellas. Como se observa en la Tabla 4.1.3, este *dataset* es con diferencia el más grande de los abordados en este trabajo. Contiene 10113 imágenes correspondientes a 46 secuencias de fragmentos de puntos de partidos de bádminton. Todas estas secuencias han sido revisadas manualmente con el objetivo de asegurar una alta calidad del dato tanto de entrenamiento como de evaluación.

De las cuarenta y seis secuencias, cuarenta y una han sido seleccionadas para entrenamiento y cinco para evaluación. Para garantizar una evaluación justa, las secuencias seleccionadas para evaluación son de torneos diferentes y posteriores en el tiempo a las secuencias del conjunto de entrenamiento.

En la Tabla 4.1.3 se detallan el número de imágenes que componen cada secuencia del dataset. Como puede apreciarse, el dataset cuenta con secuencias relativamente cortas y secuencias de varios cientos de frames. Como norma general, podríamos asumir que las secuencias con pocas imágenes corresponden a un simple golpe, mientras que las secuencias

Conjunto	Secuencias	Frames
Entrenamiento	41	8377
Evaluación	5	1736

Cuadro 4.3: Tamaño y resumen del dataset de seguimiento del volante

más largas contienen puntos enteros completamente etiquetados, es decir, estas secuencias largas contienen la ubicación del volante desde el momento del saque (donde el volante está en la mano de un jugador) hasta el final del punto donde el volante está en el suelo.

Secuencia	Frames
Entrenamiento	
0	51
1	23
2	51
3	42
4	38
5	46
6	51
7	49
8	51
9	51
10	24
11	51
12	53
13	226
14	301

Secuencia	Frames
15	401
16	202
17	122
18	194
19	237
20	259
21	225
22	176
23	648
24	201
25	149
26	150
27	567
28	125
29	537
30	190

Secuencia	Frames
31	243
32	75
33	783
34	426
35	150
36	125
37	221
38	193
39	245
40	425
Evaluación	
41	556
42	126
43	497
44	406
45	151

Cuadro 4.4: Descomposición en secuencias del dataset de seguimiento del volante

4.1.4. Detección de jugadores

El último *dataset* está compuesto por imágenes (frames) de fragmentos de juego, donde cada una de las imágenes está etiquetada con la ubicación de cada uno de los jugadores. Como el dataset contiene vídeos de partidos individuales o de dobles, cada frame puede contener dos o cuatro rectángulos anotados.

Conjunto	Frames
Entrenamiento	300
Evaluación	100

Cuadro 4.5: Tamaño del dataset de detección de jugadores

4.2. Metodología de evaluación

Para evaluar cada uno de los pasos en los que nos centraremos en este trabajo: detección de las líneas de la pista, detección de los postes, detección del volante y detección de los jugadores utilizaremos el conjunto de datos para evaluación o test que hemos definido en el apartado anterior.

Para cada desarrollo realizaremos siempre un evaluación visual cualitativa y una evaluación cuantitativa. La evaluación cuantitativa nos dará una métrica objetiva del comportamiento del modelo entrenado en el conjunto de evaluación; por su parte la evaluación visual cualitativa nos ayudará a entender los resultados en cada imagen, así como a identificar y entender los posibles casos de fallo.

Para la evaluación cuantitativa utilizaremos las métricas clásicas de recuperación de información *precision* y *recall* [5], definidas en las ecuaciones 4.1 y 4.2.

$$precision = \frac{TP}{TP + FN} \tag{4.1}$$

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

donde

- TP es el número de verdaderos positivos u objetos relevantes detectados correctamente,
- FN es el número de falsos negativos u objetos relevantes no detectados correctamente, y
- FP es el número de falsos positivos u objetos detectados que no son relevantes.

Para considerar que una predicción es un verdadero positivo (TP) respecto al *ground truth* utilizaremos la métrica *Intersection over Union* (IoU), ilustrada en la Figura 4.2. Por ejemplo, un valor IoU >0.5 implica que una región es un TP si su IoU respecto a un objeto de la misma clase en el *ground truth* es mayor que 0.5. Este valor se fija para cada evaluación en el capítulo 5, aunque el valor 0.5 es el más habitual en la literatura.

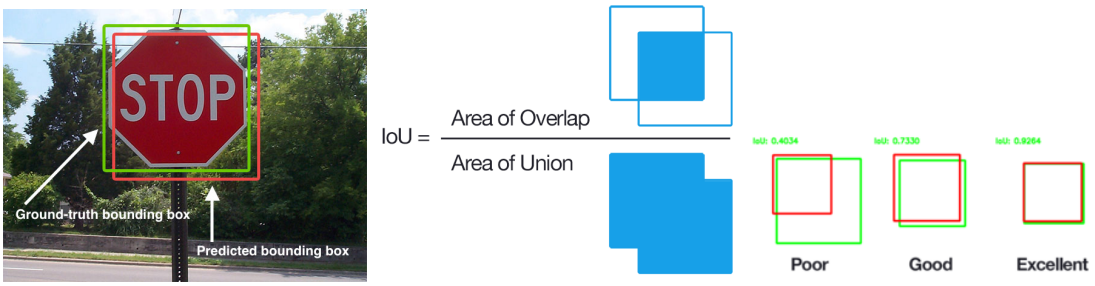

Figura 4.2: Intersection over Union (IoU) es una métrica de similitud para la detección de objetos en imágenes. Fuente: `http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection`

Capítulo 5

Resultados

A lo largo de este capítulo mostraremos los resultados obtenidos en cada una de las etapas recogidas en el capítulo 3. Los resultados obtenidos en algunas de estas etapas muestran que los procedimientos seguidos son aceptables para su uso en la industria ya que permitirían elevar el grado de automatización de esas etapas. Por el contrario, en otras etapas se necesitan mejoras adicionales para conseguir un grado de automatización con una precisión aceptable.

5.1. Detección de la pista

En esta sección se presentan los resultados para las diferentes aproximaciones propuestas para detectar las líneas de la pista de Bádminton en el conjunto de evaluación.

La Tabla 5.1 resume los resultados de la evaluación cuantitativa mientras que las sucesivas subsecciones incluyen la evaluación cualitativa, así como los casos de fallo para cada una de las alternativas propuestas. En todos los casos se ha considerado un verdadero positivo cuando el IoU entre las regiones del *ground truth* y la predicción es superior a 0.5.

5.1.1. Hough + intersecciones + Homografía

La primera de las aproximaciones planteadas ha conseguido un **recall** de 0.37 y una precisión de 0.92 en el conjunto de evaluación. Cómo muestra la Figura 5.1, la principal ventaja de esta aproximación es la exactitud de las líneas detectadas. Por otro lado,

Aproximación	**TP**	**FN**	**FP**	**Precision**	**Recall**
Hough + intersecciones + homografía	11	18	1	.92	.37
Faster R-CNN + Mask R-CNN	30	0	0	1.00	1.00
Faster R-CNN corners + homografía	28	2	0	1.00	.93

Cuadro 5.1: Evaluación cuantitativa de cada una de las aproximaciones para la detección de la pista (IoU >0.5)

presenta dos principales inconvenientes: no consigue una gran precisión en el conjunto de test y el tiempo de procesamiento por imagen es totalmente impredecible. Este tiempo varía desde unos pocos segundos hasta varios minutos en función del número de líneas detectadas por el algoritmo de Hough.

Figura 5.1: Resultados visuales para la detección de la pista: Hough + intersecciones + Homografía

5.1.2. Faster R-CNN + Mask R-CNN

La segunda aproximación está basado en la arquitectura para detección de objetos Mask R-CNN. Los resultados son excepciones ya que ha detectando la pista relativamente bien (IoU >0.5) en todas las imágenes. Si bien, debemos tener en cuenta que una precisión del 100 % no es realista y en este caso se debe al reducido tamaño del conjunto de evaluación (30 imágenes).

Además de la elevada precisión, otra ventaja importante de este método es el tiempo de procesamiento. Una GPU de última generación (NVIDIA 1080Ti) puede procesar hasta 2 o 3 fps. Por otro lado, la principal desventaja de esta aproximación es la inexactitud de las líneas detectadas, tal y como veremos en los resultados visuales. Esta falta de exactitud nos podría hacer renunciar a este método si necesitamos una precisión elevada en cuanto a la ubicación de las líneas de la pista se refiere.

Figura 5.2: Resultados visuales para la detección de la pista: Faster R-CNN + Mask R-CNN

5.1.3. Faster R-CNN (corners) + Homografía

Esta tercera aproximación es la que mejores resultados visuales consigue en el conjunto de evaluación. Aunque no son perfectos desde un punto de vista cuantitativo (tenemos un falso negativo), parece sencillo aumentar su precisión si aumentamos el conjunto de entrenamiento, que ahora consta de tan solo 100 imágenes. Además de la *precisión*, podemos conseguir tiempos de procesamiento inferiores usando una GPU de última generación. En las pruebas se han conseguido cifras de unos 10 fps en una NVIDIA 1080Ti.

La Figura 5.3 representa los resultados visuales obtenidos al enviar varias de las imágenes del conjunto de evaluación a la red neuronal. En cada una de las imágenes se representan las esquinas detectadas por la red y las líneas de la pista inferidas a partir de estas esquinas.

5.2. Detección de los postes

En esta sección se presentan los resultados obtenidos para detectar los postes de la pista de Bádminton en el conjunto de evaluación que consta de 100 imágenes. La tabla 5.2 resume los resultados de la evaluación cuantitativa y en las sucesivas subsecciones se muestran los resultados de la evaluación cualitativa para la detección de postes.

Figura 5.3: Resultados visuales para la detección de la pista: Faster R-CNN (corners) + Homografía

Procedimiento	TP	FN	FP	Precision	Recall
Faster R-CNN	200	0	1	1.00	.99

Cuadro 5.2: Resultados numéricos para la detección de postes (IoU>0.5)

5.2.1. Faster R-CNN

La aproximaciçon planteada en el capítulo 3 ha obtenido un *recall*[35] de 1.00 y una *precision* de 0.99 en el conjunto de evaluación. Aunque estos resultados parecen casi perfectos no deben entenderse como tal, pues el conjunto de test no es lo suficientemente grande como para validar esta solución.

Cómo podemos apreciar en los resultados visuales 5.4, la principal ventaja de este procedimiento es el rendimiento y la buena capacidad de generalización que consigue. Como principal inconveniente, podríamos mencionar una ligera falta de ajuste respecto a la ubicación real de los postes. Esta falta de precisión limitaría la detección de la posición exacta de la red calculada como el segmento que une el extremo superior de los postes.

Analizando tanto los resultados visuales cómo los resultados numéricos podríamos concluir que el procedimiento seleccionado es válido para abordar la detección de los postes de la pista de bádminton.

Figura 5.4: Resultados visuales para la detección de los postes: Faster R-CNN

5.3. Detección y seguimiento del volante

El tercer desarrollo abordado durante este trabajo ha sido la detección y el seguimiento del volante. Este ha sido el apartado donde más tiempo se ha invertido tanto en la parte de investigación, como en la parte de experimentación, incluyendo el etiquetado de los *datasets*. De hecho, el conjunto de datos utilizado para esta parte del trabajo es significativamente más grande, superando las 10000 imágenes que se comparan con las escasas 400 imágenes que teníamos en el conjunto de datos para la detección de postes.

Aunque la sección incluye la palabra seguimiento en el título, la aproximación empleada no resuelve el seguimiento propiamente dicho sino la detección del volante. De hecho varios de las aproximaciones evaluadas incluyen información temporal en la red para para mejorar la detección de la ubicación del volante. Con nuestra aproximación el seguimiento se podría obtener posteriormente usando algún procedimiento de interpolación para las posiciones temporales del volante detectadas por cualquiera de los algoritmos de detección planteados en esta sección.

La Tabla 5.3 muestra un resumen de los resultados obtenidos, para cada una de los aproximaciones. Como en los desarrollos anteriores, en cada una de las siguientes secciones se detallan los resultados para cada aproximación, tanto de forma numérica como visual.

Procedimiento	Entrenamiento		Evaluación	
	Precision	Recall	Precision	Recall
(1) Faster R-CNN	.94	.94	.35	.75
(2) Faster R-CNN 4 canales	.85	.86	.87	.79
(3) Faster R-CNN 6 canales	.86	.88	.83	.72

Cuadro 5.3: Evaluación cuantitativa de diferentes aproximaciones para la detección del volante (IoU >0.2)

Figura 5.5: Evaluación cualitativa para la detección del volante con Faster R-CNN

5.3.1. Faster R-CNN

La primera aproximación consiguió en el conjunto de evaluación un *recall* de 0.75 y una *precision* de 0.35, cifras muy lejanas a las alcanzadas en el conjunto de entrenamiento (0.94 y 0.94 respectivamente). Especialmente, llama la atención la diferencia en la métrica que nos indica la precisión del algoritmo, en la que tenemos una diferencia de casi 60 puntos porcentuales. Si bien, aunque esta diferencia es grande, también es bastante razonable si miramos alguno de los resultados visuales de la Figura 5.5. Como mencionamos en el capítulo 3, el volante suele ser una mancha blanca en la imagen, que incluso para un humano es difícil diferenciar en la imagen sin el contexto temporal del movimiento del volante; por eso es comprensible que el algoritmo detecte algunas de estas manchas como si se tratase del volante (falsos positivos).

Normalmente esa diferencia entre los resultados para el conjunto de entrenamiento con el conjunto de evaluación indica que el modelo no generaliza bien o que posiblemente hayamos sobre-entrenado el modelo. Algunas formas de reducir esa diferencia podrían ser entrenar con más datos o incluir mecanismos de regularización como *dropout* o *weight decay* para evitar que el algoritmo sobre-entrene (*overfit*).

Secuencia	IoU>0.2		IoU>0.5	
	Precision	**Recall**	**Precision**	**Recall**
41	.36	.90	.35	.88
42	.28	.64	.26	.60
43	.35	.67	.33	.61
44	.35	.68	.30	.58
45	.35	.71	.28	.57
Total	.35	.71	.32	.69

Cuadro 5.4: Evaluación cuantitativa para la detección del volante usando Faster R-CNN

La Tabla 5.3.1 muestra los resultados obtenidos usando *Faster R-CNN* en cada una de las secuencias del conjunto de evaluación.

Adicionalmente hemos generado una serie de gráficos que permiten analizar la trayectoria del volante en algunas de las secuencias del conjunto de datos de evaluación (véase Figuras 5.7, 5.8, 5.9, 5.10 y 5.11). Cada figura está compuesta por tres gráficas. De izquierda a derecha, la primera refleja el *recall* a lo largo del punto, es decir, la tasa de acierto del procedimiento. En el caso ideal, esta gráfica sería una línea horizontal en el valor TP. La segunda y la tercera gráfica reflejan la trayectoria del volante (en azul el *ground truth*, en verde los TP y en gris los FP). Esta representación permite comparar la trayectorias detectada con respecto a la esperada. Podemos apreciar en estas figuras que ambas trayectorias describen una curva parecida que puede ser utilizada para interpolar la trayectoria real del volante.

5.3.2. Faster R-CNN 4 canales (RGB + Substracción de frames)

En esta segunda aproximación se ha utilizado de nuevo la arquitectura Faster R-CNN añadiendo un canal adicional en un mapa de color que corresponde a la diferencia entre el *frame* en el instante actual y el *frame* en el instante anterior. Esto nos permite añadir información temporal al detector con una ventana de tiempo muy restringida pero que contribuye a mejorar la precisión.

Los resultados conseguidos mediante esta aproximación han alcanzado un *recall* de 0.79 y una *precision* de 0.87 en el conjunto de evaluación (para un IoU >0.2), lo que significa una mejora de más de 40 puntos en *precision* y 5 puntos en *recall* respecto a la anterior aproximación. Además, el tiempo entrenamiento no se ha visto afectado a pesar de añadir un canal adicional a la entrada de la red (27 horas vs. 24 horas). La Figura 5.6 muestran los resultados visuales obtenidos en esta aproximación. Nótese que se estas cuatro imágenes son las mismas que se han utilizado en la aproximación anterior para facilitar su comparación.

Siguiendo la misma metodología de otras aproximaciones, la Tabla 5.3.2 muestra los resultados detallados para cada secuencia del conjunto de evaluación. En general, este procedimiento mejora todas las métricas para todas las secuencias, excepto para la secuencia 43 donde este procedimiento es un punto peor que el anterior.

Figura 5.6: Evaluación cualitativa para la detección del volante con Faster R-CNN y 4 canales a la entrada de la red

Secuencia	IoU>0.2		IoU>0.5	
	Precision	Recall	Precision	Recall
41	.91	.95	.89	.93
42	.76	.67	.72	.63
43	.89	.66	.85	.63
44	.82	.79	.69	.67
45	.85	.79	.68	.62
Total	.87	.79	.80	.73

Cuadro 5.5: Evaluación cuantitativa para la detección del volante usando Faster R-CNN y 4 canales

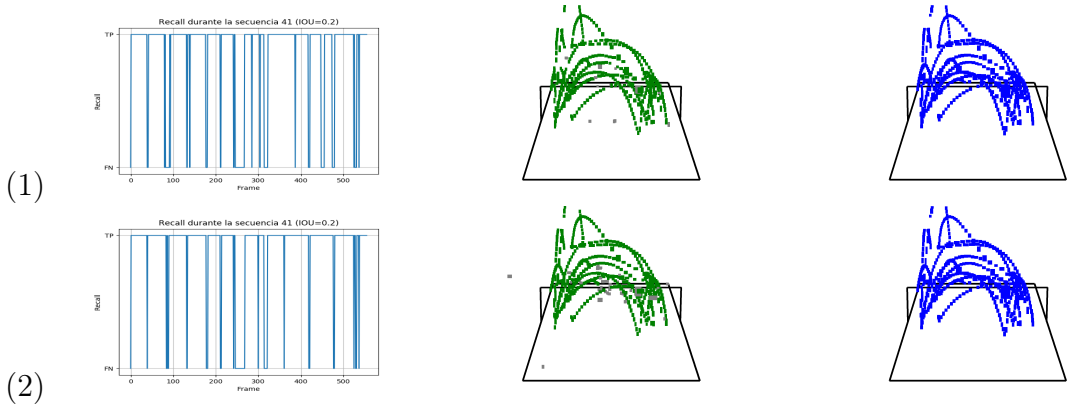

Figura 5.7: Análisis de las secuencias de evaluación: Sec. 41

5.3.3. Faster R-CNN 6 canales (RGB + Substracción de frames)

Esta tercera aproximación es añade a la anterior 2 canales más con el objetivo de enriquecer aún más la información a la entrada de la red, concatenando a la imagen original los 3 canales RGB resultantes de la substracción de los *frames* consecutivos de una secuencia.

Los resultados obtenidos son comparables a los obtenidos usando 4 canales, pero el tiempo de entrenamiento ha pasado de 27 horas a más de 40. Esto hace que la anterior aproximación sea más eficiente y escalable en el tiempo. La Tabla 5.3.3 muestra los resultados obtenidos para cada secuencia del conjunto de evaluación. En términos de precision y recall se observa como estos resultados son incluso ligeramente peores, lo que parece indicar que esta aproximación generaliza peor.

Secuencia	**IoU>0.2**		**IoU>0.5**	
	Precision	**Recall**	**Precision**	**Recall**
41	.85	.94	.84	.92
42	.72	.62	.72	.62
43	.83	.62	.81	.60
44	.81	.77	.67	.63
45	.85	.72	.72	.61
Total	.83	.76	.77	.71

Cuadro 5.6: Evaluación cuantitativa para la detección del volante usando Faster R-CNN 6 canales

5.4. Detección de los jugadores

En esta sección se presentan los resultados visuales y numéricos del método propuesto para detectar los jugadores de la pista de Bádminton en el conjunto de evaluación (que consta de 100 imágenes). Como mencionamos en el capítulo de metodología, el seguimiento de los jugadores formará parte del trabajo futuro.

(1)

(2)

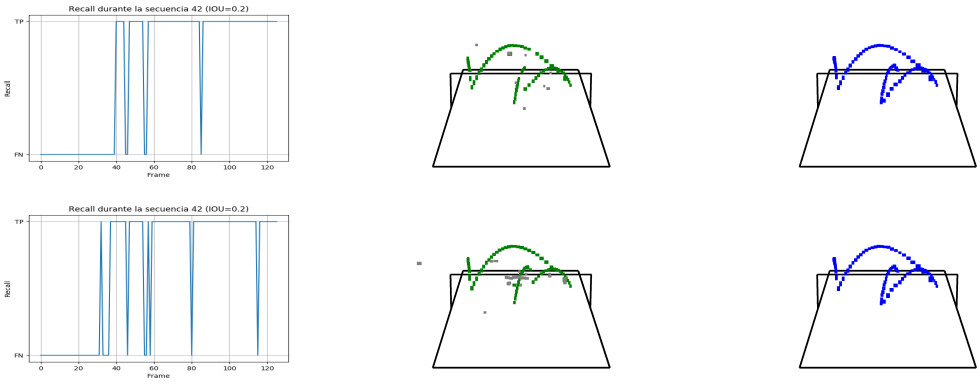

Figura 5.8: Análisis de las secuencias de evaluación: Sec. 42

(1)

(2)

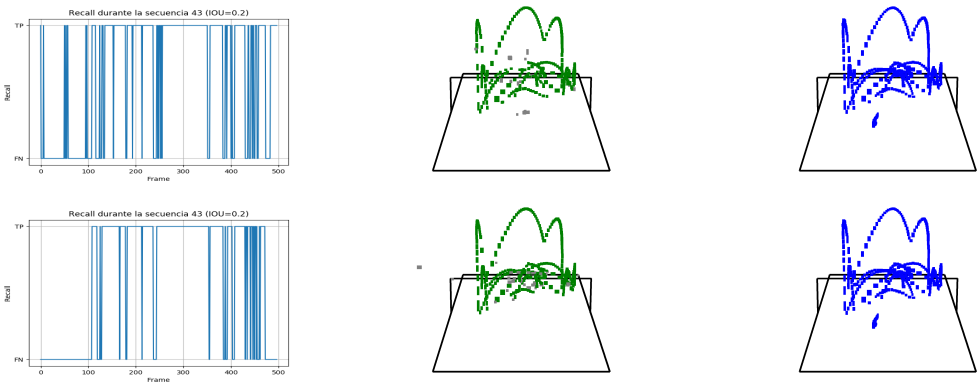

Figura 5.9: Análisis de las secuencias de evaluación: Sec. 43

(1)

(2)

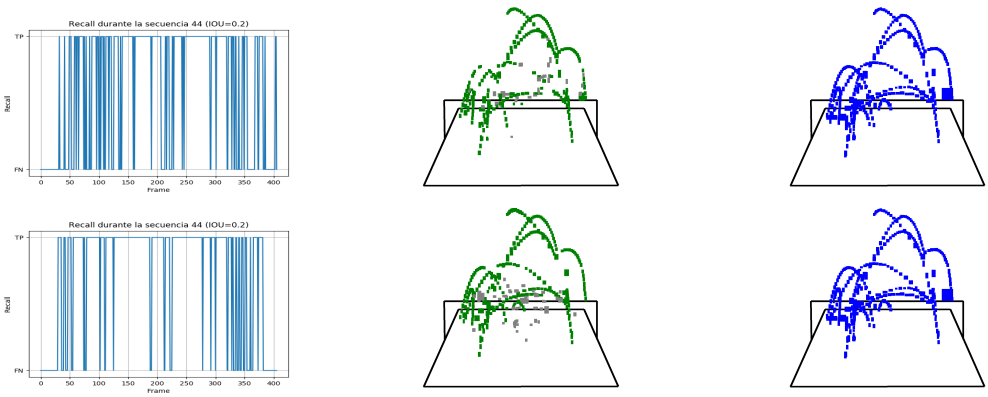

Figura 5.10: Análisis de las secuencias de evaluación: Sec. 44

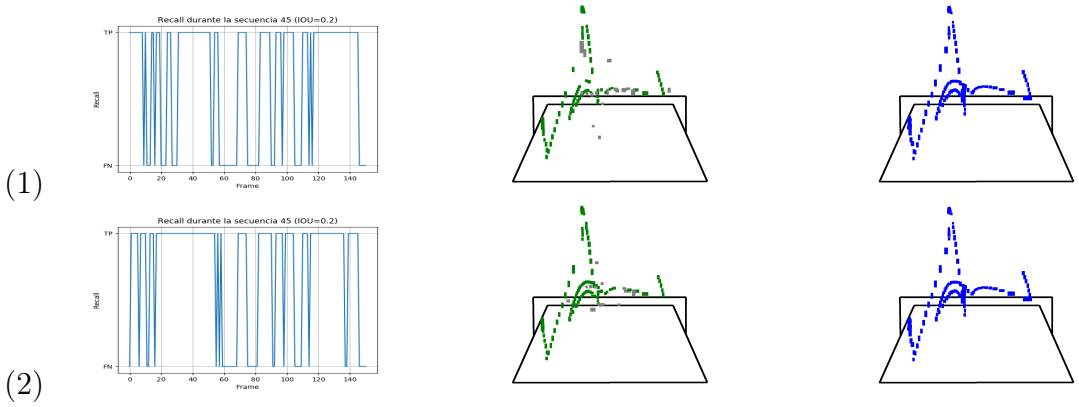

Figura 5.11: Análisis de las secuencias de evaluación: Sec. 45

La tabla 5.4 muestra el resumen de los resultados numéricos y en la correspondiente subsección se muestran los resultados visuales para la detección de jugadores.

Procedimiento	**TP**	**FN**	**FP**	**Precision**	**Recall**
Faster R-CNN	299	4	11	.96	.99

Cuadro 5.7: Resultados numéricos para la detección de los jugadores (IoU>0.5)

5.4.1. Faster R-CNN

El procedimiento planteado, explicado en el capítulo de metodología consiguió en el conjunto de test un *recall*[35] de 0.99 y una *precision* de 0.96 en el conjunto de test. Repitiendo palabras anteriores, aunque estos resultados parecen casi perfectos no deben entenderse como tales, pues el conjunto de test no es lo suficientemente grande como para dar por válido este modelo entrenado. Aunque por otra parte, estos buenos resultados sirven para corroborar que el planteamiento elegido es el adecuado y en el momento en el que existan más datos, tanto para entrenar como para evaluar, conseguiremos un modelo realmente bueno.

Cómo podemos apreciar en los resultados visuales 5.12, la principal ventaja de este procedimiento es el rendimiento y la buena capacidad de generalización que consigue. Como punto negativo, y algo inherente al modelo usado (Faster R-CNN), podríamos mencionar una ligera falta de ajuste respecto a la ubicación real de los jugadores y los problemas dados por las oclusiones. Probablemente, las oclusiones serán un problema a resolver principalmente en los partidos de dobles en los que se puede apreciar constantemente cruces entre los jugadores de una misma pareja. En la etapa de seguimiento, que dejamos para el trabajo futuro, será un problema que habrá que abordar.

Analizando tanto los resultados visuales cómo los resultados numéricos podríamos concluir que el procedimiento seleccionado es válido para abordar la detección de los jugadores a lo largo de un punto de bádminton.

Figura 5.12: Resultados visuales para la detección de los jugadores: Faster R-CNN

Capítulo 6

Conclusiones

En este capítulo se enumeran las conclusiones de los desarrollos y experimentos realizados. Además, se enumeran las principales aportaciones de este trabajo que pueden resultar de interés para la implementación de sistemas semejantes, aplicados o no al mundo del deporte. Finalmente, el capítulo concluye con una lista no exhaustiva de posibles desarrollos futuros tanto en el ámbito científico como en el ingenieril.

6.1. Desarrollos realizados

Esta sección está destinada a destacar las principales conclusiones extraidas de los esxperimentos realizados con diferentes aproximaciones para cada una de las etapas indicadas en el capítulo 3.

6.1.1. Detección de la pista

Para el problema de la detección de las líneas de la pista Faster R-CNN ha resultado ser la mejor aproximación. Este procedimiento combina la detección de las esquinas de la pista con una homografía respecto a un modelo de la pista de bádminton.

Hemos seleccionado este procedimiento como el más adecuado para este problema por la *precisión* y el *recall* obtenidos en el conjunto de evaluación, la velocidad de ejecución del algoritmo (hasta 5fps), la capacidad de generalización del procedimiento y el buen nivel de ajuste conseguido por el procedimiento respecto a la ubicación real de las líneas.

Como mencionábamos en el capítulo 5, para acabar de validar este procedimiento deberíamos obtener un *dataset* de varias miles de muestras y utilizarlo para entrenar nuestro modelo y para validarlo.

6.1.2. Detección de los postes

Para el caso de la detección de postes, la aproximación basada en Faster R-CNN ha obtenido los mejores resultados en términos de precisión y tiempos de inferencia. De

manera similar a lo comentado en las conclusiones para la detección de la pista para validar completamente la solución a este problema habría que ampliar el *dataset*.

6.1.3. Detección y seguimiento del volante

El desarrollo sobre la detección y el seguimiento del volante ha sido la etapa a la que se ha dedicado más esfuerzo en este trabajo. Aun así, no está totalmente claro si las soluciones planteadas son totalmente válidas y posiblemente habría que seguir trabajando en este problema en el futuro.

La mejor solución obtenida es la basada en Faster R-CNN con 4 canales. Los resultados está lejos de ser aceptables, por lo que surgen dudas de si realmente esta aproximación es la mejor para este problema o no. Quizás, utilizando más datos para entrenar nuestro modelo podríamos mejorar los resultados obtenidos. Aun así, como se aprecia en las Figuras 5.7, 5.8, 5.9, 5.10 y 5.11 el procedimiento falla mucho durante el comienzo y el final de los puntos, es decir, cúando el volante está en la mano de un jugador o en el suelo después de haber finalizado el punto. Intuitivamente podríamos pensar que la red está fijándose demasiado en la información procedente de la imagen diferencia (si no hay movimiento el cuarto canal no contiene información) y cuando no hay movimiento no permite detectar el volante.

Por otra parte, la aproximación base usando Faster R-CNN en las imágenes sin ningún tipo de información adicional ha consiguido un *recall* cercano a la aproximación basada en 4 canales, pero con una mejor aproximación al conjunto de entrenamiento. Por eso, posiblemente con más datos esta propuesta podría ser también válida.

La detección y el seguimiento del volante, no es un problema que esté resuelto, pero nos hemos aproximado hacia una posible solución. Probablemente, la aproximación más adecuada sea alguna variación, combinación o extensión de las ideas aquí propuestas. Intuitivamente también podríamos buscar un detector que maximizase el *recall* y posteriormente utilizar alguna mecanismo extra para eliminar los falsos positivos de este utilizando información temporal.

Como siguientes pasos existen diferentes vías. La primera de ellas, como hemos mencionado, sería seguir incrementando el tamaño del conjunto de entrenamiento con la esperanza de obtener modelos cada vez mejores. La segunda podría ser la combinación de redes neuronales convolucionales con redes neuronales recurrentes que añadieran información temporal a nuestro modelo.

6.1.4. Detección de los jugadores

Las conclusiones para la detección de los jugadores son similares a las obtenidas para la detección de la pista y los postes. La solución planteada parece adecuada pero habría que validarla con más datos reducir la incertidumbre.

6.2. Principales contribuciones

En esta sección se enumeran y describen brevemente las aportaciones realizadas en este trabajo.

La primera de las contribuciones realizadas ha sido el procedimiento o *pipeline* para entender un partido de bádminton utilizando retransmisiones televisivas. Para este procedimiento hemos desarrollado varias etapas durante este trabajo, aportando en algunas ellas una solución de gran precisión. Además, este procedimiento, es adaptable a otros deportes de raqueta como podrían ser el tenis o el voleibol.

La segunda contribución está relacionada con la detección de la pista. En este trabajo se han desarrollado varios procedimientos que funcionan con un buen rendimiento y precisión y que mejoran el estado del arte.

Finalmente, relacionado con el seguimiento del volante, mencionamos dos contribuciones que a nuestro entender pueden ser muy interesantes dentro del área del seguimiento de objetos móviles. Primero, hemos creado un *dataset* totalmente etiquetado con más de 10000 imágenes, correspondientes a 46 secuencias de vídeo, que puede resultar de gran utilidad para validar futuras aproximaciones en el campo del seguimiento de objetos pequeños. La segunda, es la utilización de redes convolucionales con canales extra en los que introducir información temporal, que, como hemos demostrado, puede mejorar el seguimiento de manera sustancial.

6.3. Trabajo futuro

En esta sección se incluyen algunas de las futuras líneas de trabajo. Estas posibles tareas no deben entenderse como una sucesión de tareas consecutivas donde finalizar la tarea A que permite realizar la tarea B, pues muchas de ellas no están directamente relacionadas.

A lo largo de este trabajo, nos hemos centrado principalmente en el proceso para entender un partido de bádminton pero el despliegue y mantenimiento de un sistema basado en *machine learning* como el que hemos planteado es tan importante como la parte científica, en esta sección mencionaremos alguno de estos aspectos.

Por otra parte, la cantidad de datos generados por el sistema propuesto no es despreciable. Por ello, el correcto modelado de los datos y su almacenamiento será crítico para la escalabilidad del mismo. También hablaremos de esto en las siguientes subsecciones.

Finalmente, una solución de este estilo no tiene ningún valor sin una aplicación o interfaz que permita a los usuarios o potenciales clientes su explotación y uso; completaremos este capítulo introduciendo este tema.

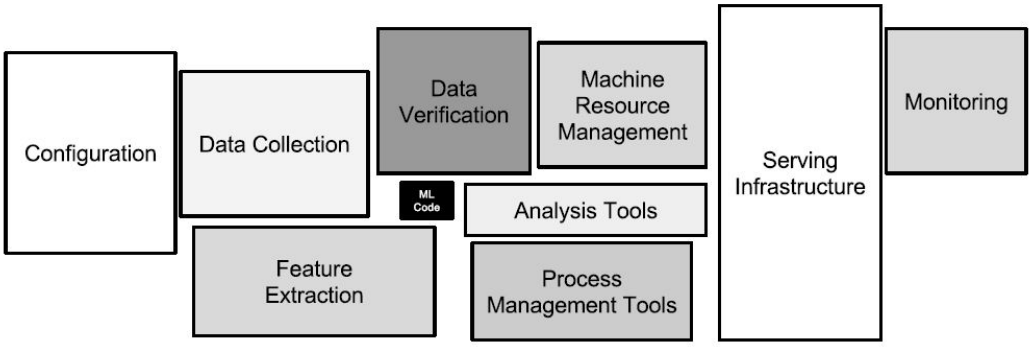

Figura 6.1: Componentes de un sistema de machine learning
Según [12] sólo una pequeña parte del código en un sistema ML es código ML, (como se aprecia en la caja negra del centro de la imagen). El resto de la infraestructura auxiliar es grande y compleja.

6.3.1. Completar *pipeline* de comprensión de bádminton

La primera línea de trabajo futuro consistiría en continuar con el desarrollo del *pipeline* que hemos propuesto en el capítulo 3. Para poder cerrar por completo los desarrollos realizados en este trabajo, una de las tareas que habrá que abordar será el incremento de los conjuntos de datos utilizados.

El principal punto de mejora es el seguimiento del volante. Algunas de las ideas para mejorarlo incluyen RNN y la ampliación del dataset de entrenamiento, con el objetivo de mejorar el *recall*.

Finalmente, podríamos trabajar en ampliar nuestro procedimiento para entender otros deportes de red o raqueta como el voleibol o el tenis, siendo el tenis sea el siguiente paso más evidente.

6.3.2. *Machine learning* en producción

Desplegar sistemas basados en *machine learning* en producción es un problema en sí mismo, ya que hay más piezas involucradas de lo que a priori podría parecer. Según Google[12], en stos sistemas el código encargado de *machine learning* es una porción realmente pequeña del mismo. Así, nuestro sistema tendrá que tener componentes encargados de servir los modelos, contar con herramientas para el análisis de los datos y la monitorización de resultados.

En este contexto, en los últimos años han surgido diversas alternativas. Quizás, la más popular actualmente es Kubeflow[38], un sistema de código abierto construido sobre Kubernetes que ofrece herramientas para gestionar todo el ciclo de vida asociado al *machine learning*, desde el preprocesado de datos, pasando por el entrenamiento, hasta el despliegue de los modelos generados como microservicios totalmente escalables. Básicamente, Kubeflow intenta minimizar el salto entre los científicos de datos y el despliegue en producción, haciendo más fácil servir modelos de *machine learning* que además sean

escalables usando contenedores en Kubernetes.

Otro aspecto importante en estos sistemas, y muchas veces también olvidado, es que la automatización requiere entrenamiento y mejora continua de los modelos. Es decir, en el hipotético caso de que consiguiéramos un sistema con una precisión cercana al 100 %, nada ni nadie puede asegurar que ese rendimiento se va a mantener a lo largo del tiempo. Por eso, es crucial definir, diseñar e implementar soluciones que faciliten un control de la calidad de la salida automática del sistema.

Además es necesario revisar manualmente un porcentaje de los datos finales, en función de la criticidad del sistema. El proceso de revisión permitirá medir el funcionamiento del sistema en comparación con lo que un humano haría, permitiendo así la obtención de métricas que podríamos usar para saber, por ejemplo, cuando es necesario re-entrenar el modelo. Un sistema basado en *machine learning* requiere constante mantenimiento, pues los modelos se van degradando con el paso del tiempo y es necesario que el sistema se vaya amoldando a los nuevos datos.

Estos y otros temas relativos al despliegue, a la monitorización y al mantenimiento de sistemas basados en *machine learning* deben ser abordados de cara a ofrecer soluciones de calidad y escalables durante el paso del tiempo; y que por supuesto para el sistema que nosotros hemos propuesto deberán ser abordadas en el futuro.

6.3.3. Gestión de los datos

El sistema aquí propuesto utiliza datos para entrenar los modelos que soportarán la solución. Asimismo, el sistema también generará multitud de información asociada a la monitorización del sistema y a la salida del mismo. Por lo tanto, una correcta gestión del proceso y el almacenamiento de los datos será crucial en el futuro para la escalabilidad de la solución.

Para empezar contamos con los *datasets* de entrenamiento y validación, que al menos estarán formados por secuencias de vídeo y por las correspondientes anotaciones. En este trabajo, el tamaño de los datos ya es de varios GB y sólo es una aproximación a lo que el sistema podría necesitar en producción.

Segundo, tendremos que manejar y almacenar la salida del sistema. Esta será una sucesión de eventos, tales como la posición de los jugadores, la posición del volante o los golpes a lo largo de cada *frame* del partido, más la correspondiente información estática como la posición de la pista o lo postes. Almacenar esta información para cada instante de un partido para, por ejemplo, todos los partidos de un torneo o todos los partidos de un jugador no es algo trivial. Además, esta información debería almacenarse de una forma estructurada o semi-estructurada que permita, posteriormente, realizar análisis de esa información de una forma ágil. No olvidemos, que justo ese, sería el objetivo final del sistema.

En tercer lugar, también deberemos gestionar información relativa al funcionamiento del sistema en sí, como *logs* o métricas de rendimiento, que nos permitirían detectar posibles errores y mejorar el sistema.

6.3.4. Futuras aplicaciones

Finalmente, el sistema debería proporcionar interfaces para que los potenciales clientes pudieran interactuar con él. Estas interfaces podrían ser APIs para que otros sistemas pudieran obtener información de forma automática y bajo demanda, o podrían ser clásicas interfaces de usuario, en las que, por ejemplo, un entrenador pudiera analizar información relativa a sus jugadores en su panel de control en la vista de administrador del sistema.

Bibliografía

[1] Y. LeCun y col. «Gradient-Based Learning Applied to Document Recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324.

[2] Jungong Han, Dirk Farin y Peter H. N. de With. «Generic 3-D Modeling for Content Analysis of Court-Net Sports Sequences». En: *Advances in Multimedia Modeling*. Ed. por Tat-Jen Cham y col. Springer Berlin Heidelberg, 2006, págs. 279-288. ISBN: 978-3-540-69429-8.

[3] Jungong Han, Dirk Farin y col. «Broadcast court-net sports video analysis using fast 3-D camera modeling». En: *IEEE Transactions on Circuits and Systems for Video Technology* 18.11 (2008), págs. 1628-1638.

[4] J. Deng y col. «ImageNet: A Large-Scale Hierarchical Image Database». En: *CVPR09*. 2009.

[5] Mark Everingham y col. «The pascal visual object classes (voc) challenge». En: *International journal of computer vision* 88.2 (2010), págs. 303-338.

[6] Andrej Karpathy. *Lessons learned from manually classifying CIFAR-10*. 2011. URL: http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/. (accessed: 22-May-2018).

[7] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, págs. 1097-1105. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[8] Tsung-Yi Lin y col. «Microsoft COCO: Common Objects in Context». En: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[9] Ross B. Girshick. «Fast R-CNN». En: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[10] Kaiming He y col. «Deep Residual Learning for Image Recognition». En: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[11] Shaoqing Ren y col. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». En: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[12] D Sculley y col. «Hidden Technical Debt in Machine Learning Systems». En: *NIPS* (ene. de 2015), págs. 2494-2502.

[13] Tzutalin. *LabelImg*. 2015. URL: https://github.com/tzutalin/labelImg. (accessed: 23-Aug-2018).

[14] Jonathan Huang y col. «Speed/accuracy trade-offs for modern convolutional object detectors». En: *CoRR* abs/1611.10012 (2016). arXiv: 1611.10012. URL: http://arxiv.org/abs/1611.10012.

[15] D. Serrano, J. Fraile y F Álvarez. *Manual del Monitor de Bádminton*. Federación Española de Bádminton, 2016. ISBN: 9788461743797.

[16] Kaiming He y col. «Mask R-CNN». En: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[17] Foxtenn - Extreme Tennis analysis. *Foxtenn - Diamond Academies*. 2018. URL: http://www.foxtenn.com/academies. (accessed: 23-May-2018).

[18] Foxtenn - Extreme Tennis analysis. *Foxtenn - Diamond Tournaments*. 2018. URL: http://www.foxtenn.com/tournaments. (accessed: 23-May-2018).

[19] Foxtenn - Extreme Tennis analysis. *Foxtenn - IN&OUT*. 2018. URL: http:http://www.foxtenn.com/in&out. (accessed: 23-May-2018).

[20] TechSmith Corporation. *Coach's Eye Sport Video Analysis App*. 2018. URL: https://www.coachseye.com. (accessed: 19-May-2018).

[21] Dartfish. *Dartfish - Federación Española de Bádminton*. 2018. URL: https://www.dartfish.tv/ChannelCollections?CR=p44660. (accessed: 21-May-2018).

[22] Dartfish. *Dartfish - Toma decisiones eficientes basadas en soluciones de video y datos inteligentes*. 2018. URL: http://www.dartfish.com/. (accessed: 19-May-2018).

[23] XOS Digital. *XOS Digital | Professional Sports Video Editing Solutions*. 2018. URL: http://www.xosdigital.com. (accessed: 19-May-2018).

[24] Hudl. *Hudl Assist - Video Breakdown and Analysis*. 2018. URL: https://www.hudl.com/products/assist. (accessed: 19-May-2018).

[25] Hudl. *One platform to help the whole team improve*. 2018. URL: https://www.hudl.com/products/hudl. (accessed: 19-May-2018).

[26] Hudl. *Sportscode - Flexible performance analysis tailored to your team*. 2018. URL: https://www.hudl.com/elite/sportscode. (accessed: 19-May-2018).

[27] Krossover. *Krossover Game Film Tools for Coaches & Athletes*. 2018. URL: https://www.krossover.com/. (accessed: 19-May-2018).

[28] Hawk-Eye Innovations Ltd. *Hawk-Eye Innovations Ltd*. 2018. URL: https://www.hawkeyeinnovations.com. (accessed: 21-May-2018).

[29] Performa Sports Ltd. *Performance Analisys Software - Video Analisys Software*. 2018. URL: https://www.performasports.com/. (accessed: 23-May-2018).

[30] Nacsport. *Nacsport | Software de Video Análisis*. 2018. URL: https://www.nacsport.com. (accessed: 19-May-2018).

[31] RII Sports Technology. *Dashboards*. 2018. URL: `http://www.r2sportstech.com/dashboards.html`. (accessed: 19-May-2018).

[32] VidSwap. *Sports Video Analysis Software and Sports Video Editing Software | VidSwap*. 2018. URL: `http://vidswap.com`. (accessed: 19-May-2018).

[33] Kentaro Wada. *labelme: Image Polygonal Annotation with Python*. 2018. URL: `https://github.com/wkentaro/labelme`. (accessed: 23-Aug-2018).

[34] Wikipedia. *Visión artificial — Wikipedia, La enciclopedia libre*. 2018. URL: `https://es.wikipedia.org/w/index.php?title=Visi%C3%B3n_artificial&oldid=107906918`. (accessed: 27-May-2018).

[35] Wikipedia contributors. *Precision and recall — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-September-2018]. 2018. URL: `https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=853202943`.

[36] Roberto Arroyo y col. «Deep Learning of Visual and Textual Data for Region Detection Applied to Item Coding». En: *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*. 2019, págs. 1-12.

[37] Roberto Arroyo y col. «Integration of Text-maps in Convolutional Neural Networks for Region Detection among Different Textual Categories». En: *Conference on Computer Vision and Pattern Recognition (CVPR). Language and Vision Workshop*. 2019, págs. 1-4.

[38] Kubeflow Authors. *Kubeflow: The Machine Learning Toolkit for Kubernetes*. 2019. URL: `https://www.kubeflow.org/`. (accessed: 27-Jun-2019).

[39] Jonathan Huang y col. *Tensorflow Object Detection API*. 2019. URL: `https://github.com/tensorflow/models/tree/master/research/object_detection`. (accessed: 5-Jul-2019).

Apéndice A

Materiales complementarios

Este apéndice describe los materiales adjuntos en el CD entregado junto a este documento.

A.1. Código fuente

El código fuente asociado a los desarrollos realizados durante este trabajo puede ser consultado en *Github* (`https://github.com/dgseten/bad-cv-tfm`) y bajo el directorio *src* en el CD entregado.

Las partes más importantes del código fuente son las siguientes:

- */src/bad-cv-tfm/README.md*: Archivo *markdown* con las instrucciones para configurar e instalar las dependencias del código fuente.
- */src/bad-cv-tfm/court*: Scripts relativos a los desarrollos realizados para la detección de la pista.
- */src/bad-cv-tfm/poles_players*: Scripts relativos a los desarrollos realizados para la detección de los postes y los jugadores. Aunque la detección de los jugadores y los postes han sido documentados de manera separada, el código fuente de ambas tareas se ha combinado.
- */src/bad-cv-tfm/shuttle*: Scripts relativos a los desarrollos realizados para la detección y el seguimiento del volante.
- */src/bad-cv-tfm/tools*: Scripts complementarios desarrollados para descargar vídeos de internet, así como para la extracción de los *frames* objetivo de los mismos. Por *frames* objetivo nos referimos a aquellos que hemos utilizado para entrenar y validar los desarrollos presentados en este trabajo.
- */src/bad-cv-tfm/other*: Otros desarrollos realizados durante este trabajo para resolver los objetivos del mismo, que por no cumplir la calidad o el rendimiento esperado no han sido continuados.

- *... :* El resto del código se corresponde a software *open source* de terceras partes que hemos integrado en el repositorio actual para facilitar el uso y configuración del mismo. Este código de terceras partes proviene en su mayoría del repositorio del framework Tensorflow Object Detection API[39].

A.2. Configuración de los entrenamientos

Todos los modelos basados en *Deep Learning* han sido entrenados usando el framework Tensorflow Object Detection API [14] [39]. Este framework contiene implementaciones para varias de las principales arquitecturas para la detección de objetos usando redes convolucionales. Entre las disponibles están ResNet, Faster R-CNN y Mask R-CNN que han sido las utilizadas en este trabajo.

En este framework, los trabajos de entrenamiento se configuran usando ficheros *protocol buffer* en formato de texto. En estos ficheros de configuración se indican los hiperparámetros para nuestro entrenamiento como el *Learning Rate*, el número de iteraciones durante las que entrenaremos (*epochs*), las métricas de evaluación que deseamos usar o los parámetros configurables para las arquitecturas seleccionadas.

Las configuraciones con las que se han obtenido los resultados expuestos en esta memoria se encuentran en las siguientes ubicaciones:

- */pipelines/model_court_corners:* Configuración para la detección de los corners de la pista.
- */pipelines/model_court_masks:* Configuración para la detección de la mascara de la pista.
- */pipelines/model_player_and_poles:* Configuración para la detección de los jugadores y los postes. Aunque la detección de los jugadores y los postes han sido documentados de manera separada, el entrenamiento de ambas tareas se ha combinado.
- */pipelines/model_shuttlecock:* Configuración para la detección del volante.
- */pipelines/model_shuttlecock_4_channels:* Configuración para la detección del volante usando imágenes de 4 canales.
- */pipelines/model_shuttlecock_6_channels:* Configuración para la detección del volante usando imágenes de 6 canales.

A.3. Datasets

Debido a las licencias de los vídeos utilizados los *datasets* no pueden ser compartidos públicamente, al menos en este momento. Si en el futuro se consiguiese autorización de los propietarios de los vídeos, los datasets generados se publicarían.