# Análisis de un partido de tenis

Jill Areny Palma Garro, Raúl Villar Casino, Erik Villarreal Gallardo

**Abstract**— This project focuses on utilizing OpenCV to detect the contours of a tennis court, stabilize the image using homographies, estimate the velocity of the tennis ball without a means of direct verification, and detect ball bounces. The project initially achieved promising results during the training phase; however, when applying the trained model to test scenarios or real-life matches, the performance noticeably degraded.

While the project demonstrated promising results during the training phase, the performance degraded significantly when applied to real-life matches or test scenarios. This disparity between training and real-world performance indicates the presence of challenges or limitations that were not adequately addressed during the development phase. Further investigation and refinement of the algorithms and methodologies are necessary to improve the robustness and generalizability of the system.

**Keywords**—OpenCV, contour detection, image stabilization, velocity estimation, ball bounce detection, training, test scenarios, real-life matches, performance degradation.

## 1. INTRODUCCIÓN

Este proyecto se basa en aprovechar grabaciones de partidos de tenis para investigar y aprender las distintas posibilidades que la visión por computador nos puede otorgar. El interés en este proyecto radica en la gran cantidad de información que se puede obtener de algo tan simple como un partido de tenis, y que luego se puede aplicar a otros deportes similares como bádminton, pádel, ping pong, etc.

Los objetivos variarán a lo largo del proyecto, a medida que se vayan completando o en función de su dificultad. Los objetivos establecidos inicialmente son:

- Rastrear la bola y detectar su ubicación en todo momento.
- Detectar los límites del campo.
- Detectar cuando y donde la bola ha rebotado.
- Calcular la velocidad de la bola.

## 2. ESTADO DEL ARTE

### 2.1. Proyectos de otros estudiantes o empresas pequeñas

Para obtener información de métodos actuales hemos investigado proyectos universitarios, de código abierto (open source) y artículos de empresas privadas. El proyecto más cercano a lo que buscamos obtener es un trabajo de fin de grado de la Universidad de Barcelona realizado por Christian José SOLER[1].En él se busca rastrear la bola en un partido de tenis y detectar cuándo ha botado mediante OpenCV. Sus resultados han sido excelentes a la hora de reconocer la bola, pero ha tenido problemas para detectar los botes debido a la calidad de la cámara que ha utilizado en sus experimentos.

Más allá de proyectos individuales, encontramos varios prototipos de Inteligencias Artificiales que son capaces de detectar nuestros objetivos e incluso muchas más mejoras, como calcular la velocidad de la bola, el estado del jugador, el tipo de jugada que se ha producido y analizar los fallos de los jugadores[2]. En la actualidad, el método más utilizado es el aprendizaje automático (machine learning), el cual, con una buena etiquetación, logra unos resultados mucho mejores que el resto de métodos y sobrepasa las posibilidades de los demás.

### 2.2. Aplicaciones en partidos actuales

El sector profesional del tenis actualmente ya utiliza el aprendizaje automático (machine learning) a diario. En los partidos se emplea el denominado .ºjo de halcón", el cual detecta la bola y los contornos del campo, y recrea la jugada para aclarar si una bola ha tocado el campo o ha salido fuera. El aprendizaje automático es una herramienta muy útil en el arbitraje de partidos y también puede serlo para un análisis más profundo.

## 3. PROPUESTA

### 3.1. Metodologías para los objetivos

Se tratarán los objetivos en el orden en que deberían ser implementados, ya que algunos dependen de otros.

#### *3.1.1. Detectar los límites del campo*

Es necesario detectar los límites del campo para poder realizar una homografía correctamente y transformar el campo de un espacio tridimensional a un plano de como se vería el campo en vista desde arriba. Para esto deberíamos utilizar un detector de esquinas para encontrar las esquinas del campo (hay que tener en cuenta el dibujo del campo, ya que hay diversas líneas interiores que pueden ayudar a detectarlo).

#### *3.1.2. Rastrear la bola y detectar su ubicación en todo momento*

Para la detección de la bola, utilizaremos un método de sustracción de fondo, ya que es bastante eficiente. Primero, se alinearán las fotos a partir de los límites del campo y se calculará la media con todas las imágenes del partido. Al restar el fondo, deberíamos obtener detección en los jugadores, la bola, algunos espectadores y, probablemente, un poco de ruido en algunas partes de la imagen.

A partir de esta segmentación, podremos clasificar a los jugadores y la bola de forma separada, para así rastrear la bola y a los jugadores por separado.

#### *3.1.3. Detectar cuando y donde la bola ha rebotado*

A partir de la información anterior, tendremos la trayectoria de la bola durante el partido. A partir de esto, utilizaremos algún tipo de aprendizaje computacional (aún por debatir) en el que emplearemos una sección de la trayectoria y detectaremos cuándo y dónde ha habido un rebote en esa secuencia. Estos rebotes estarán etiquetados manualmente en los vídeos que vamos a utilizar.

Para dividir las posiciones del vídeo en secciones, lo haremos de tal manera que cada sección represente un golpe del jugador. De esta forma, una única secuencia tendrá una duración desde el golpe del jugador 1 hasta el golpe del jugador 2 y viceversa.

#### *3.1.4. Calcular la velocidad de la bola*

Con la bola ya rastreada, y los límites del campo detectados, haremos una homografía que fuerce el tamaño del campo para que sea homogéneo en ambas partes de la cancha, de forma que podamos calcular la distancia entre cada pixel, usando como referencia los tamaños originales de un campo de tenis. De esta manera podemos obtener el desplazamiento y en cuanto tiempo lo ha hecho la bola, y calcular la velocidad.

### 3.2. Datos a utilizar

El objetivo del proyecto es poder utilizar nuestros algoritmos en una grabación de partido real, pero debido a que la mayoría son de pago y los que hay disponibles son más difíciles de tratar, ya que son las grabaciones cara al público. Obtendremos las grabaciones sacadas del videojuego AO Tennis 2[3]. De esta manera podremos manipular la cámara para acercarla o alejarla, y poder probar distintos ángulos y sus respectivos resultados.

### 3.3. Medidas de rendimiento

Nuestro proyecto, al no basarse en fotografías, sino en vídeos, presenta una mayor complicación para ser etiquetado. Aunque los partidos de tenis son conjuntos de datos bastante comunes, no hay ninguno con etiquetaciones relevantes en cada fotograma. Esto se debe a que la mayoría de los que existen se mantienen privados para evitar competencia entre las empresas.

Dado que nuestros objetivos son fáciles de detectar a simple vista (localización de la bola o cuándo ha rebotado), el rendimiento será evaluado mayoritariamente a ojo, excepto en el caso de los rebotes, donde intentaremos etiquetar manualmente algunas jugadas para realizar nuestras propias pruebas.

## 4. Experimentos, resultados y análisis

Se han realizado diversos experimentos para poder analizar los métodos que hemos utilizado para poder realizar los objetivos que se han mencionado anteriormente.

### 4.1. Preparación del dataset

Los videos utilizados han sido obtenidos del juego AO Tenis 2. Para ello, hemos puesto a dos CPU a jugar entre ellas un partido, y hemos deshabilitado todas las opciones del HUD posible para que la escena se vea lo más simple y fiel a una grabación de verdad que hemos podido. Para la captura del video hemos utilizado el programa 'OBS Studio' [4].

La ventaja de grabar nosotros es que hemos podido decidir la resolución (hemos escogido 1920x1080) y el framerate (30 fps), y siempre que necesitemos más jugadas, podemos grabar otra vez sin que el cambio de un partido a otro sea enorme.

El código ha sido ejecutado en el entorno de Google Colab, el cual nos permite tener hasta 12GB de almacenamiento RAM. Aunque hemos reutilizado mucho las variables para no ocupar demasiada memoria, hemos tenido que acortar el training set a una jugada de 8 segundos. Todo el train se basará en ese video, y una vez finalizado probaremos otras jugadas para ver si el sistema se mantiene firme en el test.

### 4.2. Detección de los límites del campo

Hemos utilizado varios métodos basados en una función de opencv llamada "findContours", a partir de esta función hemos probado diferentes soluciones que describiremos a continuación:

- "Bounding rectangle": a partir de los contornos hemos podido obtener un rectángulo que engloba el campo, por desgracia no tiene una perspectiva aplicada de forma que solo sería posible utilizarla para una homografía simple donde hacemos que el vídeo se mantenga estático.
- Detector de esquinas de Harris: a partir del contorno más grande detectado (el cual es el campo por la localización de la cámara), buscamos las coordenadas de las esquinas que encontramos en el contorno, y seleccionamos los cuatro cantones más cercanos al contorno detectado, para poder mejorar la precisión de las esquinas del campo, ya que con solo la detección del contorno la estabilización no es muy buena.
- "approxPolyDP": a partir del contorno más grande detectado (siempre es el del campo según nuestros tests) reducimos sus puntos utilizando una función especial hasta llegar a solamente 4 puntos que representen los bordes del campo

De todos estos métodos hemos visto que el método de aproximar el contorno a un polígono formado por 4 puntos es el mejor método, ya que nos proporciona los puntos necesarios para hacer cualquier tipo de homografía.

### 4.3. Detección de la bola

A partir de este punto hemos dividido el desarrollo del proyecto en dos direcciones distintas.

#### *4.3.1. Homografía aérea*

La primera dirección ha sido la detección de la bola para poder calcular su velocidad, se ha utilizado una homografía que visualiza el campo desde una vista central en la que las proporciones del campo son correctas.

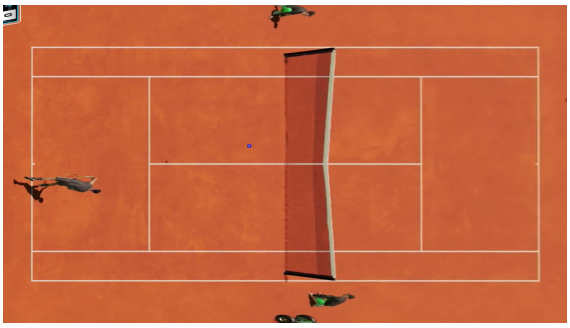

Fig. 1: Fotograma del train con la homografía aérea.

Después de tener la homografía, segmentamos la bola con una máscara HSV y hacemos operaciones de filtrado no lineal con el objetivo de conseguir que los contornos sean mucho más visibles. Pero, cuando aplacábamos esto al vídeo, había varios frames en los que no detectaba nada, ya que o la bola se deformaba tanto que no encontraba contornos o esta cambiaba de color, haciendo que la máscara no lo detectara.

Por eso, decidimos que cuando no detectara la bola en con la máscara HSV, intentara hacerlo con una máscara en escala de grises. Esto nos dió buenos resultados, ya detectaba la bola en casi todos los frames. Pero aún había algunos en los que no los detectaba, y aparte nos dimos cuanta de que las dos máscaras detectaban contornos en el público o en sitios en los que no tendría que detectar (falsos positivos).

Para solucionar la detección de contornos que no eran la bola, limitamos el sitio donde buscar la bola con ayuda del frame anterior y un número máximo de píxeles por los que se traslada la bola.

Y, para mejorar la detección de la bola en todos los frames, hicimos que cuando no detectara la bola por ninguna de las máscaras, hiciera una estimación de donde está con ayuda de los píxeles anteriores.

Aun con todas las mejoras que hemos hecho, hay algunos frames donde lo detecta mal. Principalmente, pasa esto en el frame siguiente de cuando esta hace un cambio de dirección o de sentido, y hemos llegado a la conclusión de que pasa esto porque la estimación en este frame no sirve porque esta es dependiente de una dirección y sentido que en este frame han sido alterados.

#### *4.3.2. Estabilización*

La segunda dirección ha sido la detección de la bola para poder calcular los rebotes, en este método se ha utilizado una homografía que elimina el movimiento de la cámara de manera que no afecten a la posición del campo en la imagen.

Hemos calculado la media y la desviación de la secuencia de imágenes del vídeo de train y a partir de estas dos métricas hemos aplicado un algoritmo de restado de imagen en el que hemos podido eliminar el fondo y parte de los jugadores debido a que están quietos la mayor parte del vídeo. Por lo tanto, obtenemos una imagen donde la bola es el objeto que más varía y podemos detectar su posición fácilmente.

Ha habido problemas al detectar la bola cuando está justamente en las líneas del campo y, por lo tanto, se detecta la posición en otro lugar donde haya más variación de fondo. Esto se ha solucionado analizando la velocidad de la bola y estableciendo una velocidad máxima, para evitar errores futuros se ha agregado un margen a esta velocidad máxima.

También hay problemas al detectar la bola cuando un jugador la golpea debido a que esta acción se realiza pocas veces y, por lo tanto, hay una alta variación en la imagen cuando se efectúa. Este error no ha sido posible solucionarlo debido a las limitaciones de nuestro algoritmo, pero más adelante se podría aprovechar este comportamiento para la detección de golpes de jugadores.

### 4.4. Detección de cuando y donde ha rebotado la bola

Para este objetivo hemos utilizado la detección de bola estabilizada descrita en el punto 4.3.2. Se ha analizado la trayectoria de la bola y se han llegado a diversas conclusiones que describiremos a continuación.

Para analizar la trayectoria hemos utilizado diversas métricas y hemos ajustado la detección de botes según un bote que sucede en el vídeo. Las métricas que hemos utilizado son:

- Ángulo y velocidad: Hemos multiplicado el ángulo de variación entre las últimas 2 direcciones (a partir de los últimos 3 puntos detectados) con la velocidad resultante.

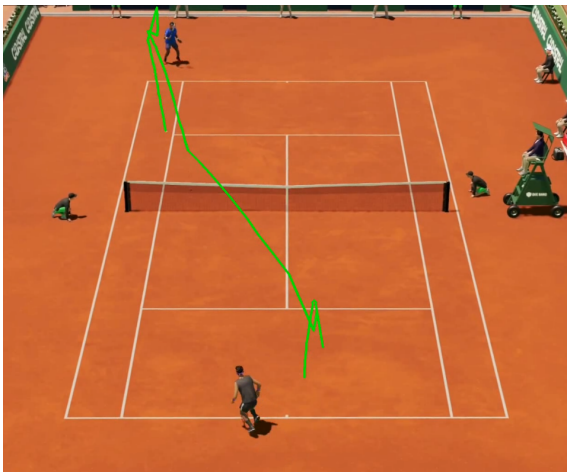

Fig. 2: Trayectoria detectada en el conjunto de train, podemos ver que hay errores de detección en los golpes de los jugadores

- Dirección del rebote: Hemos obtenido la forma de los cambios de dirección y su colinealidad.

Se han probado más métricas como la diferencia de velocidades, pero han resultado no ser indicativas de un bote después de analizarlas.

#### *4.4.1. Análisis del ángulo y velocidad*

Después de analizar esta métrica hemos llegado a la conclusión de que esta métrica obtiene un valor elevado con cambios de dirección a alta velocidad. Como los rebotes ocurren a alta velocidad, es fácil clasificar posibles botes a partir de esta métrica. Hemos considerado que un valor de 300 como mínimo para que se detecte como un posible bote da buenos resultados.

#### *4.4.2. Análisis de la dirección del rebote*

Como bien hemos mencionado anteriormente, la detección de la bola no llega a ser correcta cuando los jugadores golpean la bola, y, por lo tanto, ocurren muchos cambios de dirección y velocidad en toda la duración del golpe. Esto significa que la métrica anterior detecta estos golpes como múltiples posibles botes. Para reducir este error hemos analizado la forma y dirección que deberían de tener los rebotes.

Hemos llegado a la conclusión de que el único tipo de rebote que puede haber es el siguiente.

En la figura 3 vemos que los vectores de dirección se encuentran a la izquierda del vector rojo (0, 1). Mientras que la segunda dirección se encuentra a la derecha del vector amarillo, este vector es el mismo que la primera dirección. A partir de este análisis hemos deducido que ambos vectores deben de estar en el mismo lado, según el vector rojo, y que el segundo debe de estar en el lado contrario al primero. Es decir, si ambos se encuentran apuntando a la izquierda, el segundo vector debería rotar en el sentido de las agujas del reloj. Pasa lo mismo en el caso contrario, si ambos se encuentran apuntando a la derecha, el segundo vector debería rotar en el sentido contrario a las agujas del reloj.

El ángulo entre cualquiera de los dos vectores de dirección y el vector de arriba se calcula a partir del arcocoseno del producto vectorial multiplicado por la primera componente del vector de dirección. La posición relativa entre la primera y segunda dirección se calculan a partir de la fórmula de colinealidad entre vectores.

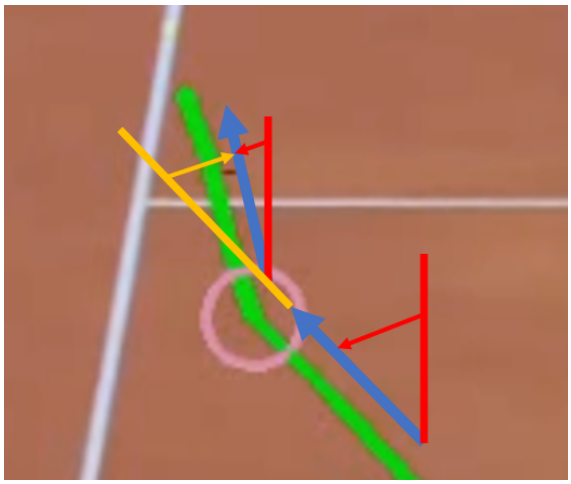

Fig. 3: Diagrama solapado con una parte de un fotograma donde ocurre un rebote, las barras rojas son la dirección de arriba, la barra amarilla es la extensión de la primera dirección para una mejor visualización

Utilizando este filtrado de botes reducimos la cantidad de rebotes de 15 a 3. Los únicos botes que quedan son el correcto y un golpe de cada jugador. Con un filtrado más avanzado se podrían extraer los golpes de los jugadores y diferenciar los golpes de los botes.

### 4.5. Cálculo de la velocidad de la bola

Para calcular la velocidad de la bola utilizaremos la homografía aérea, en la cual podemos ver la cancha al completo con las mismas proporciones en ambos lados. La desventaja de la cual partimos es que solo tenemos una cámara, y graba el partido alejado y con un ángulo parecido a unos 45 grados. Al no tener desde un inicio la cámara en unos 90 grados del suelo, esto genera que la altura de la bola va a afectar a como se va a ver su desplazamiento, pero no se va a tener en cuenta a la hora de calcular la velocidad debido a su complejidad. Haciendo que las velocidades vayan a no ser precisas, y se trate más de una aproximación.

La fórmula que hemos utilizado para obtener las velocidades es la siguiente:

$$velocity = (distancia/72.72)/(i - lastgoodframe) * framerate$$

Fig. 4: Fórmula utilizada para el cálculo de la velocidad.

La cual viene dada de los siguientes razonamientos:

- Distancia / 72.72 significa el desplazamiento, en metros. Este número está calculado con la conversión de las medidas de un campo de tenis oficial (10.97 x 23.77 metros) y las medidas en píxeles, de lo que ocupa el campo en nuestra homografía (1728 x 798 píxeles), el cual nos da que se recorre un metro por cada 72.72 píxeles de nuestro video.
- La distancia es calculada entre el frame actual, y el último frame en el que la bola fue detectada. Así que para que la velocidad no se descontrole cuando la bola no sea detectada en algunos frames, dividimos el resultado entre "i" (que indica el frame actual) y "lastgoodframe" (que indica el último frame en el que se detectó la bola).
- Por último, para pasar nuestro valor de metros/frame a metros/segundo, necesitamos multiplicar el valor por el framerate de la cámara, que en nuestro caso es 30.

Primero, mientras terminábamos de desarrollar las mejoras del rastreo de la bola, hicimos pruebas con una versión más simple, la cual detecta mucho peor las posiciones (solo la mitad de los frames detecta la bola, y al inicio de la jugada).

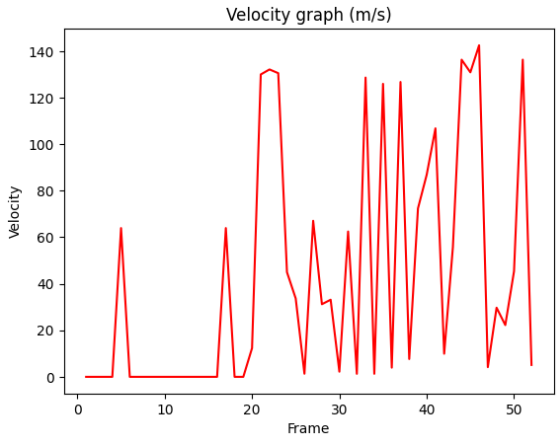


Fig. 5: Gráfica de las velocidades sin tratar

La gráfica de velocidades ha dado resultados muy disparatados, y que no tienen coherencia. Mirando las coordenadas que hemos usado para detectar la velocidad podemos averiguar que problema hay.

En la jugada original, el movimiento debería ser una línea horizontal, pero tenemos varios puntos que son mal detectados, esta es una de las razones de los malos resultados. Pero también hay que tener en cuenta, que puede haber mucha variación de velocidad, ya que no en todos los frames se va a mover la misma cantidad de píxeles, puede que en uno se mueva 1 pixel, pero en el siguiente frame aún no haya conseguido pasar del todo al siguiente y cuente como que no se ha movido. Para eso vamos a hacer una media de las velocidades en grupos de 5, y así corregir las variaciones.

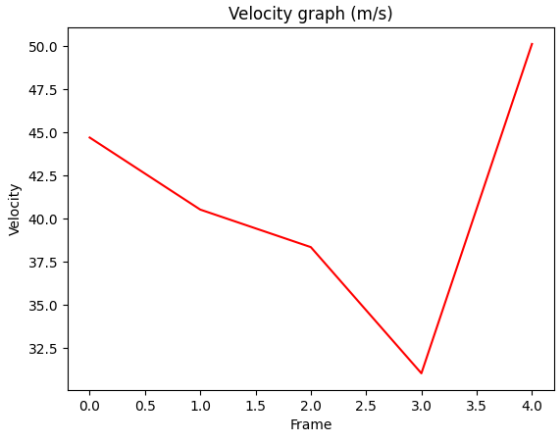


Fig. 6: Gráfica de las velocidades haciendo media.

Ahora hemos conseguido unos resultados mucho más verosímiles, ya que la jugada es un saque y hemos detectado una velocidad de 50 m/s, y en los partidos de tenis, el récord de la velocidad más alta es unos 65 m/s.

Cuando aplicamos la velocidad al rastreo de la bola mejorada, conseguimos la siguiente gráfica:

Los resultados son más adecuados a la jugada, ya que a diferencia de la figura 6, donde solo hay un cambio de ritmo. En esta

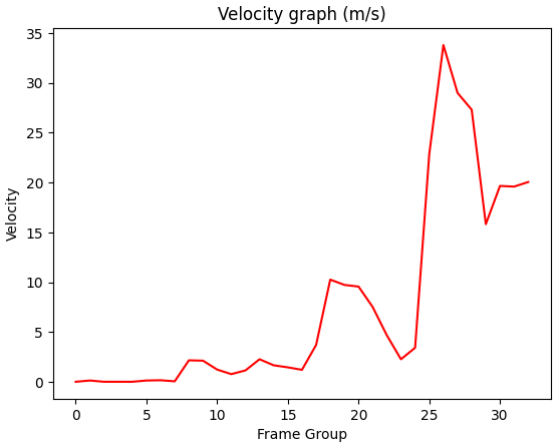


Fig. 7: Segunda gráfica de las velocidades haciendo media.

podemos observar como la bola coge y pierde velocidad cuando el jugador la lanza al aire, como sale disparada en el saque, y como pierde velocidad al ser devuelta por el adversario. Desafortunadamente, no podemos obtener información sobre la velocidad real de la bola para comprobar los resultados, así que solo podemos basarnos en que los resultados tienen una similitud de la velocidad con las de un partido de tenis de verdad, y en que podemos detectar en la gráfica los cambios de velocidad que vemos en el vídeo de training.

### 4.6. Pruebas en otras jugadas

#### *4.6.1. Jugadas con conjunto de test*

Hemos tenido bastantes problemas con la detección de la bola mediante la media y la desviación estándar de una sección de vídeo, debido a que los jugadores se encontraban muy quietos en la sección de training. Por lo tanto, hemos tenido que cambiar los parámetros del algoritmo de detección de bola y esto nos ha llevado bastante tiempo. Por desgracia el rendimiento no ha sido bueno y la detección ha sido muy incorrecta, se ha tenido que sacrificar cantidad de frames detectables para reducir la cantidad de frames incorrectos.

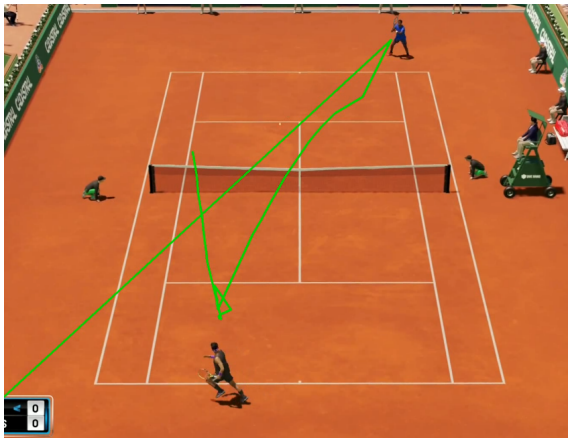

Fig. 8: Trayectoria detectada en el conjunto de test, podemos ver que hay errores de detección al final de esta y la cantidad de puntos de la trayectoria es baja

#### *4.6.2. Jugadas en grabaciones de partidos en vivo*

Hemos probado a implementar nuestro código con una grabación de unos 8 segundos de una jugada del Australian Open 2012. [5] Los resultados han sido peores de lo esperado, la estabilización por homografía ha funcionado a la perfección, incluso cuando en el video el ángulo es distinto al usado en los training y test. Pero la detección de la bola solo ha funcionado en 1 de los 150 frames analizados del partido.

## 5. Conclusiones

El proyecto ha funcionado muy bien en cuanto a rendimiento en el training. Varias jugadas del test no tienen problemas, pero cuando en el test los jugadores empiezan a moverse mucho, los resultados empiezan a empeorar, y a aparecer incongruencias. Aplicado a otros partidos reales, nuestro código no funciona a excepción de la detección del campo. Para el tiempo que hemos tenido, estamos muy contentos con los resultados, y hemos podido ver como en el estado actual del proyecto, el programa está demasiado ajustado al training, y tenemos que hacer nuevas mejoras, las cuales permitan mejorar el rendimiento en los test y otros ámbitos.

Como posibles mejoras:

- Habría que mejorar las formas de detección de la bola para el test
- Utilizar las sombras para mejorar la precisión de la detección de botes y del cálculo de la velocidad.
- Aprovechar toda la información que podemos sacar de los jugadores, para identificarlos y hacer un marcador

## Referencias

[1] Christian José SOLER, "Table Tennis Ball Tracking and Bounce Calculation using OpenCV," 22 Juny 2017, TFG, Universitat de Barcelona.

[2] Yu-Chuan Huang, I-No Liao, Ching-Hsuan Chen, Tsì-Uí İk*, and Wen-Chih Peng, "TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications," 8 Juliol 2019, National Chiao Tung University.

[3] Nacon and Big Ant Studios, "AO Tennis 2," 9 Enero 2020, videojuego de tenis. [Online]. Available: https://store.steampowered.com/app/1072500/AO_Tennis_2/

[4] OBS Project, "OBS Studio," 14 Julio 2014, software libre y de código abierto para grabación de vídeo y transmisión en vivo. [Online]. Available: https://store.steampowered.com/app/1905180/OBS_Studio/

[5] Australian Open TV, "The Best Game Of Tennis Ever? — Australian Open 2012," 23 Enero 2012, jugadas destacadas del partido de Andy Murray contra Michael Llodra. [Online]. Available: https://www.youtube.com/watch?v=oyxhHkOel2I