DETECCIÓN AUTOMÁTICA DEL TERRENO DE JUEGO DE FÚTBOL EN IMÁGENES DE VÍDEO

AUTOMATIC FIELD TRACKING IN RECORDED FOOTBALL GAMES

TRABAJO FIN DE GRADO
CURSO 2021-2022

AUTOR
PABLO JAVIER DEL PINO SÁNCHEZ

DIRECTOR
RAFAEL CABALLERO ROLDÁN

COLABORADOR
DRIBLAB SL

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

AGRADECIMIENTOS

Gracias a los miembros de DribLab S.L. por su colaboración e inestimable ayuda en la realización de este trabajo.

RESUMEN

Este trabajo se ha centrado en el tratamiento para la detección de las líneas correspondientes al terreno de juego de un partido de fútbol a partir de las retransmisiones producidas para la televisión.

Una vez aplicados todos los tratamientos previos se procede a la implementación del algoritmo encargado de la detección de las líneas correspondientes al terreno de juego, así como de su identificación y refinamiento buscando la máxima precisión posible y desestimando posibles falsos positivos o líneas irrelevantes para la detección del terreno de juego.

Por último, se documenta una breve implementación para un caso de uso concreto a partir de la eliminación del efecto de la perspectiva en la imagen, logrando una imagen resultado en un plano de dos dimensiones, a partir de la cual se abre un extenso campo de investigación y utilidad para diferentes áreas del mundo del fútbol como son los clubes, periodistas, analistas, etc...

Palabras clave

Visión artificial, Detección, Homografía, Fútbol, Vídeo, Imagen, Deporte

ABSTRACT

This work has focused on the processing for detection the lines corresponding to the field of play of a football match from the broadcasts produced for television.

Once all the previous treatments have been applied to the image, the algorithm in charge of detecting the lines corresponding to the playing field is implemented, as well as its identification and refinement, seeking the highest possible accuracy and discarding possible false positives or irrelevant lines for the detection of the playing field.

Finally, a brief implementation is documented for a specific use case based on the elimination of the perspective effect in the image, achieving an image result in a two-dimensional plane, from which a wide field of research and usefulness for different areas of the football world is opened such as clubs, journalists, analysts, etc...

Keywords

Computer Vision, Detection, Homography, Football, Video, Image, Sports

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

Capítulo 1 - Introducción

1.1 Contexto y objetivos

En el mundo del fútbol, así como en otros ambientes de la actualidad, la accesibilidad y democratización del dato es uno de los principales retos de cara a obtener un mayor rendimiento en la toma de decisiones de cualquier tipo.

En la actualidad, existe una carrera por la extracción de datos de valor que puedan ser utilizados por diferentes sectores relacionados con el fútbol (entrenadores, periodistas, directivos). En este marco la utilización de imágenes en vídeo para la extracción de información es un terreno prácticamente desconocido y con mucha investigación por delante.

La principal idea en la que se centra este trabajo es la de extraer información útil para diferentes sectores del ámbito deportivo desde las retransmisiones deportivas. Esta información se puede traducir en un cálculo de las distancias recorridas por cada uno de los jugadores, la creación de una imagen en 2D del desarrollo del juego buscando el análisis táctico del mismo, o incluso como ayuda al video arbitraje (VAR) con la regla del fuera de juego entre otros.

La complejidad del problema se vería reducida si en lugar de usar las retransmisiones públicas de los partidos de fútbol, se contara con una cámara ya calibrada y fija en cada estadio de modo que los parámetros visuales y las referencias del terreno de juego no se modificaran. A pesar de ello, buscando la accesibilidad del dato, se ha decidido realizar sobre retransmisiones públicas de modo que la extracción de la información sea genérica y aplicable a distintos contextos de países y estadios sin tener en cuenta sus recursos disponibles.

Como se mencionará en ciertas referencias posteriores, hay antecedentes de trabajos en una línea parecida a la expuesta. Es por ello, que lo que se busca mejorar a partir de las soluciones propuestas con anterioridad, es una solución adaptable de una manera genérica en diferentes contextos y con una velocidad computacional que pueda acercarse al tiempo real.

Este trabajo se centrará en la detección de las líneas del terreno de juego, entendiendo esto como el primer paso para la resolución del problema. Debido a su complejidad y extensión no se abordan los próximos pasos una vez las líneas del terreno de juego han sido detectadas, más allá de una humilde aproximación al cálculo de la homografía que derivaría en la transformación de la imagen al 2D y una propuesta de soluciones para abordar el cálculo de ella de manera automática.

Son muchos y de diferente idiosincrasia los apartados que tiene la solución de este método, y aunque no mucha, encontramos literatura también de otros sectores imprescindibles para la solución final como es la identificación de los jugadores mediante la detección de imágenes.

1.2 Tecnologías

La realización del trabajo se ha realizado íntegramente con el lenguaje de programación Python en su versión 3.10.4. Esta elección se debe a la alta disponibilidad de bibliotecas de visión artificial, así como de Machine Learning necesarias para el desarrollo del problema propuesto.

La biblioteca OpenCV ha sido la elegida para el tratamiento de las imágenes, su disponibilidad para distintos lenguajes de programación, su extensa documentación, así como su gran comunidad de usuarios han sido los principales motivos de su elección.

Por otro lado, han sido necesarias las librerías auxiliares Pandas y Numpy para el manejo y cálculo de datos, así como para el cálculo del modelo de regresión.

Se adjunta un enlace a GitHub en el que se encuentra disponible todo el código correspondiente a la solución propuesta:

ENLACE AL REPOSITORIO DE GITHUB

Capítulo 2 - Introduction

2.1 Context and goals

In the world of football, as in other current environments, the accessibility and democratisation of data is one of the main challenges in order to obtain a better performance in decision making of any kind.

Nowadays, there is a very competitive enviroment to extract valuable data that can be used by different sectors related to football (coaches, journalists, managers). In this context, the use of video images for the extraction of information is a practically unknown field with a lot of research ahead.

The main idea behind this work is to extract useful information for different sectors of the sports field from sports broadcasts. This information can be translated into a calculation of the distances covered by each of the players, the creation of a 2D image of the game looking for the tactical analysis of it, or even as an aid to video refereeing (VAR) with the offside rule among others.

The complexity of the problem would be reduced if, instead of using public broadcasts of football matches, there was a camera already calibrated and fixed in each stadium so that the visual parameters and the references of the pitch would not be modified. Nevertheless, in order to make the data accessible, it has been decided to carry out the study on public broadcasts so that the extraction of the information is generic and applicable to different countries and stadiums without taking into account their available resources.

As will be mentioned in some later references, there are different previous works along similar lines to the one described above. For this reason, what is seeking to improve based on the solutions proposed previously is an adaptable solution in a generic way and with a computational speed near to real time.

This work will focus on the detection of the lines of the playing field, understanding this as the first step towards the solution of the problem. Due to its complexity and extension, the next steps once the lines of the playing field have been detected are not addressed, beyond a humble approximation to the calculation of the homography that would

derive in the transformation of the image to 2D and a proposal of solutions to address the calculation of it in an automatic way.

2.2 Technology

The work has been carried out entirely with the Python programming language in its version 3.10.4. This choice is due to the high availability of artificial vision libraries, as well as the Machine Learning libraries necessary for the development of the proposed problem.

The OpenCV library has been chosen for image processing, its availability for different programming languages, its extensive documentation, as well as its large community of users have been the main reasons for its choice.

Also, the auxiliary libraries Pandas and Numpy have been necessary for data handling and calculation, as well as for the calculation of the regression model.

Attached is a link to GitHub where all the code corresponding to the proposed solution is available:

LINK TO THE REPOSITORY

Capítulo 3 - Máscara de fondo

Se detalla el método utilizado para la aplicación de una máscara a cada *frame* de modo que el ruido derivado de cualquier elemento ajeno al terreno de juego pueda ser eliminado.

Distintos factores han de ser tenidos en cuenta para la aplicación de esta máscara, como los factores climatológicos que influyen en la calidad de la imagen (lluvia, nieve, niebla…) o las sombras generadas en el terreno de juego en partidos jugados con luz natural.

3.1 Modelo de color HSV

El modelo de color HSV cuenta con tres canales con características distintas (Hue, Saturation, Value) que definen la totalidad del espacio de colores. A diferencia del modelo de color RGB, en este modelo nos encontramos con el matiz del color en un único canal (HUE), lo que supone que las diferencias de color debido al brillo o saturación de la imagen no repercutan a la hora de valorar si un píxel tiene un color similar a otro a pesar de las condiciones de iluminación de la imagen.

Siguiendo el método propuesto por (Ramirez Pereira Duarte, 2015) se realiza un histograma del valor HUE de la imagen suponiendo lo siguiente:

1. En una retransmisión de fútbol el color predominante será el del campo de juego.
2. Como consecuencia de la hora de juego o de las marcas del césped el terreno de juego podrá estar formado por más de un tono de verde.

3.2 Histograma de colores

Para lograr esto se realiza un histograma enfrentando la frecuencia de cada valor HUE en todos los píxeles del *frame* de modo que queden identificados los valores del HUE más repetidos, es decir, los valores de color que aparecen con más frecuencia en la imagen.

Como se ha mencionado anteriormente, una de las premisas de las que se parte es que el color predominante de la imagen será el verde correspondiente al terreno de juego.

Por ello una vez hayan sido identificados mediante el histograma los valores más repetidos, se puede saber a qué valor del HUE corresponde el máximo en caso de un único color sobre el terreno de juego, o los máximos en caso de que sobre el terreno de juego existan sombras y haya más de un color correspondiente al terreno de juego.

Una ver realizado este cálculo se selecciona el máximo absoluto y los valores que alcanzan al menos el 30% de la frecuencia máxima con un valor de HUE de ± 30 el valor máximo. De este modo se consiguen encontrar los valores HUE que representan el rango de verdes predominante en el frame. El cual coincidirá con el color del campo habiendo o no sombras en él.

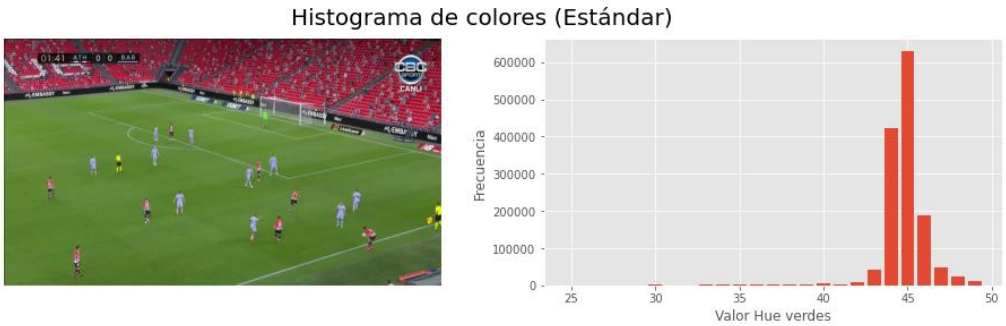


Ilustración 1. Histograma del valor HUE en un campo con color uniforme. Elaboración propia.

Ilustración 2. Histograma del valor HUE en un campo con sombras. Elaboración propia.

Como se observa en las ilustraciones 1 y 2, en el caso de un color de campo homogéneo el histograma indicará que pocos valores de HUE abarcan prácticamente la mayoría de verdes del campo. Sin embargo, en el caso del frame con una gran sombra en el terreno de juego el número de valores máximos relativos aumenta y es más cercano al valor máximo absoluto.

Esta comparación demuestra acertada la necesidad de filtrar cada frame por el rango de verdes dado desde un offset menor que el menor mínimo relativo y mayor que el

mayor máximo relativo. De esta forma conseguiremos abarcar los casos en los que el terreno de juego tenga un color homogéneo y un terreno de juego con sombras en él y por tanto más de un color.

3.3 Operaciones morfológicas

3.3.1 Operaciones básicas de filtrado morfológico (erosión y dilatación)

Estas operaciones son la base de cualquier tratamiento de imágenes binarias basadas en las formas que contiene una imagen. Tienen como entrada la imagen original y un elemento estructural. Las operaciones básicas son erosión y dilatación a partir de las cuales mediante combinaciones de estas se puede llegar a otras operaciones más complejas. (OpenCV, s.f.). El propósito de estas operaciones será el de simplificar la imagen eliminando detalles y ruido en ellas que puedan confundir al algoritmo.

3.3.1.1 Elemento estructural

El elemento estructural de entrada para las operaciones definirá los píxeles alrededor del píxel sobre el que se opera que serán tenidos en cuenta para las funciones explicadas anteriormente.

3.3.1.2 Erosión

Un píxel de la imagen original será considerado 1 si todos los píxeles del elemento estructural para ese píxel son 1. En cualquier otro caso será considerado 0 (OpenCV, s.f.). Esta operación reducirá el tamaño de las formas de la imagen e incluso eliminará las que sean muy pequeñas.

3.3.1.3 Dilatación

Opuesto a la erosión. Un píxel de la imagen será 1 si al menos un píxel del elemento estructural para el píxel original es igual a 1 (OpenCV, s.f.). Esta operación aumentará el tamaño de las formas de la imagen y eliminará los huecos dentro de las formas que no estén completas.

La librería OpenCV para Python incluye diferentes tipos de elementos estructurales para el cálculo de las diferentes operaciones morfológicas: *cv.MORPH_RECT, cv.MORPH_RECT y cv.MORPH_ELLIPSE*.

Ilustración 3. Original, erosión, dilatación. (OpenCV, s.f.).

En la Ilustración 4 se muestra la estructura con dos de los elementos estructurales proporcionados por OpenCV para un tamaño de 21x21.

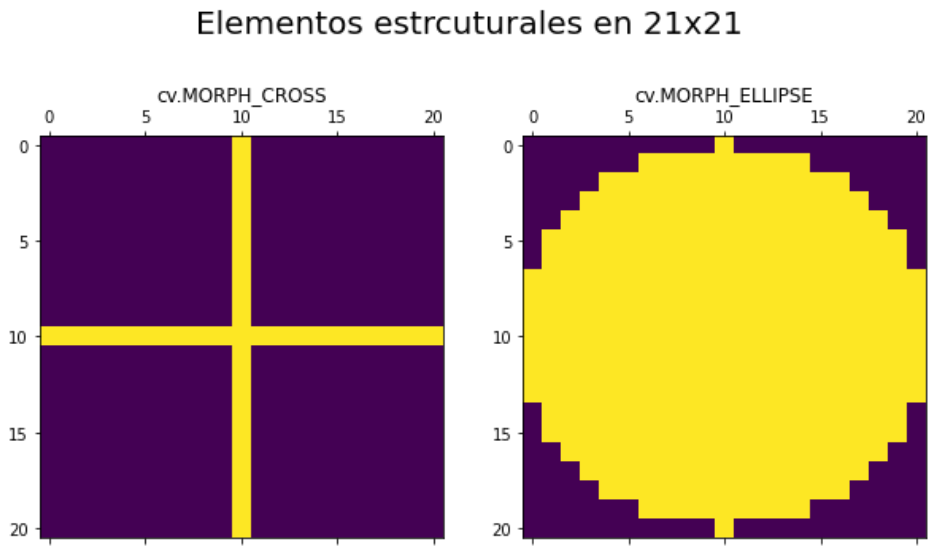


Ilustración 4. cv.MORPH_CROSS, cv.MORPH_ELLIPSE. Elaboración propia.

3.3.2 Operaciones morfológicas

Las operaciones morfológicas son utilizadas para eliminar ruido en imágenes binarias. Consisten en la combinación de las operaciones básicas de erosión y dilatación.

3.3.2.1 Cierre (Closing)

La operación de cierre es útil para reducir tapar los agujeros interiores de una figura. Consiste en realizar una operación de dilatación seguida de una de erosión.

En el caso que nos ocupa, esta operación será clave para eliminar agujeros creados en el interior de la máscara por los jugadores y por las líneas blancas del campo que cortan la imagen y distorsionan el contorno del campo.

Ilustración 5. Ejemplo operación de closing. (OpenCV, s.f.).

3.3.2.2 Apertura (Opening)

A diferencia que en la operación de cierre, en este caso se realizará en primer lugar una operación de erosión seguida de una de dilatación. De este modo se elimina el ruido alrededor de la máscara.

Ilustración 6. Ejemplo operación de opening. (OpenCV, s.f.).

3.3.3 Implementación sobre el caso de uso

Una vez aplicada la máscara de verdes se realiza una operación de cierre de modo que la zona del terreno de juego se limpie del ruido proveniente de los jugadores o las líneas del campo para obtener así el contorno del campo de manera limpia.

Una vez realizada la operación morfológica de *closing* la imagen tiene un tratamiento suficiente para asegurar la selección de todo el terreno de juego sin ruido seleccionando el mayor contorno encontrado en la imagen. El máximo contorno servirá como máscara final para eliminar todos los píxeles exteriores al terreno de juego.

En la Ilustración 6 se observa como la máscara no cumple los requisitos propuestos si no es previamente tratada con la operación de cierre debido a los huecos generados por las líneas blancas del campo y los jugadores que se encuentran cercanos a los límites del campo.

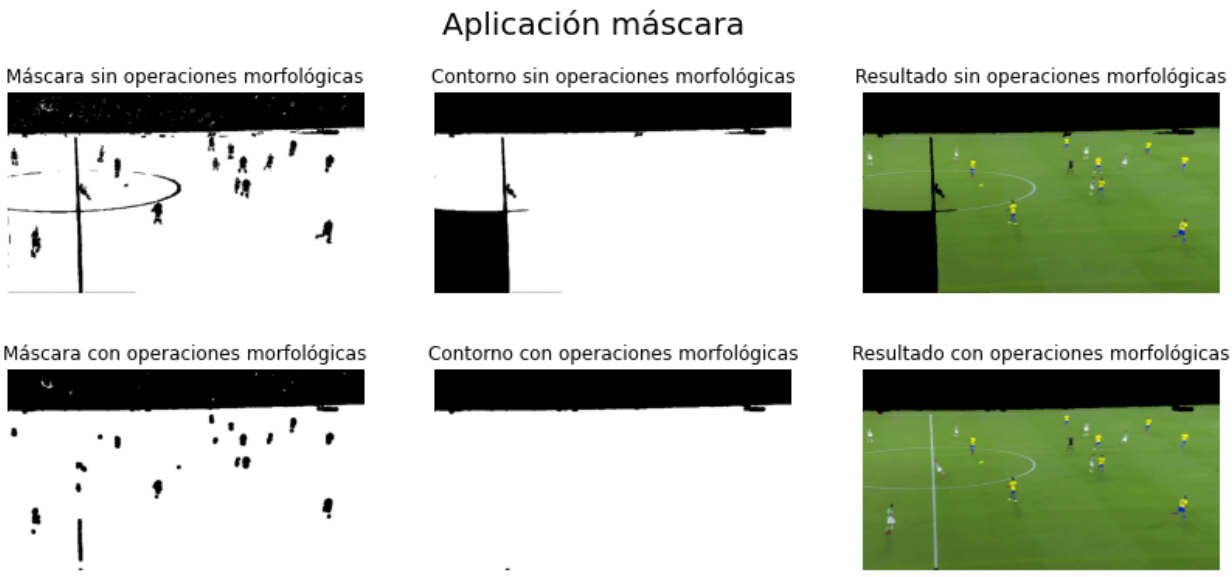


Ilustración 7. Comparación de la aplicación de la máscara con operaciones morfológicas y sin ellas. Elaboración propia.

Capítulo 4 - Detección de líneas

En la última década la detección de líneas y objetos en una imagen ha sido uno de los temas de estudio más recurrentes. Es por ello que se han desarrollado distintos algoritmos buscando la solución óptima al problema variando su rendimiento y su precisión.

4.1 Detección de píxeles blancos

Una vez eliminado el fondo del frame, será necesario realizar otro tratamiento sobre la imagen con el fin de aislar los píxeles blancos que en su mayoría corresponderán a las líneas del campo.

4.1.1 Operación morfológica Top-Hat

Para la obtención de los píxeles blancos que corresponden a las líneas del campo se realiza otra operación morfológica, en este caso la llamada Top-Hat.

Esta operación consiste en realizar la diferencia entre la imagen original y la operación de apertura sobre la imagen, obteniendo como resultado los elementos de la imagen más pequeños que el elemento estructural y más brillantes que los que lo rodean.

Es decir, esta operación aísla los píxeles más brillantes correspondientes a las líneas del campo para obtener así una imagen tratada a partir de la cual se podrán detectar solo las líneas correspondientes al terreno de juego.

4.1.2 Implementación sobre el caso de uso

En primer lugar, la imagen resultada del capítulo anterior se pasa a una escala de grises con el objetivo de destacar de una forma más clara los píxeles blancos sobre el resto.

Una vez realizado este primer paso se procede a la aplicación de la operación Top-Hat con un elemento estructural de 15x15, resultando una imagen con un tratamiento idóneo para la posterior aplicación del algoritmo para la detección de líneas.

Ilustración 8. Proceso de aplicación Top-Hat. Elaboración propia.

En la salida de la operación (última imagen de la Ilustración 7), a simple vista no se aprecia la detección completa de los píxeles blancos, sin embargo, al tratarse de una escala de grises, aunque la diferencia con el color negro es mínima, a la hora de la aplicación del algoritmo para la detección de líneas y el almacenamiento de los píxeles blancos de la imagen, la diferencia es suficiente para un resultado óptimo.

4.2 Transformada de Hough

Para garantizar la comprensión de los métodos de detección de líneas es imprescindible conocer la Transformada de Hough.

Según (Fisher, Perkins, Ashley, & Erik, 2003) se trata de una técnica utilizada para la detección de formas dentro de una imagen. Normalmente utilizada para la detección de formas simples como líneas, círculos o elipses.

Esta técnica se basa en la representación de rectas en forma polar, es decir:

$$x\ \cos\theta + y \sin\theta = \rho$$

Cada punto en el plano x-y será representado en el Espacio de Hough como una curva sinusoidal, correspondiendo a todas las rectas del plano que pasan por ese punto. Es por esto que el punto de corte entre dos curvas que representan dos puntos distintos en el plano x-y, representará la línea recta que las une. (Lee, 2020)

4.3 Detección de líneas

El algoritmo final mapea todos los puntos de la imagen considerados bordes de algún objeto al Espacio de Hough. Como si varios puntos de la imagen pertenecen a la misma línea en la representación sobre el Espacio de Hough tendrán todos ellos un punto de corte común. El algoritmo detectará una línea donde el número de curvas que cortan en un punto común sea mayor que un mínimo marcado.

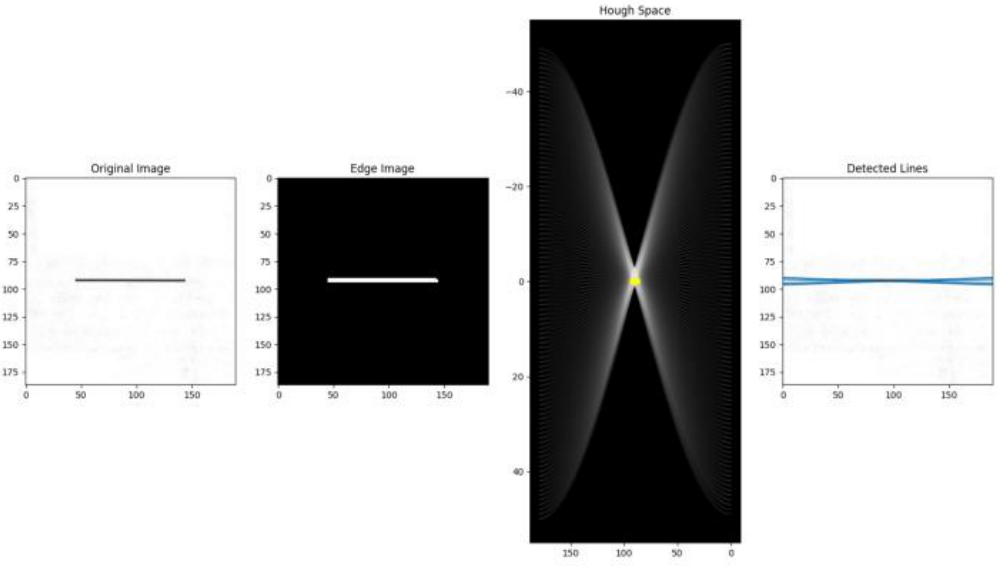


Ilustración 9. Proceso transformada de Hough. (Lee, 2020)

A continuación se muestran tres funciones de la librería OpenCV para la detección de líneas que se evaluarán para determinar cuál se adapta mejor a las necesidades del problema.

4.3.1 HoughLines() OpenCV

Esta función de OpenCV aplica el algoritmo de la transformada de Hough de la siguiente manera: (Lee, 2020)

1. Selección del rango para Rho y Theta (Por defecto)
2. Creación de un array 2D como acumulador con dimensión (nº máximo rho, nº máximo theta)
3. Detección de bordes en la imagen (En este caso se usa el algoritmo Canny)
4. Para cada píxel correspondiente a un borde se calcula su rho y theta y se suma al acumulador
5. Recorriendo todos los valores del acumulador para cada uno que supere un valor mínimo se considera una recta detectada

4.3.2 HoughLinesP() OpenCV

Se trata de una función que implementa el algoritmo Probabilistic Hough Transform desarrollado por (J. Matas, C. Galambos, & J. Kittler, 2000). El cual utiliza un conjunto de píxeles aleatorio con el objetivo de no evaluar todos los píxeles de la imagen y obtener así un rendimiento mejor que el algoritmo original.

4.3.3 Fast Line Detector

El último de los métodos disponibles en la biblioteca OpenCV es el llamado Fast Line Detector. En (Han Lee, Lee, Zhang, Lim, & Hong Su, 2014) se explica el algoritmo en el que está basada esta solución. En rasgos generales se aplica un algoritmo de verificación geométrica con unos resultados a nivel de acierto en la detección y eficacia muy positivos.

4.4 Comparativa y elección del método

Se realizó un estudio comparativo entre los métodos de detección de líneas expuestos en las anteriores secciones. Dando como resultado la elección del Fast Line Detector por encima de los métodos basados en la transformada de Hough por distintos motivos.

En primer lugar, el método HoughLines() quedó descartado por su falta de eficacia. Al valorar todos los píxeles que forman parte de bordes dentro de una imagen de las características que se han expuesto anteriormente, los tiempos de detección resultan muy elevados por lo que intentar acercar al tiempo real la solución del problema resultaría muy complicado. En la propia documentación de la librería (OpenCV, s.f.) se ofrece el método HoughLinesP() como una solución con un rendimiento mejorado.

Por otro lado, se realizaron pruebas sobre el caso de uso con el método probabilístico de la Transformada de Hough, el cual, al usar un conjunto de píxeles aleatorio de tamaño menor al utilizado por el método original, el tiempo de ejecución fue aceptable. Y, aunque el comportamiento en la detección de líneas dentro del terreno de juego fue bueno, se detectaron inestabilidades muy frecuentes en los bordes del frame, así como en los bordes generados por la aplicación de la máscara de fondo. En la Ilustración 9 se

puede observar líneas verticales inexistentes detectadas en el borde izquierdo de la imagen, así como una línea detrás de la portería correspondiente al borde de la máscara del fondo aplicada en el capítulo anterior.

La técnica utilizada por el método Fast Line Detector ha sido la que mejores resultados ha arrojado, encontrando equilibrio entre tiempo de ejecución y acierto en la detección. Comparando la Ilustración 10 y 9, se observa como las irregularidades mencionadas desaparecen siendo una detección mucho más estable.

Ilustración 10. HoughLinesP(). Elaboración propia

Ilustración 11. Fast Line Detector. Elaboración propia.

4.5 Implementación sobre el caso de uso

La aplicación de cualquiera de estos métodos no es trivial. Se hace necesaria un correcto ajuste de los parámetros configurables en cada uno de ellos. La estrategia adoptada ha sido la de minimizar los falsos positivos, es decir, aunque aumente la posibilidad de que líneas del campo no sean detectadas se minimiza el riesgo de detección de líneas correspondientes a sombras en el campo o incluso a jugadores alineados en un momento dado. Se ha optado por esta estrategia al entender muy costosas las identificaciones de esos falsos positivos.

Para esto, en el caso del Fast Line Detector existen dos parámetros configurables que cobran especial importancia:

- *length_threshold:* Longitud mínima de la línea detectada
- *distance_threshold:* Distancia máxima entre píxeles a partir de la cual se descarta que dos píxeles formen la misma línea

En la Ilustración 11 se observa como con un valor pequeño de length_threshold todas las líneas correspondientes al terreno de juego se detectan con mejor rendimiento que en la Ilustración 10. Sin embargo, se observa como además de esas líneas se genera

mucho ruido con líneas detectadas en los contornos de los jugadores y en las inmediaciones del terreno de juego.

Ilustración 12. Length_threshold=25. Elaboración propia

Gracias a la configuración de estos dos valores se ha conseguido cumplir con la estrategia marcada de forma que solo sean detectadas líneas de una mínima longitud considerable y con una distancia entre píxeles grande. Esto hace que se obtengan buenos resultados al ser las líneas del campo, gracias al tratamiento de imagen previo comentado, las líneas identificables con mayor nitidez.

Capítulo 5 - Refinamiento y clusterización

Se observa en las ilustraciones 9, 10 y 11 del apartado anterior como el resultado de la detección de cada una de las líneas del terreno de juego corresponde a un conjunto de líneas normalmente mayor que uno. Para que la relación de líneas detectadas y líneas del terreno de juego sea de 1:1 y no existan varias líneas que representan una única línea del terreno de juego se realiza un proceso de refinamiento y clusterización de modo que cada línea del terreno de juego esté representada por una única línea del conjunto detectado.

5.1 Refinamiento

En función de ciertas variables dependientes del detector de líneas, así como el grosor de cada una de las líneas del terreno de juego en función de su lejanía con la cámara hacen que la detección de líneas no tenga siempre una precisión razonable y a partir de cierto punto la línea detectada se desvíe de la línea que está representando.

En la Ilustración 12 se observa como en la detección de la línea de medio campo, el conjunto de líneas que la representan tienen una pendiente distinta, e incluso si se prolongaran algunas coincidirían en pocos píxeles con la línea que representan y por tanto siendo una representación poco precisa.

Ilustración 13. Elaboración propia

El proceso de refinamiento del conjunto de líneas detectadas busca conseguir la máxima precisión en la representación de las líneas detectadas. Es decir, que el máximo número de píxeles correspondientes a una línea del terreno de juego coincidan con las líneas que lo representan. Solucionando así el problema expuesto en la Ilustración 12.

Para ello, se almacena en una matriz lógica de dimensiones iguales a la imagen los píxeles que corresponden a líneas del campo. La lógica utilizada en este proceso se ha basado en la propuesta realizada por (Ramirez Pereira Duarte, 2015).

5.1.1 Espacio de color YCbCr

Utilizando como entrada la imagen tratada con la operación Top Hat mencionada en el Capítulo 3 de este trabajo, se realiza una conversión al espacio de color YCbCr.

Esta conversión es útil al pasar a un espacio de color en el cual se puede aislar la componente luma (Y) responsable de la codificación de la luminosidad de la imagen. Como la imagen de entrada tiene aplicada la operación Top-Hat y está en escala de grises, se puede afirmar que los píxeles con una luminosidad considerable serán los correspondientes a los píxeles blancos de la imagen.

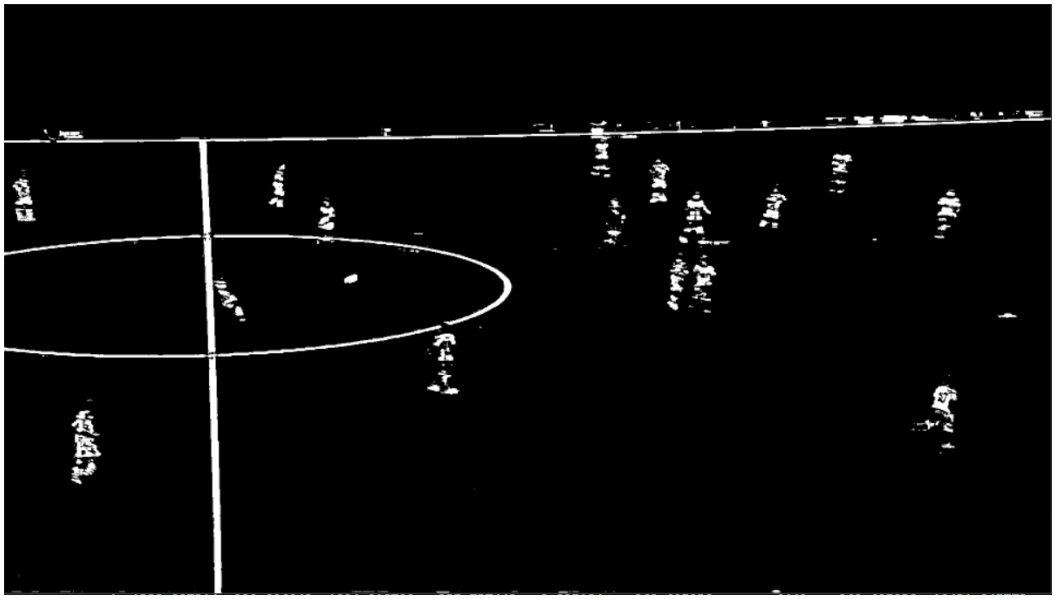


Ilustración 14. Píxeles blancos detectados en la imagen. Elaboración propia.

En la Ilustración 13 observamos el resultado del proceso anterior. Como se verá en las próximas secciones es crítico de cara al rendimiento del refinamiento y la clusterización,

así como para su resultado el reducir el ruido con píxeles blancos correspondientes a las vallas publicitarias, camisetas de jugadores, etc. Para ello se aplicará a la matriz una lógica que determine que píxeles pertenecen a una línea y cuáles no. Como ejemplo de esta lógica, tratándose de una línea horizontal, un píxel que sea considerado blanco perteneciente a la línea tendrá los píxeles horizontales vecinos blancos, sin embargo, los píxeles verticales cercanos no serán blancos. Esta misma lógica se aplica para líneas verticales.

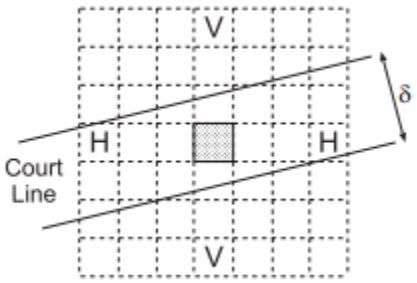


Ilustración 15. Detección de píxeles pertenecientes a líneas. (Ramirez Pereira Duarte, 2015)

Una vez aplicadas las dos condiciones a la matriz lógica de la imagen, se obtienen los píxeles blancos pertenecientes a líneas de la imagen. En la Ilustración 15 se observa una mejoría en relación con la Ilustración 14.

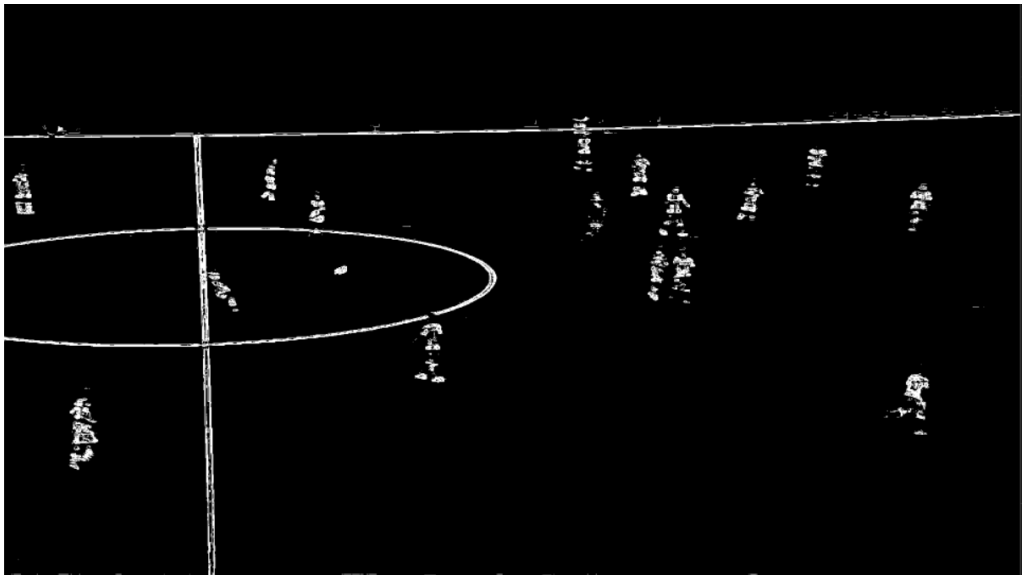


Ilustración 16. Detección de píxeles pertenecientes a líneas. Elaboración propia.

5.1.2 Regresión lineal

Una vez obtenida la matriz con los píxeles pertenecientes a las líneas del campo se realiza el refinamiento de cada una de las líneas detectadas mediante un proceso de regresión lineal.

La regresión lineal (CurveFit.com, s.f.) analiza la relación entre dos variables para encontrar la línea que mejor predice un valor Y frente a X. Este modelo usa la técnica de mínimos cuadrados para encontrarla. En el caso que nos ocupa, la representación de cada píxel blanco perteneciente a una línea del campo de juego nos hará mediante la regresión lineal conseguir una precisión mayor de las líneas detectadas previamente.

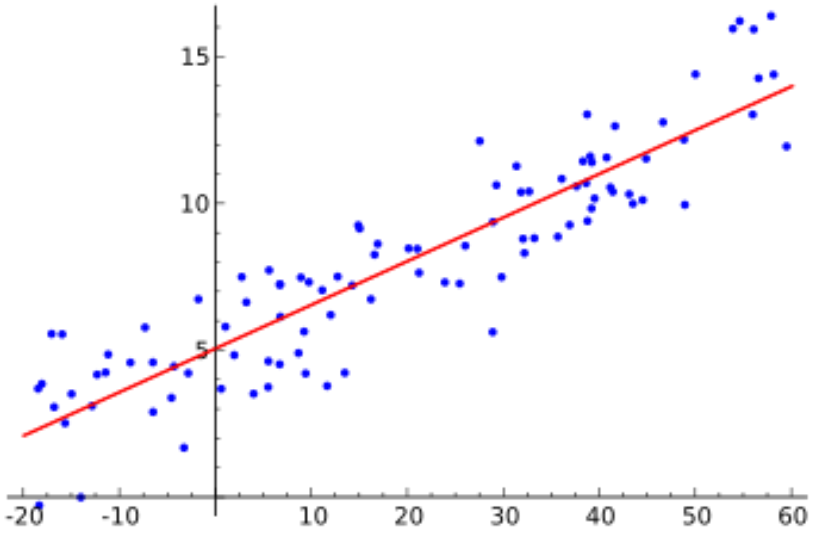


Ilustración 17. Wikipedia.org

En primer lugar, se calcula la distancia de cada uno de los píxeles pertenecientes a las líneas del terreno de juego con respecto a la línea detectada. Seleccionando en último lugar los píxeles que estén a una distancia menor a un cierto offset parametrizado. De esta forma, gracias a la detección de líneas se obtiene el conjunto de píxeles pertenecientes a una línea del campo. Este conjunto es introducido en el modelo de regresión lineal dando por resultado una línea que representará la línea del terreno de juego en cuestión.

5.2 Clusterización

La relación de líneas detectadas frente a la línea que representan es normalmente mayor que uno por lo comentado anteriormente. Es por esto que se hace necesario someter al conjunto de líneas detectadas a un proceso de clusterización de forma que la relación de líneas detectadas sea 1:1 frente a las líneas que representan.

Para este proceso, se han seguido dos estrategias distintas con el fin de reducir el porcentaje de error.

5.2.1 Rho y Theta

Anteriormente se mencionó la forma polar de representación de una recta con los parámetros rho y theta. Estos parámetros se utilizan para comparar lo similares que son dos rectas en forma polar. Para ello se les asignará un mismo identificador a las rectas que tengan una rho y una theta menor a un intervalo parametrizable.

5.2.2 Cortes con la imagen

La segunda estrategia utilizada para la clusterización de las líneas detectadas se enfoca en los puntos de corte de cada recta con la imagen. Cada recta detectada tendrá dos puntos de corte con la imagen, de tal modo que mediante la ecuación explícita de la recta ($y = mx + n$) se podrán calcular los puntos de corte de cada recta y agrupar bajo un mismo identificador a aquellas cuya distancia entre los puntos de corte sea menos que un cierto parámetro de offset.

5.2.3 Selección final

Después de cada una de las estrategias anteriores se debe seleccionar entre todas las rectas agrupadas bajo un mismo identificador cual será la recta candidata seleccionada. Para ello, apoyándonos en el cálculo de la recta mediante regresión lineal se almacena como información adicional de la recta el número de píxeles a partir de los cuáles se realizó la operación como "score". La recta seleccionada será para cada uno de los grupos la que tenga el mayor "score" al ser la recta con más píxeles blancos que pasan por ella, y por tanto, la que tiene una precisión mayor que el resto.

El proceso de refinamiento y clusterización de cada una de las líneas se puede repetir varias veces con el fin de conseguir una precisión mayor. Aun así, se debe tener en cuenta la penalización en rendimiento del proceso en cada una de las iteraciones, sobre todo en las primeras al tener aún un número elevado de rectas candidatas.

Ilustración 18. Detección de líneas. Elaboración propia

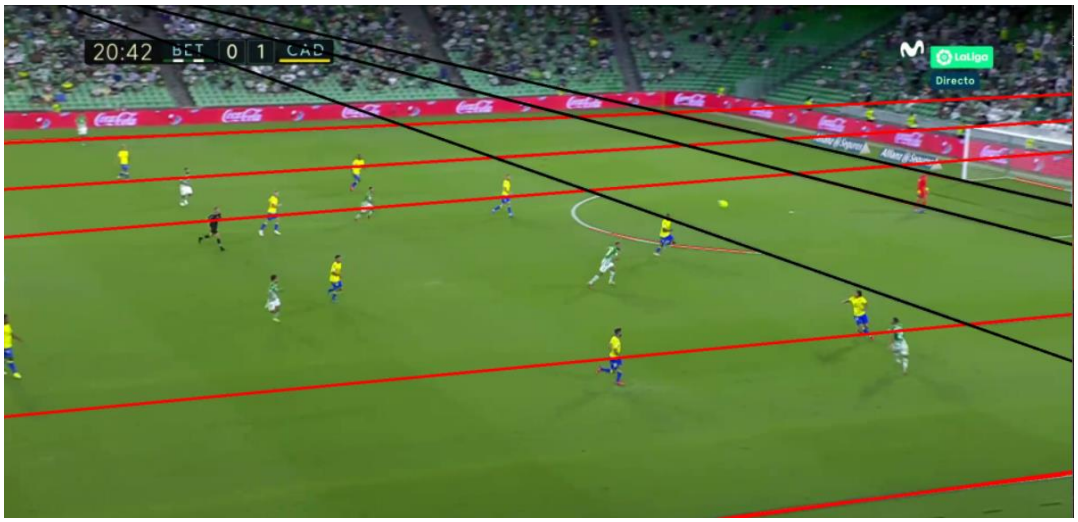


Ilustración 19. Aplicado refinamiento y clusterización. Elaboración propia

Como se puede observar comparando las Ilustraciones 17 y 18, tras el proceso de refinamiento y clusterización descrito, se dan unos resultados aceptables con precisión y tan solo una recta candidata por cada una de las líneas del terreno de juego.

Capítulo 6 - Identificación de líneas

Una vez obtenidas las líneas del campo maximizando su precisión y rendimiento, se buscará la identificación de ciertas características relevantes de cara a encontrar las líneas que hayan sido falsos positivos, así como la eliminación de líneas correspondientes al círculo central y la semiluna de cada una de las porterías, tema que no resulta trivial al tratarse de arcos y no de líneas rectas.

6.1 Orientación de rectas

Es necesaria la identificación de las líneas como verticales y horizontales. Este proceso no es trivial, ya que por la perspectiva de la imagen se encuentran situaciones en las que ambos tipos de rectas tienen el mismo signo de pendiente. Además, el tiro de cámara hacia la parte derecha o izquierda del campo hace que los parámetros tenidos en cuenta para la determinación de la orientación de cada recta no sean iguales.

Antes del procesamiento de cada frame se determinará si la cámara está orientada hacia la derecha o izquierda del campo. Esto se ha realizado de forma manual en cada uno de los frames analizados, entendiendo fuera del alcance del trabajo este cálculo y orientándolo a posibles mejoras en un trabajo futuro. Esto podría ser resuelto mediante diversas soluciones. Una de ellas, la implantación de una red neuronal con una gran cantidad de imágenes de retransmisiones deportivas que califique la orientación de la cámara en izquierda, derecha o centro.

Para determinar la orientación de cada una de las rectas se han utilizado los atributos rho y theta de cada una de las rectas en forma polar.

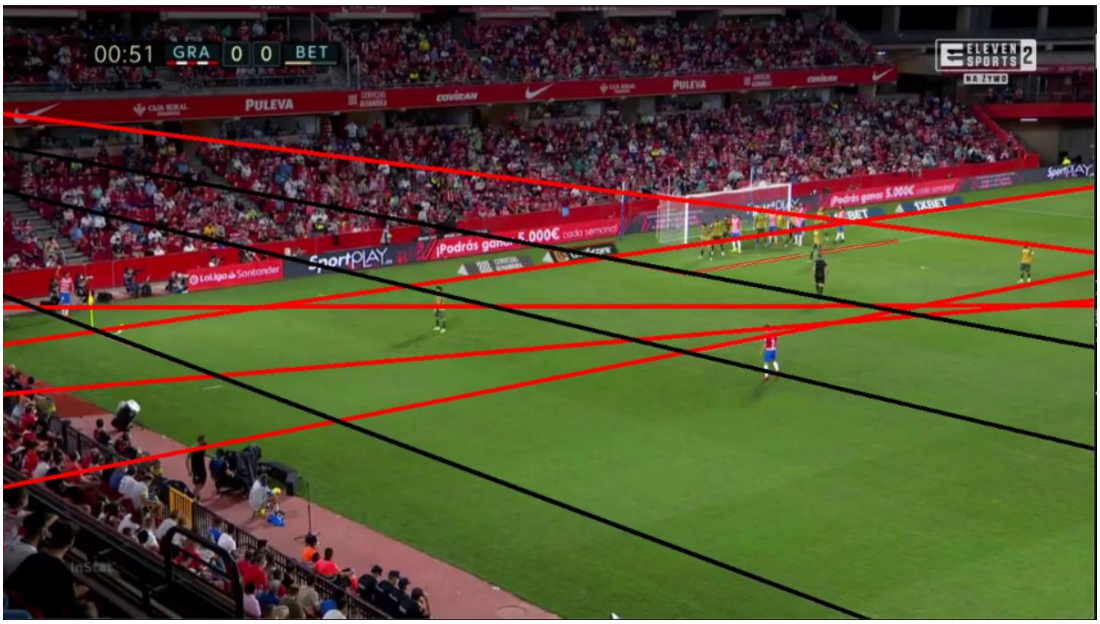


Ilustración 20. Tiro de cámara izquierdo. En rojo líneas horizontales. Elaboración propia.

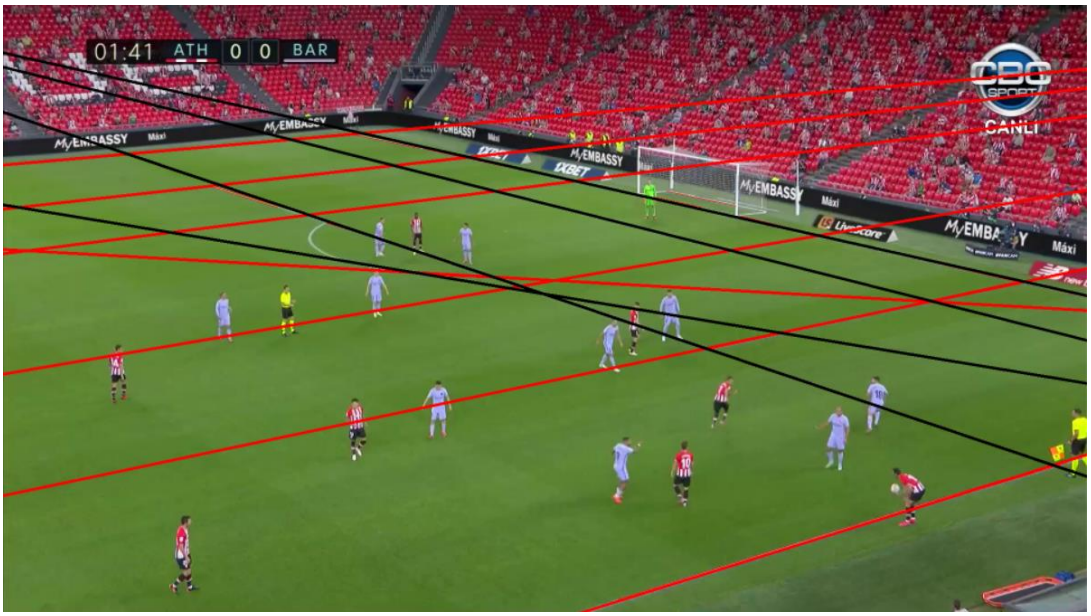


Ilustración 21. Tiro de cámara derecho. En rojo líneas horizontales. Elaboración propia.

Se puede observar en la Ilustración 19 como con los rangos de orientación para el tiro de cámara derecho la determinación de la orientación de las rectas detectadas no es correcto. Sin embargo, en la Ilustración 20 esos mismos rangos para rho y theta dan un resultado óptimo en la clasificación de rectas en función de su orientación.

6.2 Identificación de líneas

La identificación de ciertas líneas del terreno de juego resultan muy importantes en su utilización como referencia para la eliminación de líneas auxiliares como las detectadas sobre el círculo central o la semiluna del área de penalti.

6.2.1 Línea de medio campo

A través de la observación de numerosos frames, se ha llegado a la conclusión de poder identificar la línea de medio campo en relación con su pendiente. Esta línea es la única representada siempre con una pendiente mayor que 30, y no existen otras líneas que puedan tener una pendiente cercana. Esto es debido a la ubicación de la cámara en las retransmisiones deportivas prácticamente alineado con esta línea, lo que la hace tener una pendiente muy pronunciada. Sin embargo, el resto de rectas verticales del terreno de juego se encuentran en una ubicación muy lateral con respecto a la posición de la cámara lo que las hace tener una pendiente menos pronunciada.

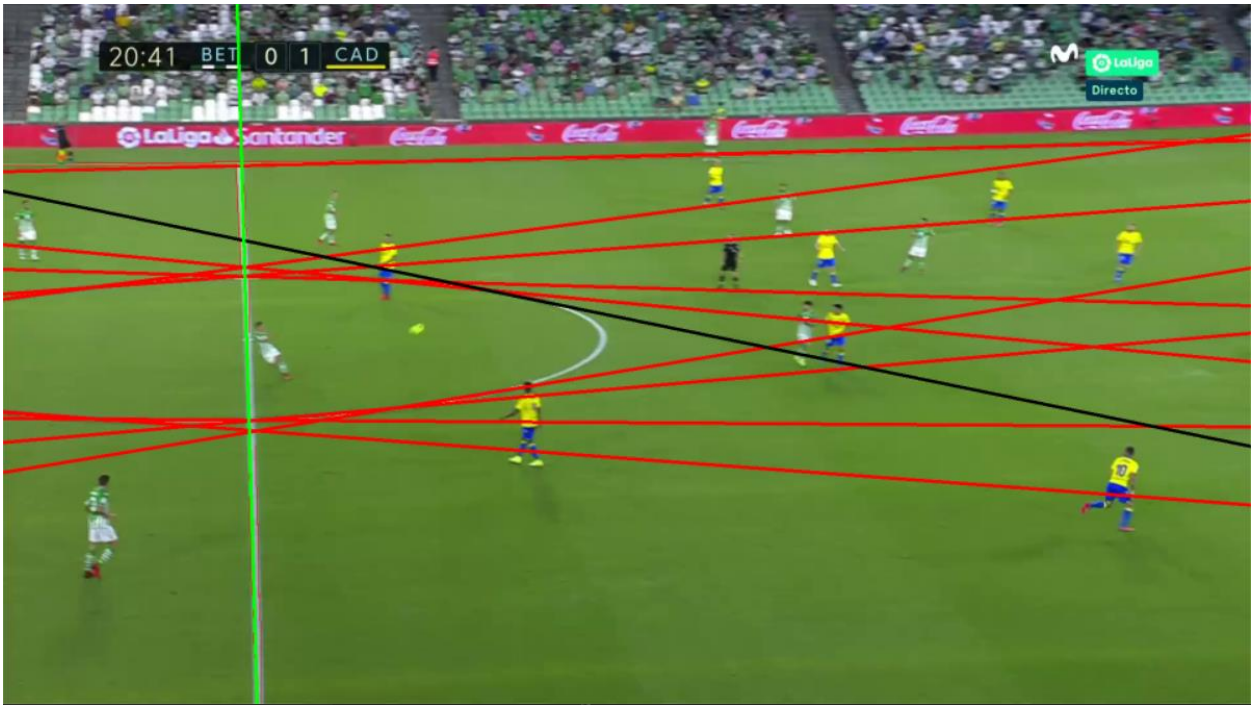


Ilustración 22. En verde la línea detectada de medio campo. Elaboración propia.

Se observa en la Ilustración 21 la diferencia de pendiente entre la línea de medio campo pintada en verde y el resto de líneas detectadas en la imagen.

6.2.2 Línea de área de penalti

La identificación de la línea de área de penalti resulta clave para los objetivos mencionados anteriormente. Una vez detectada la orientación de cada línea se puede afirmar con seguridad que tras la línea de medio campo, esta será la línea vertical con una pendiente mayor. Es por esto que la identificación de esta línea se realizará posteriormente a la identificación de la línea de medio campo, para que esta última sea descartada como candidata en caso de encontrarse dentro de la imagen.

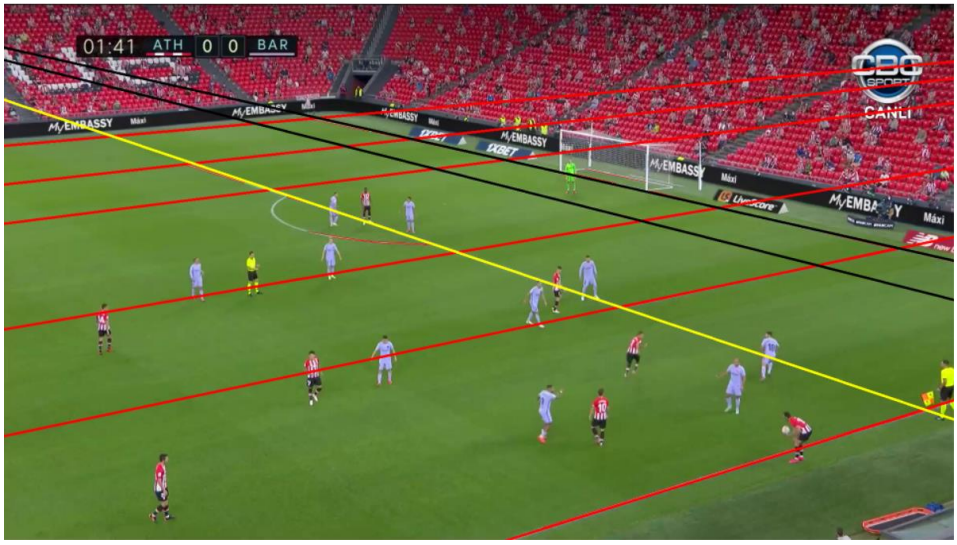


Ilustración 23. En verde la línea detectada del área de penalti. Elaboración propia

Ilustración 24. Detección de línea de medio campo y línea de área de penalti en el mismo frame. Elaboración propia.

6.3 Circulo central y semiluna

La identificación de las líneas mencionadas en el apartado anterior resulta de gran relevancia a la hora de eliminar las líneas detectadas correspondientes al círculo central y a la semiluna del área de penalti.

6.3.1 Círculo central

En el caso del círculo central, conociendo la ubicación de la línea de medio campo, se realiza el cálculo medio de los puntos de la recta para el eje X. Por otro lado, se calcula una aproximación del valor que tendría el eje Y en el centro del campo. Esto no resulta trivial ya que la imagen puede contener parte de la grada en la parte superior o inferior, haciendo que en la mitad del eje Y no se encuentre el círculo central. Es por ello, que el cálculo de ese punto se realiza teniendo en cuenta el mayor y menor valor del contorno seleccionado en la máscara del campo, para así, calcular el punto medio correspondiente exclusivamente a la zona del terreno de juego de la imagen.

Gracias a esos dos valores, con un offset parametrizable se crea una zona de seguridad dentro de la cual cualquier línea detectada será detectada por entenderse relacionada con las líneas del círculo central.

En posteriores mejoras, se plantea la posibilidad de encontrar la Elipse generada a partir de los puntos descartados, para tener una referencia más a la hora del cálculo de la homografía de la imagen.

Se pueden observar los resultados de este proceso comparando las Ilustraciones 21 y 23, teniendo la segunda imagen las líneas del círculo central detectadas pero eliminadas mediante el método mencionado.

6.3.2 Semiluna

Para descartar las líneas detectadas sobre la semiluna del área de penalti, se sigue una estrategia parecida a la descrita en el apartado anterior. En este caso es necesario eliminar la línea de medio campo si estuviese detectada del conjunto de líneas a valorar. De este modo, serán descartadas todas las rectas verticales que se encuentren a la izquierda de la recta identificada como la de área de penalti.

En el caso de las rectas horizontales, serán descartadas todas aquellas con una pendiente de mismo signo que las rectas verticales detectadas. Esto se debe a los parámetros con los que se ajustó la detección de líneas en la sección 4.3, de modo que al tener la condición de una cierta longitud solo podrán ser detectadas aquellas líneas que se encuentran en la parte inferior de la semiluna como se observa en la Ilustración 24.

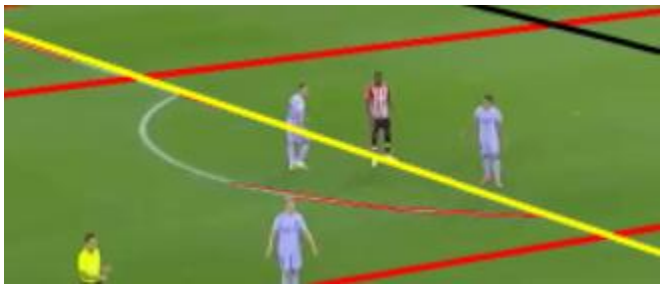


Ilustración 25. Detección de líneas en la parte inferior de la semiluna. Elaboración propia.

Capítulo 7 - Homografía

Tras el resultado de las secciones anteriores, se obtiene un conjunto de líneas identificadas de forma estable a partir de las cuales se podrá cada uno de los frames procesados a un plano 2D con un modelo del terreno de juego.

La matriz de homografía es una matriz de 3x3 la cual relaciona cada uno de los puntos de la imagen con su representación en el modelo. Es decir, buscaremos eliminar el efecto de la perspectiva de la imagen obteniendo una representación de esta en el plano como si de una vista cenital se tratase. De este modo, tras la identificación de jugadores se podrá tener una visión del terreno de juego con utilidad en aspectos del juego como la táctica o la distancia recorrida por cada uno de los jugadores.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

Ilustración 26. Relación de puntos en dos planos mediante la matriz de homografía.

Para el cálculo de esta matriz, será necesario utilizando la función de la librería OpenCv "findHomography()" relacionar al menos cuatro puntos de la imagen procesada con cuatro puntos del modelo. De esta forma cualquier punto dado de la imagen será relacionado con un punto dentro del plano 2D del modelo.

Aunque fuera del alcance de este trabajo, se ha decidido realizar una aproximación al cálculo de la homografía con el fin de obtener una introducción a la utilidad de la detección de líneas explicada acercándose al fin último para el que el objetivo del proyecto cobra sentido.

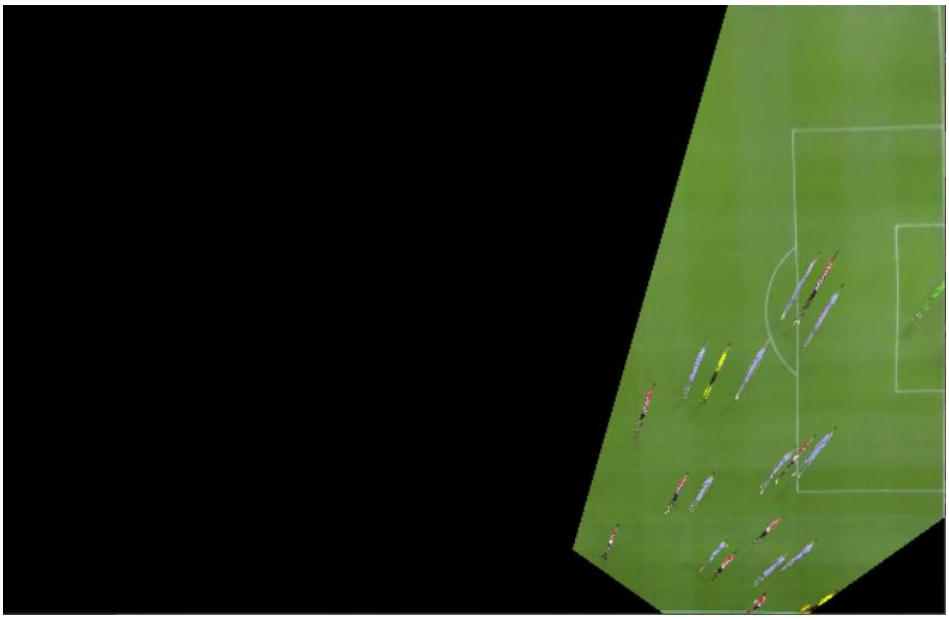


Ilustración 27. Transformación a modelo 2D. Elaboración propia.

En la Ilustración 25 se observa el resultado de transformar un frame de la retransmisión de vídeo mediante el cálculo de la matriz de homografía, de modo que la imagen resultada está representada en un modelo 2D con las proporciones del terreno de juego.

A partir de esta solución y el cálculo de la homografía en tiempo real durante la retransmisión de un partido de fútbol, unido a la detección e identificación de los jugadores aporta una gran cantidad de información de utilidad para distintos sectores relacionados el fútbol. Entre ellas se pueden destacar algunas como la utilización de esta tecnología para reducir el tiempo en determinar un fuera de juego, así como las distancias recorridas por los jugadores en el ámbito físico, o bien en el ámbito táctico una visión mucho menos sesgada del juego.

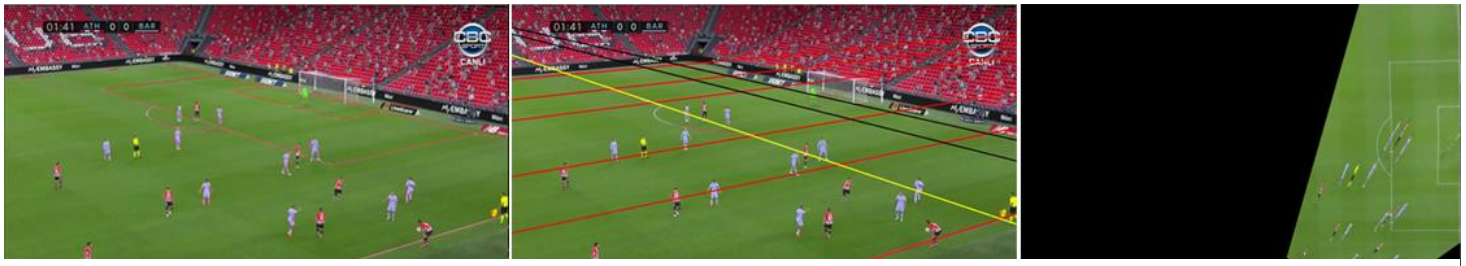


Ilustración 28. Proceso completo desde la detección de líneas hasta la homografía. Elaboración propia.

Capítulo 8 - Conclusiones y trabajo futuro

La detección automática de las líneas del terreno de juego ha tenido un resultado satisfactorio con la clara premisa de minimizar los falsos positivos de forma que la solución facilite los procesos posteriores que utilicen este método.

Como se ha mencionado anteriormente, el siguiente paso y la línea de trabajo futuro más importante será la de conseguir la transformación de cada uno de los frames al modelo 2D mediante homografía en tiempo real.

Para la realización de la homografía es necesario relacionar al menos cuatro puntos de la imagen procesada con cuatro puntos del modelo. Partiendo de esa base, el entendimiento de la solución propuesta para el cálculo de la homografía hará que las mejoras introducidas en el proceso descrito sean de un tipo u otro.

Es por esto que cobra una importancia mayúscula la forma en la que se decida relacionar los puntos de la imagen con los del modelo.

8.1 Identificación de todas las líneas detectadas

Una de las soluciones propuestas es relacionar los puntos de corte de las líneas detectadas con los del modelo teniendo una identificación completa de cada una de las líneas detectadas en el frame. Es decir, conocer a que línea del modelo corresponde cada una de las líneas detectadas.

Esta solución haría necesaria una mejora en la identificación de líneas. En el proceso descrito, tan solo las líneas de medio campo y área de penalti son identificadas (por la necesidad de eliminar las líneas detectadas sobre el círculo central y la semiluna del área de penalti). Sin embargo, se tendrían que buscar distintas relaciones para la identificación todas las líneas detectadas teniendo en cuenta que no siempre se puede dar por hecha la identificación de una línea en función de otra al no estar garantizada la identificación de ninguna de las que puedan ser tomadas como referencia.

Una vez conseguida la identificación de al menos dos líneas verticales y dos horizontales, será trivial encontrar la matriz de homografía relacionando con seguridad los puntos de cada una de ellas.

El principal inconveniente de esta solución es la baja tolerancia a fallos. Si alguna de las líneas utilizadas para el cálculo de la homografía estuviera mal identificada, la homografía sería incorrecta.

8.2 Estimación de la homografía a partir de las características de los puntos de corte en la imagen

Esta solución se basa en la propuesta realizada por (Kasai, y otros). Los autores proponen la identificación de los puntos de corte mediante un patrón que tenga en cuenta la direccionalidad de las líneas del campo.

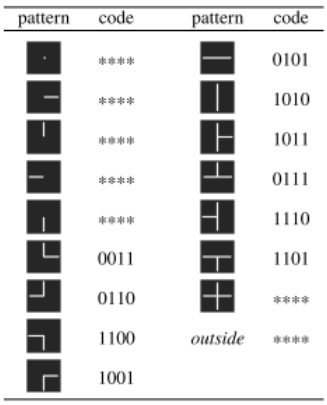


Ilustración 29. Patrón de reconocimiento de puntos de corte. (Kasai, y otros).

Una vez obtenido el patrón de cada uno de los cuatro puntos de corte seleccionados, se cruzará mediante combinatoria con todos los posibles puntos del modelo que tengan un patrón similar. De este modo, se obtienen diversas homografías que en última instancia se valoran mediante una función de puntuación para quedarse con la que mejor puntuación tenga, y, por tanto, mejor represente todos los puntos de la imagen.

A diferencia de la anterior solución, en este caso la tolerancia a fallos es mayor, ya que el cálculo de la puntuación de cada una de las homografías determinará cuál es la idónea sin necesidad de relacionar directamente cada línea seleccionada con las líneas del modelo. Sin embargo, el principal inconveniente observado en este proceso reside en el coste computacional existente en el cálculo de cada homografía para todas las combinaciones posibles. Aunque este se vea reducido por la relación de cada punto con un punto del modelo en función del patrón detectado, en casos en los que

no sea fácilmente detectable el patrón del punto de la imagen, o el punto de corte se de fuera de la imagen (habría que probar con todos los tipos de puntos disponibles), el número de combinaciones distintas es muy elevado haciendo que la solución se aleje de unos tiempos cercanos al cálculo en tiempo real.

Capítulo 9 - Conclusions and future work

The automatic detection of pitch lines has been successful with the clear premise of minimising false positives so that the solution facilitates subsequent processes using this method.

As mentioned above, the next step and the most important line of future work will be to achieve the transformation of each of the frames to the 2D model by means of real-time homography.

In order to perform the homography it is necessary to relate at least four points of the processed image to four points of the model. On this basis, the understanding of the proposed solution for the calculation of the homography will determine the type of improvements introduced in the described process. This is way the way in which you choose to relate the dots in the image to the dots in the model is of maximum relevance.

9.1 Identification of all lines detected

One of the proposed solutions is to relate the cut-off points of the detected lines with those of the model by having a complete identification of each of the lines detected in the frame. I.E. To know to which line of the model each of the detected lines corresponds.

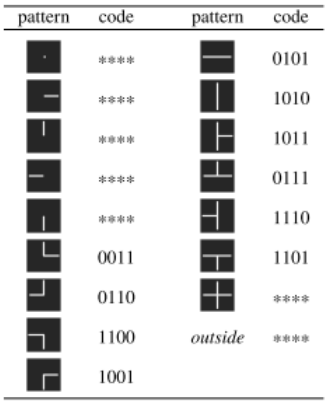
This solution would require an improvement in the identification of lines. In the process described, only the midfield and penalty area lines are identified (due to the need to eliminate the lines detected on the central circle and the penalty area crescent). However, different relationships should be sought for the identification of all the lines detected, bearing in mind that the identification of one line cannot always be taken for granted in relation to another, as the identification of any of the lines that can be taken as a reference is not guaranteed.

Once at least two vertical and two horizontal lines have been identified, it will be trivial to find the homography matrix by reliably relating the points on each of them.

The main drawback of this solution is the low fault tolerance. If any of the lines used for the homography calculation were misidentified, the homography would be incorrect.

9.2 Estimation of homography from the characteristics of the cut-off points in the image

This solution is based on the proposal made by (Kasai, y otros). The authors propose the identification of the cut-off points by means of a pattern that takes into account the directionality of the field lines.



pattern	code	pattern	code
	****		0101
	****		1010
	****		1011
	****		0111
	****		1110
	0011		1101
	0110		****
	1100	*outside*	****
	1001		

Ilustración 30. Cut-off point recognition pattern. (Kasai, y otros).

Once the pattern of each of the four selected cut-off points has been obtained, it will be crossed with all the possible points of the model that have a similar pattern by means of combinatorics. In this way, various homographies are obtained which are ultimately evaluated by means of a scoring function to obtain the one with the best score and, therefore, the one that best represents all the points in the image.

Unlike the previous solution, in this case the fault tolerance is greater, since the calculation of the score of each of the homographs will determine which is the ideal one without the need to directly relate each selected line with the lines of the model. However, the main drawback observed in this process lies in the computational cost of calculating each homography for all possible combinations. Although this is reduced by the relation of each point with a point of the model according to the detected pattern, in cases where the pattern of the point in the image is not easily detectable, or the cut-off point is outside the image (it would be necessary to try with all the types of points available), the number of different combinations is very high, making the solution far from a time close to the real-time calculation.

BIBLIOGRAFÍA

CurveFit.com. (s.f.).

Fernández, L. Á. (s.f.). *Aplicación de filtros morfológicos en imágenes*.

Fisher, R., Perkins, S., Ashley, W., & Erik, W. (2003). *HIPR2*.

Han Lee, J., Lee, S., Zhang, G., Lim, J., & Hong Su, I. (2014). Outdoor Place Recognition in Urban Environments using Straight Lines.

J. Matas, C. Galambos, & J. Kittler. (2000). Robust Detection of Lines Using the Progressive Probabilistic Hough Transform, Computer Vision and Image Understanding.

Kasai, K., Kawakita, K., Kubota, A., Tsurusaki, H., Watanabe, R., & Sugano, M. (s.f.). Robust and Efficient Homography Estimation Using Directional Feature Matching of Court Points for Soccer Field Registration.

Lee, S. (2020). *Towards Data Science.*

Li Sun, G. L. (2009). *FIELD LINES AND PLAYERS DETECTION AND RECOGNITION IN SOCCER VIDEO*.

OpenCV. (s.f.). *Open Source Computer Vision.*

Ramirez Pereira Duarte, M. (2015). *Camera Parameters Extraction Using Real-Time*.