

---

# Proyecto final: Detección automática de la posición del balón en relación con el área durante un partido de Baby fútbol

EL5206-2 Laboratorio de Inteligencia Computacional y Robótica - Primavera  
2023

---

*Profesor:*  
Carlos Navarro  
Martin Adams

*Ayudantes de Laboratorio:*  
Eduardo Jorquera  
Gonzalo Olguín  
Manuel Zamorano  
Vanessa González

*Estudiante:*  
Eduardo Calatayud Muñoz

*Fecha de entrega: Viernes, 1 de diciembre de 2023*

## **Índice.**

<b>1. Objetivo</b>	<b>2</b>
<b>2. Metodología</b>	<b>2</b>
2.1. Toma de imágenes . . . . .	2
2.2. División y reconstrucción del vídeo . . . . .	3
2.3. Diferenciación de zonas en la cancha de fútbol . . . . .	3
2.3.1. Localización de la línea del área . . . . .	3
2.3.2. Creación de máscaras de zonas . . . . .	4
2.4. Detección del balón . . . . .	5
2.5. Detección de intersecciones entre el balón y las áreas . . . . .	6
<b>3. Resultados</b>	<b>7</b>
<b>4. Análisis de los resultados</b>	<b>8</b>
<b>5. Conclusiones</b>	<b>10</b>
<b>Bibliografía</b>	<b>10</b>

**Lista de figuras.**

1. Imágen binaria de las tonalidades objetivo resaltadas (izquierda) e imágen con la línea del área calculada y dibujada en rojo (derecha). Los frames proceden de un partido real en día nublado. . . . .	4
2. Máscaras del área (derecha) y del balón en un frame concreto (izquierda). . . . .	7
3. Frames del resultado de la detección automática durante un partido real en un día nublado. . . . .	8
4. Frames del resultado de la detección automática durante un partido ficticio en un día soleado. . . . .	8
5. Distintos ejemplos de fallos en la detección. . . . .	8
6. En la fila de arriba, se muestra la imagen binarizada (derecha) y con línea (izquierda) de un video enfocando la portería de la izquierda en un día soleado. En la fila de abajo, se puede ver lo mismo pero de la portería de la derecha. . . . .	9

**Lista de tablas.**

## 1. Objetivo

Las decisiones de los jueces siempre han sido una fuente de polémicas en todos los ámbitos de la sociedad y los deportes no iban a ser menos. Muchos deportes ya han introducido tecnologías que ayudan a los árbitros en la toma de decisiones justas e incluso, en ocasiones, esta sustituyendo la labor arbitral, como en el tenis. Uno de los últimos deportes en incorporar esta tecnología ha sido el fútbol. La Fifa ya aprobó en 2012 el uso del sistema de Ojo de Halcón en la línea de gol para evitar los goles fantasma [1] y, en las ligas más grandes del fútbol mundial, el árbitro puede consultar repeticiones para valorar mejor los incidentes del juego.

En el campus de Beauchef de la Universidad de Chile es muy popular la práctica del Baby fútbol, organizándose todos los semestres torneos. El Baby fútbol es una modalidad de fútbol que se juega cinco contra cinco y que tiene unas normas específicas. Una de estas normas que genera una gran cantidad de conflictos es el hecho de que para que un gol suba al marcador, el balón ha de ser chutado desde dentro del área. Si el jugador golpea la pelota desde fuera del área rival y entra sin golpear en nadie dentro de dicha área el gol no cuenta. Debido a la velocidad del juego, no siempre es fácil saber en qué punto de la cancha se ha golpeado el balón.

Por ello, el objetivo de este proyecto es crear una herramienta capaz de detectar si el balón se encuentra dentro o fuera del área durante el juego. De esta forma, el árbitro, en caso de duda, podrá consultar la repetición y decidir fácil y rápidamente si el gol es válido o no.

## 2. Metodología

Para la detección de la zona donde se encuentra el balón durante el juego se realiza una serie de procesamientos sobre las imágenes tomadas, necesarios para obtener un resultado de calidad. Estos procesos se describen en esta sección.

La herramienta se ha desarrollado en Python usando las siguientes librerías:

- OpenCV, para el tratamiento de las imágenes.
- Numpy, para el tratamiento de los datos
- google.colab.patches, para utilizar ciertas funciones de OpenCV en Google Colab
- math, para realizar cálculos matemáticos.
- tqdm, para ver la eficiencia de las funciones.
- torch, para implementar Yolov5.
- shutil y os, para el manejo de los archivos.

### 2.1. Toma de imágenes

Para que un correcto funcionamiento del algoritmo desarrollado, las imágenes deben de ser tomadas siguiendo unas pautas determinadas.

- Los videos se deben grabar a la altura de la zona entre las áreas de la cancha. Si no es así, la detección automática del área fallaría.
- El uso de un trípode es necesario para evitar desencuadres en los frames de la grabación.

Es muy recomendable que en algún frame del vídeo aparezca la cancha sin personas ni objetos para mejorar el resultado de la detección del área.

## 2.2. División y reconstrucción del vídeo

Tras cargar el video a analizar, se divide este en frames para tratar cada sección del video de manera individual. Una vez terminados todos los procedimientos de tratamientos de imágenes, se vuelve a montar el vídeo con el mismo ratio de frames por segundo, para que el resultado sea natural. En este caso, los videos tomados por el teléfono móvil están grabados en 30 fps (frames per second).

Cabe remarcar que para una mejor eficiencia del código, los frames se guardan como imágenes en una carpeta temporal y se cargan para ser analizados. Esto hace que su tratamiento sea más veloz al no guardados en la RAM. Cuando se caraga un nuevo video para su procesamiento, la carpeta anterior se borra y se crea una nueva.

## 2.3. Diferenciación de zonas en la cancha de fútbol

En primer lugar, se debe localizar la línea del área que delimita las zonas donde el gol es válido y no. Como los vídeos se tomaron con ayuda de un trípode, no hace falta calcular el área en todo momento, basta con encontrar la zona en uno de los frames y se utiliza ese valor para el resto de imágenes. En este caso, se toma siempre el primer frame del vídeo, por lo que es recomendable que al comienzo del video no haya muchas figuras sobre la cancha.

### 2.3.1. Localización de la línea del área

La línea del área de la cancha del edificio 850 del campus de Beauchef tiene un color rojiza. Para que nuestro algoritmo detecte de la mejor manera posible esta línea, se resalta la tonalidad de la pintura de la raya en un frame, que preferiblemente no tenga muchos objetos sobre el terreno de juego para evitar confusiones. Para ello, se obtienen las capas de colores del frame seleccionado (azul, verde y roja) y nos quedamos con la resultante de eliminar la capa verde de la azul. Para remarcar aún más la distinción se binariza la imagen aplicando un umbral. Todos los píxeles cuyo valor esté por encima de dicho umbral se les asigna el valor 255 (blanco) y al resto 0 (negro)

Sobre esta imagen tratada se aplica el algoritmo de Canny para detectar los bordes de las figuras resaltadas. El algoritmo de Canny, desarrollado por John F. Canny en 1986, aplica distintas etapas a una imagen para detectar los bordes de los objetos que aparecen en ella [2].

Por último, se aplica la Transformada de Hough Probabilística para sacar la ecuación de la recta que le corresponde a la línea del área. La Transformada de Hough Probabilística es una técnica usada para la detección de figuras que se pueden expresar matemáticamente (rectas, circunferencias, elipses,...) en una imagen. La transformada fue propuesta por Paul Hough en 1962 para detectar rectas y más tarde generalizada para el resto de formas por Richard Duda y Peter Hart [3]. La Transformada de Hough probabilística se basa en usar un subconjunto de los puntos de borde en lugar de todos, como hace la standar, para disminuir la complejidad del problema [4].

La implementación de ambos algoritmos se realiza mediante el uso de la librería de OpenCV para Python.

Para filtrar el resto de líneas de la imagen se ajustan los parámetros de umbral, longitud mínima de la línea y distancia máxima de los huecos de la Transformada de Hough Probabilística. Con la pendiente,  $m$ , y la ordenada en el origen,  $b$ , se obtiene la ecuación de la recta de la forma  $y = mx + b$  y se dibuja una recta que cruza toda la imagen para verificar la buena predicción del algoritmo de la línea del área.

En la figura 1, se pueden observar los resultados de estos procesos sobre el primer frame de un video durante un partido real.

El código comentado para la implementación de este algoritmo se muestra a continuación.

Código 1: Función para resaltar las figuras con tonos rojos de la imagen.

```

1 # Función para resaltar las figuras con tonos rojos de la imagen
2 def resaltar_linea_de_area(imagen):
3     # Se divide la imagen en sus canales y se trata para resaltar el color de la
4     # linea
5     b,g,r = cv2.split(imagen)
6     im_tratada = cv2.subtract(b,g)
7
8     # Aplicar umbralización binaria
9     umbral = 20
10    im_tratada[im_tratada < umbral] = 0
11    im_tratada[im_tratada >= umbral] = 255
12
13    # Mostrar imagen
14    cv2_imshow(im_tratada)
15
16    return(im_tratada)

```



Figura 1: Imagen binaria de las tonalidades objetivo resaltadas (izquierda) e imagen con la línea del área calculada y dibujada en rojo (derecha). Los frames proceden de un partido real en día nublado.

### 2.3.2. Creación de máscaras de zonas

Para poder realizar una comparación de la posición del balón con la del área, es necesario crear unas máscaras binarias que se ajusten a las zonas de la imagen ocupadas por el balón y por el área.

En primer lugar, para crear la máscara del área, se inicializa una máscara de ceros (negro en bits) del tamaño del frame. Para saber cuál de las dos áreas se está detectando, se observa el ángulo de la recta encontrada con la horizontal. Si este ángulo es menor de 90 grados, quiere decir que estamos observando la portería de la derecha y, por lo tanto, el área se encuentra por encima de la línea. En caso contrario, estamos detectando el otro área y estará por debajo de la recta. Después, se procede a cambiar a 255 (blanco en bits) todos los pixels del área y 0 los del resto de la imagen. En código 2 implementa estas funcionalidades.

Código 2: Función que localiza el área y crea una máscara binaria sobre ella.

```

1 # Función que localiza el área y crea una máscara binaria sobre ella
2 def diferenciacion_de_zonas(imagen_original, imagen_tratada):
3
4     imagen_bordes = cv2.Canny(imagen_tratada, 45, 50)
5
6     # Aplica la Transformada de Hough Probabilística sobre la imagen binarizada
7     lines = cv2.HoughLinesP(imagen_bordes, rho=1, theta=np.pi/180, threshold=20,
8                             minLineLength=50, maxLineGap=15)
9
10    # Se definen las máscaras
11    mask_area = np.zeros_like(imagen_original[:, :, 0])
12    mask_fuera_area = np.zeros_like(imagen_original[:, :, 0])
13
14    # Dibuja las líneas en la imagen original
15    if lines is not None:
16        line = lines[0] #Tomamos la primera línea encontrada

```

```

17     x1, y1, x2, y2 = line[0]
18     if (x2-x1) != 0:
19
20         m = (y2 - y1) / (x2 - x1) # Pendiente de la recta
21         b = y1 - (x1 * m) # Ordenada en el origen
22         angulo = math.degrees(math.atan(m)) # ngulo de la linea con la horizontal en
23             radianes
24
25         # Se calculan los puntos de intersección de la linea encontrada con la
26             imagen con la ecuación de la recta
27         h, w = imagen_original.shape[:2]
28         intersection_point1 = (0, int(b))
29         intersection_point2 = (w - 1, int(((w - 1 - x1) * m) + y1))
30         # Se dibuja la linea que abarca toda la imagen
31         imagen_con_linea = imagen_original.copy()
32         imagen_con_linea = cv2.line(imagen_con_linea, intersection_point1,
33             intersection_point2, (255, 0, 0), 2)
34
35
36         # División de la imagen por la linea
37         # Crea una máscara utilizando la ecuación de la linea
38         # Si el ngulo es mayor que 90 grados el rea est a la derecha de la linea
39             , si no a la izquierda.
40         if -angulo<90: # Se pone el ngulo negativo por los ejes que asigna opencv a
41             la imagen
42         for x in range(w):
43             y = int(m * x + b)
44             if y<0: break
45             else:
46                 mask_area[:y, x] = 255
47             else:
48                 for x in range(w):
49                     y = int(m * x + b)
50                     if y<0: break
51                     else:
52                         mask_area[y:, x] = 255
53
54         # Crea una máscara para la regi n a la derecha
55         mask_fuera_area = 255 - mask_area
56
57         # Aplica las máscaras a la imagen original
58         area = cv2.bitwise_and(imagen_original, imagen_original, mask=mask_area)
59         fuera_area = cv2.bitwise_and(imagen_original, imagen_original, mask=
59             mask_fuera_area)
59
59     return mask_area

```

## 2.4. Detección del balón

La detección del balón en los distintos frames se realiza mediante el uso del modelo de detección de objetos YOLOv5.

Al aplicar el modelo YOLOv5, sobre la imagen, se genera un dataframe con las siguientes informaciones sobre los cuerpos detectados:

- La posición del objeto en forma de los puntos que conforman el rectángulo de localización (xmin, xmax, ymin e ymax)
- El nivel de confianza de la detección entre 0 y 1.
- La clase del objeto detectado y el nombre de la clase.

En este caso, es de interés los objetos de la clase 32 (*sports ball*), ya que es a la que pertenece el balón de fútbol. Para filtrar los falsos positivos se le asigna un umbral a la confianza, de manera que todos los objetos con un valor de confianza menor que el umbral se descartan. Con la información sobre la posición del balón se crea una máscara binaria con todo ceros excepto el interior del cuadro de posición donde los píxeles tiene todos el valor 255.

Además, con la información obtenida es posible generar el dibujo del rectángulo de posición de la pelota. La elección del color de esta figura ayuda rápidamente a la comprensión de si la posición del balón es legal para marcar o no. Cuando sea posible marcar el rectángulo se dibujará de verde mientras que donde no lo sea aparecerá de color rojo.

Aquí se muestra el código de la función que se encarga de realizar lo explicado en este apartado.

Código 3: Función de detección del balón

```

1 # Función de detección del balón, dibujo del rectángulo de posición y creación
2 # de la máscara del balón.
3 def deteccion_de_balon(modelo, imagen_original, color):
4     results = modelo(imagen_original)
5     P = results.pandas().xyxy[0]
6     #print(P[P["class"]==32])
7
8     # Crea una máscara
9     mask_balon = np.zeros_like(imagen_original[:, :, 0])
10
11    # Se recorren todos los objetos detectados y solo actuamos sobre los sports ball
12    # con una confianza mínima
13    for i in P.index:
14        if P["class"][i] == 32 and P["confidence"][i]>=0.25:
15            x_min = int(P["xmin"][i])
16            x_max = int(P["xmax"][i])
17            y_min = int(P["ymin"][i])
18            y_max = int(P["ymax"][i])
19
20            imagen_con_rectangulo = imagen_original.copy()
21            cv2.rectangle(imagen_con_rectangulo, (x_min, y_min), (x_max, y_max), color, 2)
22
23            # Se pasan a 1 los píxeles del balón
24            mask_balon[y_min:y_max, x_min:x_max] = 255
25            # Aplica las máscaras a la imagen original
26            balon = cv2.bitwise_and(imagen_original, imagen_original, mask=mask_balon)
27
28            return imagen_con_rectangulo, mask_balon
29
30    return imagen_original, mask_balon

```

## 2.5. Detección de intersecciones entre el balón y las áreas

El objetivo final del proyecto es conocer de manera automática si el balón está dentro o fuera del área durante la jugada. Para ello hay que hacer una comparación entre la zona que ocupa el área y la que ocupa el balón.

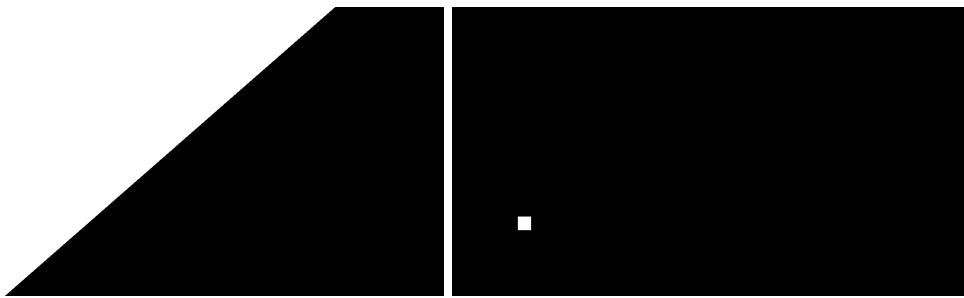


Figura 2: Máscaras del área (derecha) y del balón en un frame concreto (izquierdo).

En los pasos anteriores ya se calcularon las máscaras binarias del área y de la pelota. Para la comparación se realiza la operación binaria AND píxel a píxel entre las máscaras de forma que:

- Si los dos píxeles son 0, el resultado es 0.
- Si un píxel es 0 y el otro 1, el resultado es 0.
- Si ambos píxeles son 1, el resultado es 1.

Como las zonas del área y del balón son aquellas donde los píxeles son 1, habrá intersección y, por tanto, se considerará que el balón está dentro del área, cuando la máscara resultante tenga al menos un píxel igual a 1, es decir, mayor que 0. Si, por el contrario, todos los píxeles del resultado son ceros, se puede afirmar que la pelota está fuera del área y un gol desde esta zona no será legal. En la figura 2, se puede observar un ejemplo de máscara de área y balón entre las cuales hay que hacer la comparación descrita.

La siguiente función realiza calcula la posición relativa del esférico con respecto al área.

Código 4: Función que comprueba la situación del balón en la cancha.

```

1 # Función que comprueba la situación del balón en la cancha.
2 def balon_dentro_area(mask_area, mask_balon):
3
4     intersection = cv2.bitwise_and(mask_area, mask_balon//255)
5
6     # Si la intersección tiene al menos un bit a 1, hay intersección
7     if np.any(intersection > 0):
8         return True
9     else:
10        return False

```

### 3. Resultados

Las imágenes utilizadas en el estudio han sido tomadas desde la terraza junto a la cancha del edificio 850 del campus de Beauchef de la Universidad de Chile. Un set de imágenes fue tomado en un día soleado con actores y, el otro, en día nublado durante un partido real. De esta forma, se puede testear la herramienta en distintas condiciones.

Las pruebas de este informe se han realizado con los siguientes valores:

- Umbral de binarización: 20
- Umbrales del proceso de histéresis de Canny: 45 y 50
- Umbral de la Transformada de Hough Probabilística: 20
- Tamaño mínimo de la línea: 50
- Tamaño máximo de los huecos: 15

- Umbral de la confianza en la detección: 0,25

Los resultados de la detección automática de la posición del balón se muestran en las figuras 3, 4 y 5.



Figura 3: Frames del resultado de la detección automática durante un partido real en un día nublado.



Figura 4: Frames del resultado de la detección automática durante un partido ficticio en un día soleado.

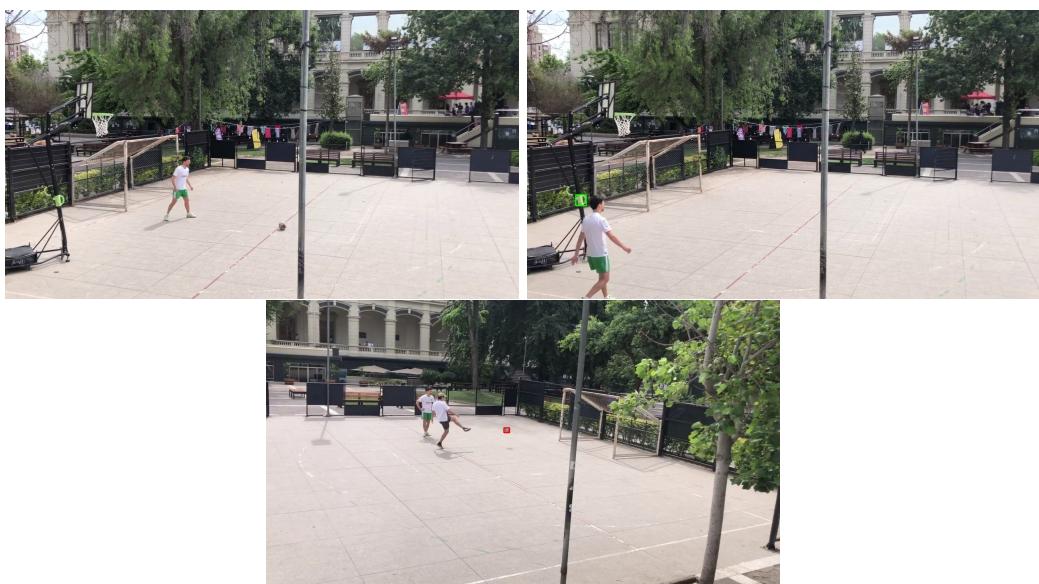


Figura 5: Distintos ejemplos de fallos en la detección.

## 4. Análisis de los resultados

En la figura 3, se puede ver el desempeño de la herramienta analizando una grabación de un partido real. En la primera imagen, se ve claramente que el balón está fuera del área por lo que un gol desde esa posición no cuenta y el rectángulo se muestra de color rojo. Sin embargo, en el otro frame, como la pelota ya está en la posición permitida para marcar, el cuadro se muestra verde.

La figura 4 muestra ejemplos de detección correcta en un día soleado en la zona derecha de la cancha. El hecho de que la figura este más o menos iluminada no interfiere en el correcto funcionamiento del programa. Esto demuestra la robustez del modelo frente a los cambios meteorológicos.

Las imágenes de la figura 5 son ejemplo de fallos de la herramienta de detección. El primer y el segundo error están relacionados con el cálculo de la posición del balón. En la primera imagen se puede observar como el modelo YOLOv5 no es capaz de reconocer la pelota como un objeto. Por otro lado, en el segundo ejemplo, el modelo admite como balón una figura redonda en la canasta. En el caso de la no detección, nuestro poder de acción sería disminuir el umbral de confianza para ser menos estrictos con los balones detectados. Se podría intentar también, descargar un modelo con una versión más actual donde se halla refinado su poder de acción. Para las situaciones de detecciones erróneas, a parte de buscar una versión del modelo más potente, también se puede aplicar un umbral más alto a la confianza de la detección. De esta manera se descartarán como balones deportivos los objetos en los que no se esté muy seguro de haber acertado. Esta claro que la variar el umbral mejora la respuesta frente a uno de estos errores mientras que el otro empeora. Como la herramienta se revisará por un humano, se considera más importante que se detecte lo máximo posible la pelota, incluso cuando haya muchas piernas delante, aunque se dejen pasar algún falso positivo. Como estos objetos son ridículos a simple vista no genera confusiones en el criterio humano. Por lo tanto, es preferible escoger un valor pequeño del umbral.

El tercer caso error es a la hora de detectar el área de la parte derecha del terreno de juego. En concreto, el problema falla a la hora de encontrar la línea del área. Esto se debe a las malas condiciones de la cancha, ya que las líneas están muy desgastadas y han perdido el color rojo característico. Por ello, al resaltar la tonalidad de la línea, como se hace en el lado derecho, se potencian otras zonas y no el límite del área. Una comparación del cálculo de la línea entre las mitades del campo se muestran en la figura 6. Se observa claramente como la recta se destaca frente al resto de formas en el lado derecho, pero en el lado izquierdo, solo se pueden detectar algunos puntos aislados de ella. Esto se resolvería fácilmente si se repintase la pista.

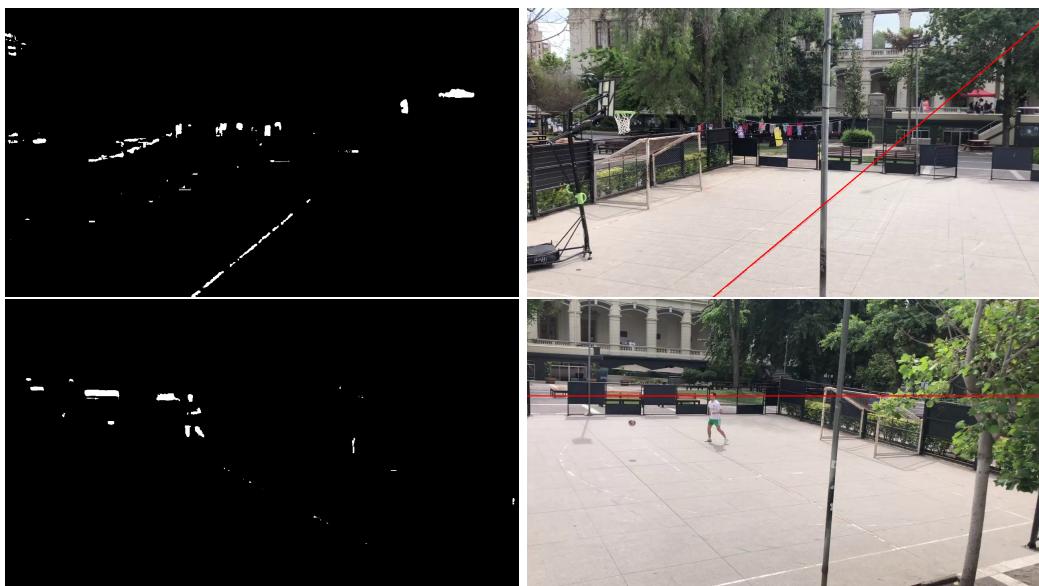


Figura 6: En la fila de arriba, se muestra la imagen binarizada (derecha) y con línea (izquierda) de un video enfocando la portería de la izquierda en un día soleado. En la fila de abajo, se puede ver lo mismo pero de la portería de la derecha.

## 5. Conclusiones

Se puede afirmar que la intrusión de la tecnología en los deportes es muy significativa. Quedan pocos deportes que no cuenten todavía con algún tipo de sistemas que haga más fácil actuar con justicia. Teniendo en cuenta la facilidad de acceso a métodos de grabación el área de tratamiento de imágenes puede ser y es muy útil en este campo.

En el aspecto técnico del proyecto, debido a las características rectilíneas de las figuras a detectar, el uso de los algoritmos de Canny y de la Transformada de Hough ha sido acertado. Al igual que el uso del modelo entrenado YOLOv5 para localizar el balón.

Este proyecto puede llegar a demostrar la factibilidad de implementar una tecnología similar en el ámbito del deporte amateur universitario. Aunque se han obtenido resultados satisfactorios, el margen de mejora es enorme. Se podrían trabajar en nuevas funcionalidades que podrían mejorar el rendimiento de la herramienta, como por ejemplo: detección de un frame óptimo (sin objetos detectados o muy pocos) para localizar el área, ajustar de manera más inteligente el umbral de la variable de confianza de la detección de objetos para refinar los resultados, correcciones de la perspectiva, detección de los límites de la cancha (no solo las áreas) o análisis de las imágenes en tiempo real. El tiempo muy limitado para realizar este proyecto de cero no ha permitido profundizar más en el tema. También, finalmente, se podría pensar en una interfaz amigable para el usuario para que se presentase al CDI con el objetivo de llegar a usar la herramientas en partidos oficiales.

Se ha tratado de probar el detector en diferentes condiciones, pero la variedad de dichas pruebas ha sido muy escasa. Se deberían hacer pruebas en distintas condiciones, como a distintas horas (por la mañana, por la tarde y de noche) o con diferentes posiciones de la cámara. Esto ayudaría a definir de manera más exhaustiva los límites de la herramienta.

Aunque el programa tiene mucho espacio para mejorar, también existen limitaciones externas que empeoran el desempeño de la herramienta. Las condiciones de la cancha no son las mejores en lo que a distinción de líneas se refiere. En algunas zonas cuesta reconocerlas a simple vista. Si se llegaran a repintar las líneas con el color rojo original la detección de zonas mejoraría significativamente.

## Bibliografía

- [1] EFE. *Aprobado el uso del 'Ojo de Halcón' en la línea de gol.* URL: <https://www.20minutos.es/deportes/noticia/tecnologia-futbol-ojo-halcon-linea-gol-1531623/0/>. (accessed: 22.11.2023).
- [2] *Algoritmo de Canny.* URL: [https://es.wikipedia.org/wiki/Algoritmo\\_de\\_Canny](https://es.wikipedia.org/wiki/Algoritmo_de_Canny). (accessed: 27.11.2023).
- [3] *Transformada de Hough.* URL: [https://es.wikipedia.org/wiki/Transformada\\_de\\_Hough](https://es.wikipedia.org/wiki/Transformada_de_Hough). (accessed: 27.11.2023).
- [4] Iain Macdonald. *Probabilistic Hough Transform.* URL: [https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/AV1011/macdonald.pdf](https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/macdonald.pdf). (accessed: 27.11.2023).

Se ha usado ChatGPT para obtener ideas sobre las cuales profundizar la investigación y como apoyo para la programación. También se han revisado de nuevo los materiales docentes de las clases.