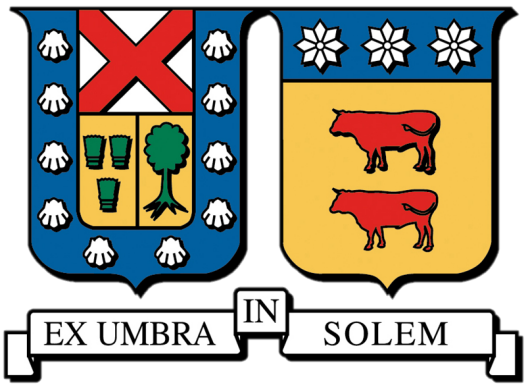
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA



MONITOREO DE RUGBISTAS MEDIANTE VISIÓN POR COMPUTADOR PARA IDENTIFICACIÓN DE CHOQUES

CLAUDIO BLAS ARNALDO SANTANA DOMINGUEZ

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

Guía: Marcos David Zuñiga Barraza

Co-referente: Nicolás Alonso Jara Carvallo

DICIEMBRE-2024

Resumen

Si bien recién en el 2022 las inteligencias artificiales (IA) por redes neuronales convolucionales (CNN) experimentaron un crecimiento acelerado en su uso en varias áreas llegando a ser incluso una herramienta del día a día, la visión por computador (CV), es un área de las IA que lleva mas de 70 años siendo desarrollada, viendo su avance a día de hoy por ejemplo, en las cámaras de seguridad, en donde actualmente varias de ellas son capaces de realizar identificación de personas y objetos que tienen en el rango de su visión. El uso de monitoreo e identificación a llegado a ser de interés incluso en el deporte, por ejemplo, hoy la UEFA es capaz de realizar modelados 3D de los jugadores y el balón en fotogramas de interés para detectar de mejor manera un fuera de juego, falta, o gol. Así también se pueden nombrar varias otras herramientas especializadas en el análisis de acciones en el deporte, como la cámara Veo, la cual es usada por un club de rugby chileno: All Brads. Si bien la IA implementada por Veo logra identificar momentos de interés en el partido (los Try, Scrum, Conversion, Lineout y comienzo/final de las 2 mitades del partido), no logra identificar otros momentos que son de interés para los entrenadores. Esta memoria se realiza con el fin de cubrir alguno de estos parámetros que Veo no logra cubrir, como son los choques entre jugadores que suceden en el partido, a través de la identificación de las caídas, tackles y colisiones. Esto se desarrolla utilizando diversos frameworks con herramientas de identificación y seguimiento contenedoras de modelos ya entrenados, utilizar estas identificaciones para detectar posibles choques de rugbistas en los frames de la grabación, confirmar si corresponde a alguna de las 3 identificaciones de interés mediante el uso de una arquitectura IA diseñada en PyTorch y medir su gravedad a través de clips cortos generados a partir de los frames identificados por la arquitectura, junto con la indicación de las partes del cuerpo impactadas.

Abstract

While it was not until 2022 that artificial intelligence (AI) based on convolutional neural networks (CNN) experienced accelerated growth in its use across various fields, even becoming an everyday tool, computer vision (CV) is an area of AI that has been developed for over 70 years. Today, its advancements are evident in security cameras, many of which can now identify people and objects within their field of vision.

The use of monitoring and identification has even become relevant in sports. For example, UEFA can now create 3D models of players and the ball in key frames to better detect offside calls, fouls, or goals.

Similarly, there are other specialized tools for analyzing actions in sports, such as the Veo camera, used by a Chilean rugby club: All Brads. While the AI implemented by Veo can identify key moments in a match (Try, Scrum, Conversion, Lineout, and the start/end of the two halves), it fails to identify other moments of interest to coaches.

This thesis aims to address some of the parameters Veo does not cover, such as collisions between players during a match, by identifying falls, tackles, and collisions. This is achieved using various frameworks with identification and tracking tools containing pre-trained models. These identifications are used to detect potential collisions between rugby players in the video frames, confirm whether they correspond to one of the three target identifications using an AI architecture designed in PyTorch, and measure their severity through short clips generated from the frames identified by the architecture, along with indicating the parts of the body impacted.

Agradecimientos

A mi familia por criarme y amarme incondicionalmente, a mis amigos de infancia por la lealtad y por siempre ser oído, a mis ángeles del cielo, mis abuelas y abuelos, tías y tíos, por ser mi sombra y nunca dejarme aun en mis momentos mas bajos, por siempre acompañarme aun después de su partida. A mi amigos y compañeros de la universidad por aguantarme y entenderme, y llenar de lindos recuerdos este largo viaje. Y a Jesucristo, por demostrarme que vale la pena pelear y morir buscando un mundo mejor.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

1.1 Contexto

Desde 2022, el equipo chileno de rugby All Brads ha estado utilizando una cámara Veo para grabar sus partidos, lo que les ha permitido mejorar sus entrenamientos y corregir diversas tácticas. Teniendo como norte la constante mejora del equipo en el menor tiempo posible y considerando el uso tanto de Visión por Computador como de sensores en ambientes profesionales (como en el reciente mundial de rugby), se pretende diseñar un sistema tecnológico que sea de ayuda para el club All Brads.

Además de la cámara Veo, el equipo All Brads posee algunos sensores: un GPS y un acelerómetro/giroscopio. También poseen sostenes deportivos para guardar los sensores en los jugadores y una placa ESP 32.

1.2 Problema a Resolver

Los análisis de video en donde no se identifiquen de forma inmediata rendimientos de interés de los entrenadores (como, por ejemplo, aceleración en 5/10 metros y de los choques ocurridos), inducen a una mejora más lenta del equipo. Es por ello, que resulta de gran valor levantar y conocer estos datos, no solo para entregarlos a los entrenadores, sino que además brindarle cierto grado de soporte al momento de tomar decisiones técnicas del equipo, utilizando análisis de datos e inteligencia artificial. Este es el valor que el sistema aportará para los rugbistas chilenos, utilizando los sensores mencionados previamente y sacándole el máximo provecho a las grabaciones de la cámara Veo.

1.3 Acercamiento a la Solución

En sí, el trabajo previo (y literatura en general) es bastante extensa para IoT y Visión por Computador aplicado a deportes, para mencionar un enfocado en rugby realizado en el año 2016, en donde se diseña un sistema que despliega la línea de seguimiento de los jugadores, ayudados de un GPS y el análisis de video [1]. Esto sirve como base para asegurar que es posible la implementación, pero no obstante, cada parte de la implementación del sistema fue dividido en 4 grandes secciones además de la de visión por computador que se abordará en esta memoria:

- Fabricación de contenedor de los sensores con sus placas
- Configuración de los sensores para la toma y envío de datos
- Procesamiento de los datos de los sensores para entregar información relevante
- Visión por Computador aplicada a los videos

El gran enfoque en lo que respecta la parte de visión por computador del sistema (y lo que es de interés en este escrito), es la identificación de los choques sucedidos en los partidos y medir su gravedad, puesto que este análisis ayuda para saber los tiempos de recuperación que tiene cada jugador y así tener clara la disponibilidad de estos en diversas jugadas o situaciones que ocurran durante los partidos, además de poder tener un historial de choques que haya sufrido cualquier jugador, el cual sirve para prevenir lesiones y generar entrenamientos adaptables a la situación de cada uno.

Sin embargo, tanto para el uso de sensores como de las grabaciones en el seguimiento de los jugadores, no se puede asegurar mediciones exactas de la velocidad y aceleración: en los sensores por diversos factores ambientales externos y propios del funcionamiento del sistema (como los tiempos en la toma de la ubicación de los jugadores y el funcionamiento propio de las antenas) y en las grabaciones por el solapamiento constante que hay entre jugadores, se dificulta la re identificación de los mismos cuando vuelven a ser visibles en la grabación luego de un periodo de tiempo considerable.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar una rutina o algoritmo que utilice las grabaciones de rugbistas en situaciones de juego para identificar choques, caídas y tackles que ocurran en los videos con el fin de medir su gravedad, e informar del tiempo en la grabación en que ocurre.

1.4.2 Objetivos Específicos

- Identificar jugadores de rugby
  - Encontrar un modelo pre entrenado que realice buenas identificaciones de los jugadores (realizar comparaciones en diversos momentos de algún partido)
- Seguimiento de los jugadores
  - Identificar los momentos en que se deje de seguir instancias (indicador de choque o caída) y extraer el fotograma en que sucede
  - Realizar estimaciones de velocidad y aceleración de las instancias, identificar variaciones importantes (indicador de choque) y extraer el fotograma
- Identificar colisiones, caídas y tackles
  - Crear un dataset con imágenes de alta calidad de los 3 eventos de interés
  - Crear arquitecturas que sean de utilidad para la clasificación de estos eventos
  - Probar la eficiencia de estas arquitecturas en fotogramas extraídos de las grabaciones de Veo
- Identificar gravedad de los choques
  - Generar clips cortos a partir de los fotogramas de los eventos encontrados
  - Encontrar a partir de estos clips, las articulaciones involucradas

1.5 Hipótesis

Es posible identificar los choques entre jugadores en un partido de rugby analizando, la velocidad, poses y perdida de detección de los jugadores, utilizando modelos IA de arquitectura simple para clasificar si es una colisión, tackle, caída o no, a pesar de tener presente varios momentos de solapamiento entre jugadores.

1.6 Capítulos Siguientes

La descripción del trabajo realizado se seguirá en los siguientes 4 capítulos:

- Primero se describirá el estado del arte, que es todos los trabajos realizados previamente que tienen relación directa o indirecta con la problemática y/o hipótesis a resolver, para luego de

realizar una discusión interpretando los avances desarrollados y lo nuevo que podría aportar la solución de este escrito, se define el marco teórico del problema, describiendo las definiciones, herramientas y técnicas que fueron más útiles para el desarrollo de la solución.

- En el capítulo siguiente, se hará una descripción del sistema completo, indicando los requisitos de este, el perfil de los usuarios que usarán la solución, cual es la arquitectura del sistema, para luego realizar una descripción detallada de lo realizado por visión por computador aplicado a rugby

- Luego, se expondrán los resultados visuales al aplicar la solución en las grabaciones del análisis, y algunas interpretaciones de estos resultados.

- Finalmente, se describirán las conclusiones encontradas luego de ver los resultados, respondiendo si fue posible cumplir con la hipótesis y luego describir los trabajos futuros, es decir, que cosas podrían corregirse, incluirse, o potenciarse en la solución para tener mejores resultados.

Capítulo 2

Estado del Arte

2.1 Marco Teórico

2.1.1 Definiciones clave en el contexto del rugby

- Tackle: Una acción específica donde un jugador busca detener el avance de un oponente, tratando de derribarlo al suelo, tradicionalmente a través de una lanza (choque entre 2 hombros tratando de realizar un abrazo a medias, es decir con los brazos abiertos). Puede ser frontal, al costado, o atrás del rival a partir del hombro hacia abajo, incluidas las piernas.

- Colisión: Contacto entre jugadores que no necesariamente implica un intento de detener al oponente, pero que puede inducir cierta gravedad por el nivel y brusquedad de esta. Este tipo de interacción es frecuente y puede involucrar cambios bruscos en la velocidad y dirección de los jugadores.

- Caída: Una acción específica donde un jugador busca detener el avance de un oponente derribándolo al suelo. Es uno de los eventos más característicos y regulados en el rugby.

2.1.2 Algoritmos de detección de objetos y personas

- R-CNN (Regions with Convolutional Neural Networks): Primer modelo que combinó regiones propuestas con redes neuronales profundas para detectar objetos.

- Fast R-CNN y Faster R-CNN [21]: Mejoras en velocidad y precisión con respecto a R-CNN, introduciendo redes como selectores de regiones.

- YOLO (You Only Look Once): Algoritmo de detección en tiempo real, capaz de procesar imágenes completas en una sola pasada. Es uno de los más usados en el estado del arte,

principalmente porque puede tener mejor rendimiento en tiempo de ejecución, en compensación con una precisión más baja que otras soluciones

- RetinaNet[25]: Es un modelo de detección de objetos de una sola etapa que destaca por su capacidad para manejar de manera efectiva el problema del desequilibrio entre clases en datasets de detección. Utiliza Focal Loss para dar menor peso a ejemplos fáciles priorizando los más complejos y una red piramidal de características para detectar objetos de diferentes escalas de manera eficiente, mejorando la detección de objetos pequeños o menos representados.

2.1.3 Elementos complementarios a la detección

- Articulados y keypoints: Los articulados son representaciones del cuerpo humano mediante puntos clave (keypoints), que corresponden a las partes que lo componen. Estos puntos se usan ampliamente en visión por computador para estimación de poses y detección de eventos en deportes [10]. Por ejemplo, el uso de OpenPose permite generar un esqueleto virtual que facilita identificar interacciones como tackles en rugby, al analizar ángulos corporales y sus cambios bruscos durante el contacto [12].

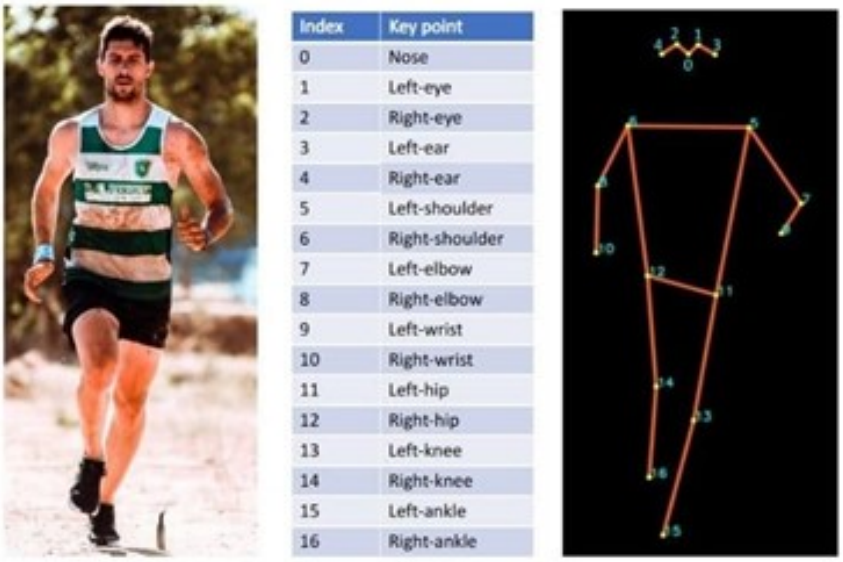
- Segmentación por Máscaras: Suma de técnicas donde se incluye uso de filtros, contraste de colores/grises (como el usado en [12]), y/o saturación de imágenes para resaltar elementos de interés presentes en la imagen, con o sin la ayuda de algoritmos de detección de objetos.

2.1.4 Modelos de detección de los elementos complementarios

- Mask R-CNN [26]: es una extensión del modelo Faster R-CNN que añade una rama para la segmentación de instancias. Esto permite generar máscaras de píxeles para cada objeto detectado, además de sus respectivas cajas delimitadoras y clases. Mask R-CNN se adapta de manera efectiva a tareas que requieren una comprensión detallada de los objetos en una escena, como la segmentación y el seguimiento en deportes. El modelo opera primero con una propuesta de regiones candidatas donde podrían existir objetos, para luego realizar la predicción de máscaras y clases, haciendo un refinamiento de las regiones propuestas para predecir la clase del objeto, ajustar la caja delimitadora y generar la máscara binaria de segmentación.

- Keypoint R-CNN: extiende Mask R-CNN para la detección de puntos clave en objetos, como articulaciones humanas. A diferencia de otros métodos, este modelo predice cada punto clave como un mapa de calor que indica la probabilidad de la ubicación del punto clave en la imagen. Esta capacidad lo hace particularmente útil para el análisis de posturas y eventos en

deportes, donde los puntos clave pueden ser utilizados para inferir dinámicas como caídas, colisiones o tackles. Utiliza un código de numeración similar al del dataset COCO (Common Objects in Context) (ver figura 2.1).



Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle

Figura 2.1: Código de Keypoints de Coco

2.1.5 Seguimiento de Múltiples Objetos

El seguimiento de múltiples objetos (MOT) es una extensión natural de la detección, donde el objetivo es asignar identificadores únicos y persistentes a cada objeto a lo largo de una secuencia de fotogramas. De la variedad de algoritmos de seguimiento (tracking), hay 3 de interés:

- BotSort [30]: Algoritmo que combina detecciones actuales con un historial de movimientos previos, ajustando las trayectorias de manera más precisa en escenarios complejos.
- StrongSort [31]: Su fortaleza radica en el uso de descriptores visuales más avanzados que otros algoritmos, permitiendo identificar mejor cada objeto según su apariencia, incluso en situaciones donde hay varios similares o con cambios abruptos en sus trayectorias. Además, StrongSort optimiza la forma en que combina esta información visual con los datos de movimiento.
- ByteTrack [32]: Es un método eficiente de seguimiento que utiliza todas las detecciones (incluso las de baja confianza) para mantener un seguimiento continuo y preciso, ideal en entornos con muchas oclusiones.

Ahora bien, estos modelos varias veces no funcionan por sí solos, necesitan una forma de Re-Identificar las detecciones. Aquí es donde aparecen los modelos de Re-Identificación (ReID), diseñados para reconocer y emparejar una misma entidad (persona en nuestro caso) en diferentes imágenes o cuadros de video, considerando cambios en el ángulo de visión, iluminación, postura o entorno. Estos modelos generan un descriptor visual único para cada entidad, que luego se utiliza para compararlas y determinar si corresponden al mismo objeto. Estos modelos ayudan a emparejar detecciones a lo largo del tiempo, por ejemplo, cuando un objeto desaparece por un momento (por una oclusión). De estos modelos, se pueden destacar 3:

- CLIPReID [27]: Modelo con un enfoque que combina información textual y visual para generar descriptores. Permite emparejar objetos en contextos donde las características visuales varían mucho, aprovechando representaciones de múltiples modos para mejorar la precisión del ReID.
- LightMBN [28]: Es un modelo ligero diseñado para dispositivos con recursos limitados. Utiliza una arquitectura basada en redes neuronales de bajo costo computacional para generar descriptores visuales. LightMBN ofrece un buen equilibrio entre precisión y velocidad.
- OSNet (Omni-Scale Network) [29]: Es un modelo ReID avanzado que puede capturar tanto detalles finos como características globales de los objetos gracias a su diseño de red "multi-escala". Esto lo hace especialmente eficaz para escenarios donde los objetos cambian de escala, como personas vistas de cerca o a lo lejos.

2.1.6 Frameworks a utilizar

- Detectron2: Es un framework avanzado de visión por computadora desarrollado por Facebook AI Research (FAIR) en PyTorch. Está diseñado para tareas relacionadas con la detección de objetos, segmentación de instancias, segmentación semántica, segmentación panóptica y detección de keypoints. Contiene los modelos de detección de objetos y los modelos de detección de elementos complementarios mencionados previamente, los cuales fueron pre entrenados con el dataset COCO.
- Boxmot: Es un framework especializado MOT, diseñado para unir detecciones individuales a lo largo del tiempo y generar trayectorias coherentes de los objetos en una secuencia de video. BoxMOT típicamente integra algoritmos de seguimiento, Modelos ReID para mejorar la asociación de detecciones y técnicas de procesamiento para ajustar las trayectorias, como el uso de filtros de Kalman o análisis de intersecciones de bounding boxes (IoU). Contiene todos los modelos mencionados en el apartado de seguimiento de múltiples objetos y más. Los modelos de ReID pueden ser descargados automáticamente por BoxMot al ser usados especialmente los de OSNet,

2.2 Sensores en Deportes

Desde hace ya casi una década, el uso de sensores en el deporte tiene un lugar importante en el comercio, como queda plasmado en la review de Rana y Mittal [3]. Más allá de los deportes de movimiento como carreras y fútbol (con el fin de medir aceleración y velocidad), también se ha utilizado en natación con los mismos fines y en el tenis con el fin de medir la velocidad angular en los brazos. Lo clave en la review es la mención de que estos son elementos no invasivos y también que la medición es realizada mediante unidades de medida inercial (IMU), destacándose como una alternativa accesible y adecuada para un proceso más rápido, fiable y rentable de medición. En el mercado hay marcas más posicionadas para la medición, de las más nombradas es el Catapult Optimeye s5, el cual tiene incrustado un GPS, giroscopio, acelerómetro y magnetómetro. Una implementación usando este sensor [4] fue realizada en jugadoras de basquetbol según 6 niveles de intensidad en las jugadas, en correlación con un Vicon Motion (traje negro para realizar representaciones 3D, generalmente en películas) para confirmar la exactitud del Optimeye s5, hallándose que los datos son válidos, a pesar de encontrarse importantes discrepancias para el tracking. No obstante, la implementación de este sensor en la identificación de eventos (específicamente Scrum) [5], se logra una exactitud mayor al 90%. En síntesis, la literatura acepta el uso de sensores en los deportes, mas recomienda utilizar los datos tomados como un complemento al posterior análisis por Visión por Computador que se realice, debido a los fallos externos y propios del sensor que pueda tener al momento de tomar los datos y la condicionante en el rendimiento propio de los atletas cuando usan estos aparatos [6], puesto a que como se indica en la review, es posible diseñar contenedores no invasivos para los jugadores.

2.3 Visión por Computador en Deportes

Esta área a tenido grandes avances a lo largo del tiempo y desde hace mucho que se viene trabajando su aplicación en deportes. Un gran avance tecnológico fue el uso del ojo de águila en el tenis para identificar si la pelota golpea la línea o no, al ser una grabación fija y un entorno relativamente pequeño y controlado [7]. Sin embargo, tuvieron que pasar muchos años para que la inteligencia artificial entrara de lleno a la visión por computador [7] y a pesar de su inclusión fuerte en los últimos años, varios de los problemas y desafíos pre IA siguen sin tener una solución global, como el solapamiento de jugadores y el balón (característica crítica en el seguimiento), identificación de eventos en el partido, identificación de formaciones, segmentación de máscara de los jugadores, percepción y giros de la cámara, entre otros [8]. Uno que es importante de mencionar, es la poca cantidad de datasets públicos de rugby que hay disponibles. De hecho, es más cuantioso encontrar datasets de su adaptación norteamericana, la NFL de Estados Unidos. El deporte rey (fútbol) es el

que ms trabajos tiene, los cuales han ido creciendo en importancia gracias al VAR.

2.3.1 Seguimiento/tracking

En la figura 2.2 se aprecia el uso de TRACAB [8] para el tracking de jugadores, usándose para detectar una posición adelantada en fútbol a través de la línea trazada por las articulaciones de los jugadores.

Figura 2.2: TRACAB Gen5 para seguimiento de jugadores

Las detecciones por sí solas no son capaces de identificar si una instancia pertenece a un jugador, sino que necesitan de un algoritmo de seguimiento y el más usado en deportes es ByteTrack [32], especialmente para grabaciones fijas, de buena definición y con poco solapamiento como las corridas olímpicas y basketball. Debido a la importancia del seguimiento y de tener una buena base para detecciones de calidad, es que se han desarrollado datasets públicos para facilitar además del seguimiento correcto de los jugadores, la clasificación misma del deporte de la grabación, como SportsMOT [16], diseñado principalmente para fútbol, basquetbol y volleyball, aunque en varias situaciones se hace necesario además tener identificada las regiones de las canchas para sacar mayor información [17], a través de la extracción de características usando diversos encoders como ResNet [34] y ViT-tiny [35], cuya implementación gráfica de este último encoder se aprecia en la figura 2.3.

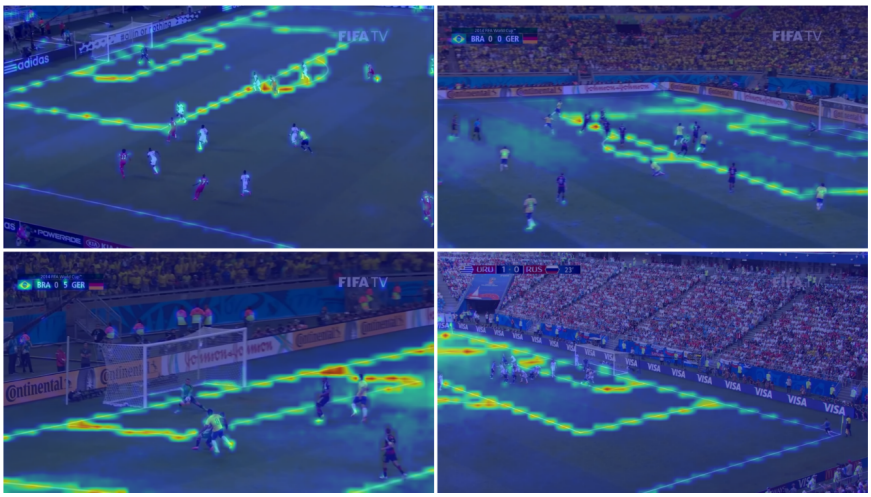


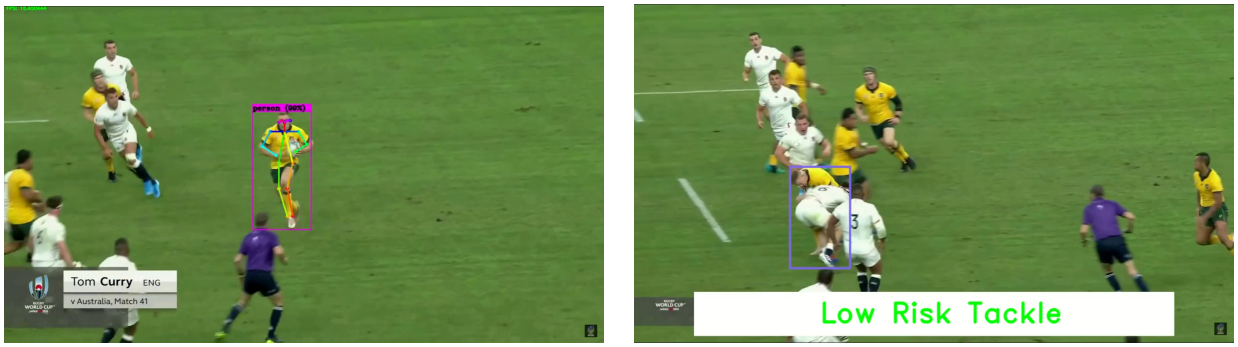
Figura 2.3: Extracción de características usando ViT-tiny [17]

2.3.2 Reconocimiento de Acciones

Como se aprecia en la figura 2.2, el seguimiento viene muchas veces de la mano con la representación del articulado de los jugadores y en un deporte como el rugby donde el cuerpo está en constante choque, saber las áreas afectadas es de especial interés. Por ello, cualquier trabajo centrado en la identificación de lesiones, o situaciones de riesgo va a ser de interés, como sucede en [9] que es un trabajo centrado en ejercicios anaeróbico, como el levantamiento de pesas. A pesar de lo mencionado anteriormente, es posible lograr un análisis de desempeño de los jugadores a través de acciones sin recurrir a articulado y solo utilizando "tags" de momentos [16], siempre y cuando un usuario pueda determinar estos momentos. Son varias las arquitecturas utilizadas para el reconocimiento de acciones en el deporte. Dentro de las más explotadas se encuentra el uso de Long Short Term Memory (LSTM), una red neuronal recurrente (RNN) que ha hecho posible la identificación exitosa de varias acciones en diferentes deportes, combinando su uso con otras arquitecturas [17]. El uso de vectores de características también ha sido usado en pos de reducir la complejidad de las redes que identifican las acciones [18] y también un híbrido entre LSTM y extracción de características [19]. Tal es el alcance del uso de LSTM en este contexto, que ha llegado a incluirse en arquitecturas más globales para el reconocimiento de poses, como HRNet [20], la cual procesa imágenes de alta a baja calidad y de baja a alta calidad, logrando que la estimación de poses sea adaptable para cualquier contexto en que se utilice la arquitectura. Todos los trabajos relacionados a LSTM dan el siguiente fundamento: la identificación de acciones no debe ser bajo arquitecturas complejas en su diseño. En síntesis, la identificación de acciones puede desglosarse en gestos, acciones, interacción y actividad en grupo, siguiendo la definición de HAR (Human Action Recognition) [13], aunque la mayor efectividad de este concepto se da para la identificación de deportes más que momentos de uno en específico, dada la diferencia en las interacciones.

2.4 Visión por Computador en Rugby

Existen algunos trabajos realizados para rugby donde se combina seguimiento e identificación de acciones, como el realizado por la Universidad de Keio en Japón [15], en donde logran resolver los problemas de solapamiento trabajando en tomas únicas de los jugadores para luego realizar una representación de vista aérea, logrando crear mapas de calor según la velocidad de los jugadores para realizar la identificación de Kick-Offs, Scrums, Lineouts, Turnovers y Penalties, además de un contador de patadas. Uno de los trabajos desarrollados específicamente para rugby más destacados es el realizado por la Universidad del Cabo [10], donde usando YOLOv4 alimentado con etiquetas de LabelImg, filtros de Kalman para la generación exitosa de las bounding boxes (a pesar del ruido) y OpenPose, se detectan tackleadas recibidas por el jugador que tiene el ovoide y se mide su gravedad, según el ángulo de las partes del cuerpo involucradas en la colisión. Este es de los pocos trabajos que ofrece un dataset público de videos de tackleadas (de 2 segundos). En la figura 2.4 aparece el renderizado que hace esta solución, donde se muestra el articulado de OpenPose del jugador que acarrea el balón y luego la clasificación del tackle.



(Articulado del jugador que tiene el balón) (Clasificación del tackle)

Figura 2.4: Solución para clasificación de riesgo de un tackle de la Universidad del Cabo [10]

En línea con la identificación de acciones, se tiene también el uso de redes neuronales en problemas complejos de mapeo de condiciones para la detección de determinados eventos. Tomando como base la implementación de esta arquitectura para la detección de factores de riesgo en redes eléctricas [11], es que surge una implementación para la detección de tackles en rugby [12], en donde luego de un pre procesado de las imágenes por escalado de grises, se logra la creación de una segmentación de máscara, para luego generar un articulado de 12 puntos clave del cuerpo que servirán para una red neuronal creada por algoritmos genéticos con el fin de identificar el lado en que ocurrió el tackle (si este fue de costado, frontal, de atrás o por varios jugadores). El articulado generado luego de la creación de la máscara se puede observar en la figura 2.5

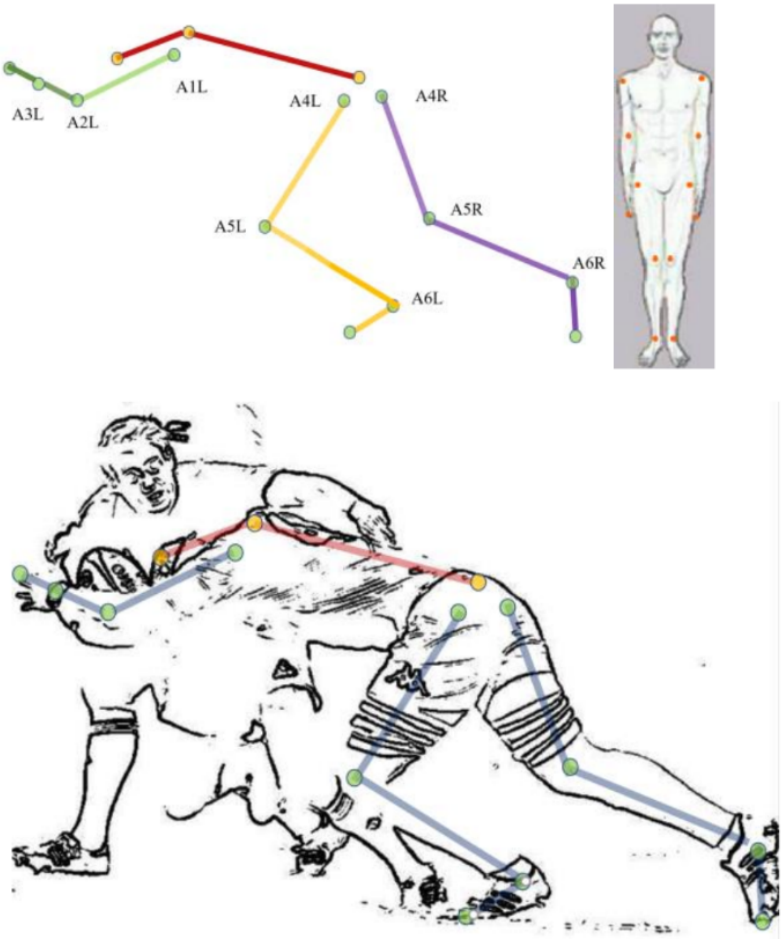


Figura 2.5: Articulado de tackle mediante máscara [12]

El articulado si bien es una gran ayuda para la estimación de poses, este no es necesariamente indispensable, puesto que puede realizarse a través de análisis netamente por videos cortos [14]. En el trabajo referenciado, la detección de tackles se divide en 4 partes: selección de un fotograma por ResNet modificado para 3 dimensiones de convolución, detección del tackle con modelo de detección entrenado con bounding box, estimación de poses con HRNet [20] y finalmente clasificar el riesgo del tackle ocurrido. Una vista general de estos pasos descritos se aprecia en la figura 2.6.

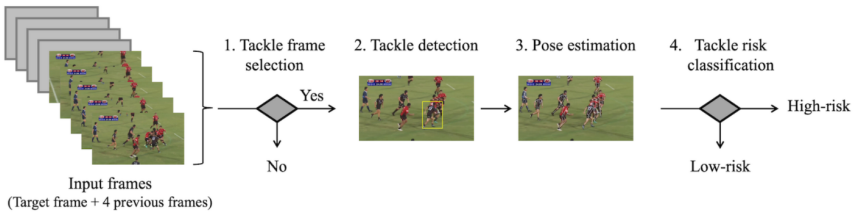


Figura 2.6: Secuencia general de funcionamiento del análisis de 4 pasos [14]

Sin embargo, la mayoría de estos trabajos de seguimiento e identificación fueron realizados en

entornos donde las imágenes eran de alta calidad y distinción de jugadores (bajo solapamiento), incluso algunos solo consistían en imágenes. Para entornos amateur, se encontraron solo 2 trabajos. En el primero [1] se hace uso de un GPS en los jugadores, con el objetivo de identificar y amortiguar los fallos de medición que pueda tener el sensor (propios del ambiente, el medio por el que se envía y del mismo GPS). De esta forma se logra una representación tridimensional del recorrido de los jugadores, tras también trazar exitosamente las líneas marcadas en la cancha, como se aprecia en la figura 2.7.

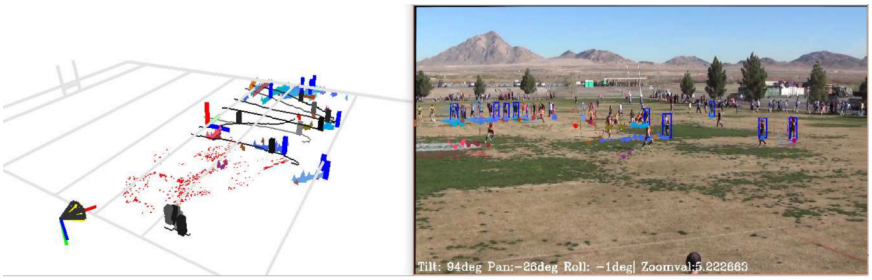


Figura 2.7: Salida del seguimiento con video y GPS [1]

En el segundo trabajo [2], utilizando una arquitectura de detección de eventos generales de deportes llamado NETVLAD++, la cual considera contextos temporales del pasado para realizar mejores estimaciones del futuro. Esta arquitectura es usada principalmente para la detección de todos los eventos que suceden en el rugby, específicamente Maul, Ruck, Scrum, Kick, Lineout, Try, e incluso Tackles.

2.5 Discusión

Una vez revisado el estado del arte, se puede decir que:

- Es posible según lo trabajado en [1] la integración entre IoT y visión por computador para análisis de rugby, en [2] también se demuestra que es posible trabajar en entornos no profesionales.
- En [2] también, la arquitectura general para la identificación de eventos en el juego no es específica de rugby, por lo que no es descabellado utilizar arquitecturas pre entrenadas.
- A su vez, el articulado es un elemento muy repetido para la identificación de tackles y su gravedad, al igual que también ha sido usada la máscara de instancias para ello.
- En la gran mayoría de trabajos, la identificación se hace mediante el entrenamiento con los mismos datos a analizarse posteriormente y con imágenes cercanas y de muy alta calidad, sin

considerar el solapamiento, cosa que en la realidad sí ocurre mucho cuando hay colisiones, caídas y tackleadas. Esto último es relevante, ya que hace ms única la solución desarrollada en este escrito.

- En [15] si bien se usa de forma indirecta la velocidad para encontrar momentos del partido, no se hace un análisis directo a variaciones de velocidad para la identificación de las mismas, siendo que una colisión o caída implica directamente un cambio brusco en la velocidad.

El aporte que viene a entregar este escrito, es la consideración de los solapamientos durante todo el desarrollo de la solución (no eliminarlas), puesto que no se encontró algún trabajo en el que fuese considerado para el análisis por visión por computador (en [1], se soslaya al hacer el trazado del recorrido usando los sensores, y el video como soporte para su representación tridimensional). El uso de las tecnologías mencionadas también vienen a ser una novedad, ya que el estándar en la literatura es usar YOLO y ByteTrack únicamente, por lo que considerar usar Detectron2 y BotSort viene a brindar otro enfoque a la problemática.

Capítulo 3

Algoritmo Propuesto

El algoritmo propuesto para mejorar los análisis de los All Brads, viene a seguir de manera casi secuencial los objetivos específicos encontrados al momento de desarrollar la solución (en el apartado de Visión por Computador). A continuación se desglosan las consideraciones que se tuvieron en cuenta para el desarrollo del algoritmo y el detalle completo de su implementación junto con las partes que lo componen

3.1 Requisitos del Sistema

3.1.1 Requisitos funcionales

1. El sistema entrega resultados con un nivel de precisión aceptable
2. El sistema identifica los momentos de riesgo que puede inducir a lesiones en los jugadores (tackles, caídas y colisiones)
3. El sistema realiza las líneas que representan el recorrido de los jugadores
4. El sistema encuentra los metros recorridos en determinado intervalo de tiempo
5. El sistema una vez finalizado el procesamiento, entrega por pantalla los momentos en que ocurrieron jugadas riesgosas
6. El Sistema identifica las partes del cuerpo involucradas en las jugadas de riesgo

3.1.2 Requisitos No funcionales

1. El tiempo de ejecución del sistema es aceptable
2. El sistema entrega resultados claros para los usuarios

3.2 Requisitos de Ambiente

3.2.1 Hardware de Desarrollo

- Intel Xeon CPU @2.30 GHz
- 13 GB RAM
- GPU Tesla T4 16GB VRAM

3.2.2 Software de Desarrollo

- Python 3
  - Selenium
  - Numpy
  - Matplotlib
  - PyTorch
  - OpenCV2
  - sklearn.metrics
  - Detectron2
  - BoxMot

3.3 Perfiles de Usuario

En la tabla 3.1 se describen los 2 perfiles de usuario del sistema

3.4 Diseño de arquitectura del sistema

Aquí se define a grandes rasgos, como el sistema va a operar por los perfiles de usuario y los módulos que contendrá, destacando los desarrollados para esta memoria

3.4.1 Diagrama de contexto

En la figura 3.1 se ilustra cómo es la interacción con el sistema de parte de ambos tipos de usuario y cómo recibe los datos con los que trabajará.

Perfil	Socioeconómico y Cultural	Ocupacional	Etario	Características físicas, fisiológicas, psicológicas	Otros
Entrenador de Rugby	Tiende a incluir personas con un nivel educativo medio a alto, con un fuerte interés en el deporte	Entrenadores profesionales de rugby o personas dedicadas al entrenamiento deportivo. Pueden trabajar en clubes deportivos, escuelas o instituciones deportivas	Generalmente adultos jóvenes a adultos mayores, en el rango de edad de 25 a 60 años	No hay requisitos específicos en cuanto a características físicas o fisiológicas. Se espera que tengan habilidades de liderazgo, capacidad para analizar datos y tomar decisiones estratégicas	Deben tener un buen conocimiento del juego de rugby y una comprensión sólida de cómo mejorar el rendimiento de los jugadores. También pueden tener experiencia en el uso de tecnología para el análisis deportivo
Jugador de Rugby	Varía ampliamente, pero tiende a incluir personas de diversos trasfondos socioeconómicos y culturales con un interés común en el rugby	Jugadores de rugby amateurs o profesionales, que pueden estar afiliados a equipos escolares, clubes deportivos o ligas profesionales	Principalmente jóvenes y adultos, en el rango de edad de adolescentes hasta adultos mayores, generalmente entre 15 y 35 años	Se espera que tengan un buen estado físico y salud general, así como habilidades deportivas como resistencia, fuerza y agilidad. Mentalmente, deben tener capacidad para trabajar en equipo, resistencia a la presión y capacidad para aprender y adaptarse	Los jugadores de rugby pueden tener una variedad de habilidades y roles en el equipo, desde jugadores de línea hasta medio scrums o fullbacks. También pueden tener diferentes niveles de experiencia, desde principiantes hasta jugadores experimentados

Tabla 3.1: Perfiles de Usuario

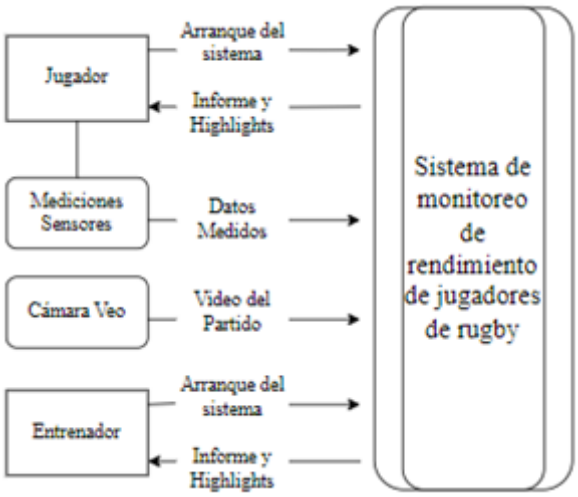


Figura 3.1: Diagrama de contexto del sistema

3.4.2 Diagrama de Arquitectura

En la figura 3.2 se muestra la arquitectura general del sistema. Cada módulo se representa como un círculo azul y los datos de apoyo o generados por el sistema como círculos naranjas.

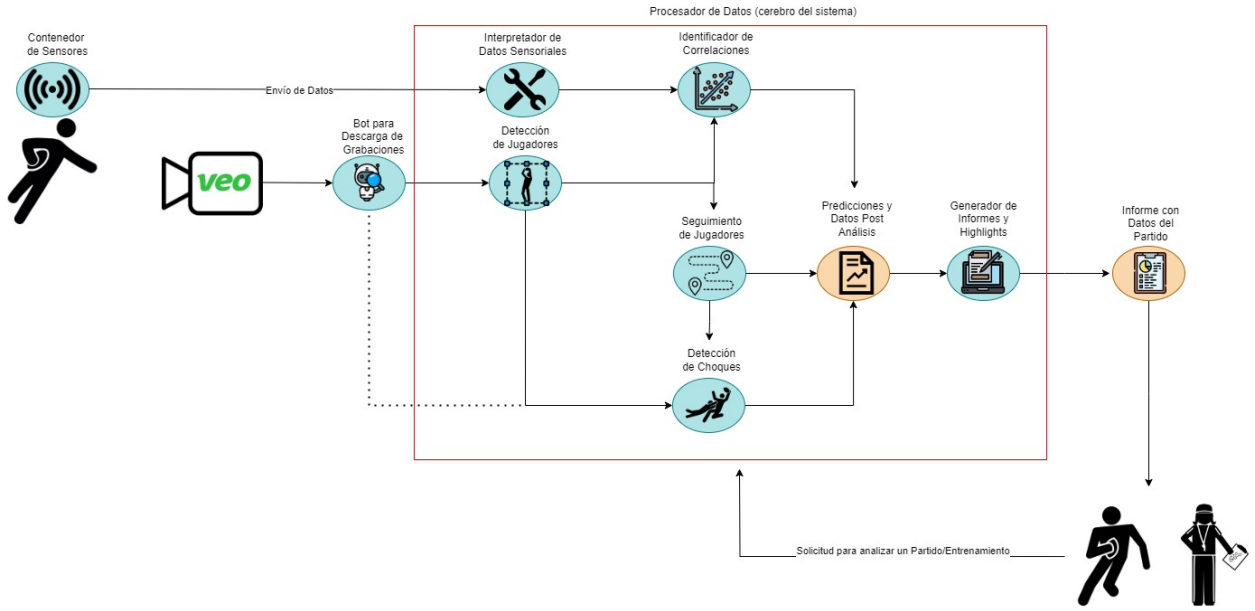


Figura 3.2: Diagrama de arquitectura general del sistema

3.4.3 Enumeración de Módulos

En la tabla 3.2 aparecen los módulos que componen el sistema completo, junto con el propósito de su existencia. Es fácil notar que son los módulos de la arquitectura.

Módulo	Propósito
Contenedor de Sensores	Módulo Hardware del sistema. Es el encargado de tener en su interior los sensores que toman datos de los jugadores y enviarlos usando un micro controlador
Interpretador de Datos Sensoriales	Módulo Software encargado de pre procesar los datos enviados por el micro controlador (programado en la IDLE de Arduino) para la correcta interpretación de los datos en el sistema
Bot para descarga de grabaciones	Ser una ayuda para agilizar la obtención de las grabaciones que serán analizadas
Detección de jugadores	Modulo que realizará las detecciones de interés en los jugadores: las instancias de estos, la segmentación de las instancias y generación de los keypoints de articulado
Identificador de Correlaciones	Encargado de pre procesar toda la data trabajada por los otros 3 módulos y encontrar las primeras correlaciones que servirán de base para la formación del informe a entregar al final del proceso del sistema
Seguimiento de Jugadores	Inteligencia Artificial que traza las líneas de seguimiento de cada jugador y calcula la velocidad y aceleración de los jugadores. Sirve como base para la detección de posibles choques
Detección de choques	Inteligencia Artificial que encuentra jugadas y momentos de la grabación que podría inducir lesiones inmediatas o futuras en determinado(s) jugador(es), mediante la identificación de colisiones, caídas y tackles
Generador de Informes y Momentos Destacados	Tomando todos los datos generados por las IA's y el pre procesado de los sensores, genera un informe con todos los datos y análisis relevantes para el entrenador y sus jugadores

Tabla 3.2: Módulos de la arquitectura del sistema

Los módulos desarrollados en esta memoria fueron todos los relacionados a video, es decir, el Bot para descarga de grabaciones, Detección de Jugadores, Seguimiento de Jugadores y Detección de Choques. Estos se describen en detalle a continuación.

3.5 Bot para descarga de grabaciones

- Propósito: Ser una ayuda para agilizar la obtención de las grabaciones que serán analizadas.
- Alcance: Además de obtener las grabaciones, el bot obtiene las etiquetas con los tiempos de momentos del partido, como los inicios de cada mitad, cuando ocurren los Scrums, Lineouts, Trys y Conversions, los cuales se muestran al momento de acceder a una grabación desde la página de Veo.
- Dependencias: No depende de ningún modulo.
- Supuestos: No hay errores al momento de ingresar los datos para hacer login en la página de Veo y también se cuenta con espacio suficiente para guardar las grabaciones en Google Drive.
- Restricciones: Solo se puede hacer descarga o de la grabación completa o de los clips generados. Si se desea tener ambos, se debe correr el módulo 2 veces con el mismo video cambiando la opción de descarga. También, solo se muestran las 20 primeras grabaciones hechas (las que aparecen en la página 1).
- Estructura General: Carga e ingreso a la vista de Login de Veo, ingreso de los datos para iniciar sesión. Luego se despliegan los títulos de las grabaciones, debiendo seleccionar una para luego elegir si se desea descargar la grabación completa o los clips. Al finalizar la descarga, se genera una lista con los momentos del partido indicando cuándo sucede cada uno.

3.5.1 Detalles del Bot

El bot es programado usando la biblioteca Selenium en Python, al ser fácil de configurar la rutina de seguimiento y de tener tiempos de espera dinámicos según vayan cargándose los elementos de la página. Es importante mencionar que la rutina no abre ventanas adicionales al script para mostrar lo que realiza y ejecuta. El funcionamiento de esta web scraping es el siguiente:

1. Ingresa a la vista Login de Veo.
2. Pide los datos de ingreso como input (correo y contraseña).
3. Entra a la vista de grabaciones y retorna una lista con los títulos de las grabaciones.

4. Dependiendo de la grabación de interés, se hace click en el título de la que se desea descargar.

5. Una vez cargada la vista de la grabación, se pregunta si se desea descargar la grabación completa o solo los clips generados por Veo.

6. Dependiendo de lo seleccionado, comienza el proceso de descarga hacia el espacio de Google Drive de la cuenta que está ejecutando el código.

7. Se genera una lista con los tiempos en que sucede cada momento del partido.

Es importante destacar que las etiquetas HTML de la vista de la grabación no han sido estáticos, han ido cambiado según diversas versiones de la página web de Veo, lo que impacta directamente en la obtención de los elementos con los que el bot interactúa. Esto ha llevado a modificar el código del bot en algunas ocasiones, con tal de obtener de forma más dinámica los elementos de esta vista.

3.6 Detección de jugadores

- Propósito: Encontrar un modelo pre entrenado para realizar identificaciones de calidad aceptable a los jugadores de rugby

- Alcance: Detección de jugadores de rugby en videos, Encontrar modelos adecuados para realizar buen enmascaramiento para segmentación de instancias y realizar el articulado de los jugadores

- Dependencias: En si no depende de otros módulos, pero si de forma indirecta del Bot de descarga de grabaciones

- Supuestos: Los videos están grabados en calidad HD (1920 x 1080) y los jugadores llevan vestimenta que permite diferenciar claramente los equipos a los que pertenece

- Restricciones: El solapamiento va a ser una condicionante al momento de la detección de los jugadores, así como su cercanía/lejanía de la cámara. También, el rendimiento del módulo puede variar dependiendo del hardware en que se ejecute

- Estructura General: Carga de clips (try, scrum, lineout, half start) y de la imagen de un jugador All Brad, para luego encontrar primero cual es el rango de colores HSV que representa el uniforme del club y los mejores modelos de Detectron2 para realizar la identificación de los jugadores, segmentación de instancia de estos y la representación de las articulaciones por keypoints.

3.6.1 Espacio de colores que representa uniforme All Brads

Como la detección en sí solo identifica personas y no logra discernir entre equipos, árbitros y entrenadores, se hace necesario encontrar una forma de saber cuándo un jugador es del equipo que nos interesa y se escoge utilizar el espacio de colores HSV, ya que además de ser el estándar para el trabajo de colores y generación desde cero de máscaras para segmentación en OpenCV, otorga un espacio de colores mucho más representativo que RGB e incluso en ocasiones que el código hexadecimal. En la figura 3.3 se encuentra la representación en 2 dimensiones de este espacio de colores que, en realidad se compone de 3, pero que resulta mucho más ilustrativo esta representación. HSV hace referencia al inglés para Hue, Saturation y Value (en español Matriz, Saturación y Valor).

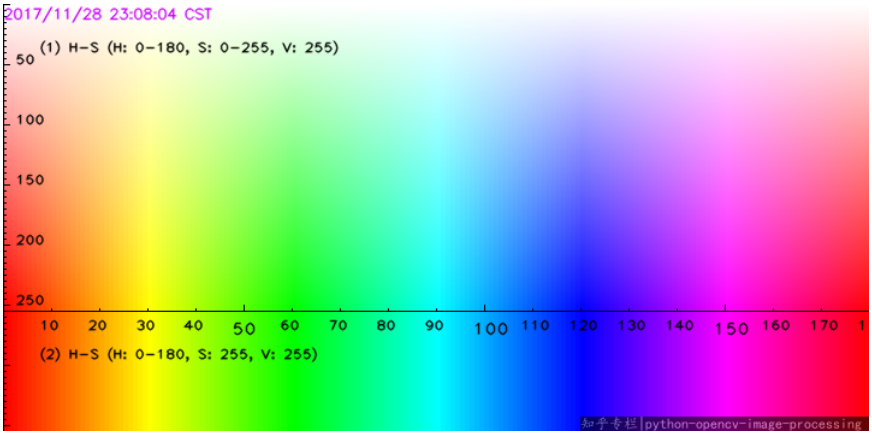
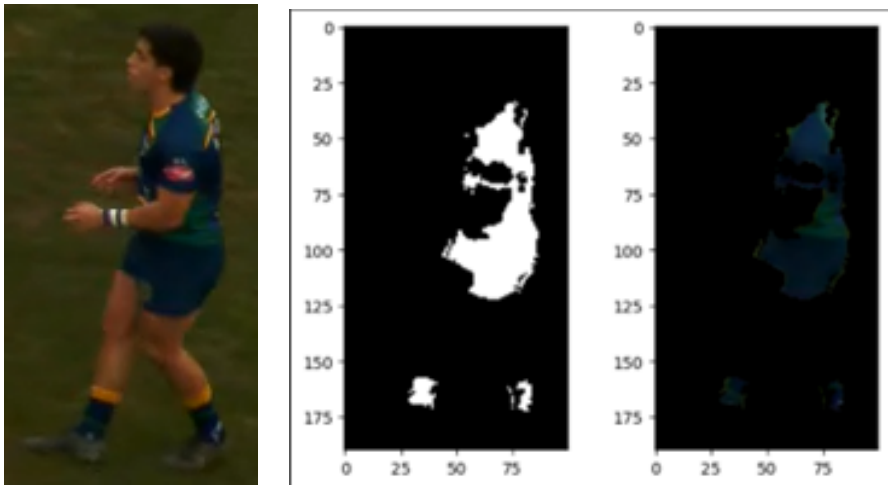


Figura 3.3: Espacio de colores HSV

Una vez encontrado todos los valores que pueden representar el uniforme de los All Brads (ya que no existe un valor único en HSV), se determinó que el rango en que se define el uniforme va entre (42, 100, 20) mínimo y (106, 255, 68) máximo, aunque un límite máximo de (125, 255, 255) da el mismo resultado. En la figura 3.4 se observa la obtención de la máscara según este rango de colores junto con la imagen original en que se basó este análisis, en donde si bien no aparece representado todo el uniforme (faltan los auspiciadores que son de color amarillo), basta para identificar el uniformado del equipo de nuestro interés.



(Jugador) (Mascara del jugador)

Figura 3.4: Generación de mascara por rango de colores HSV

Ahora si, se puede proceder con la identificación de jugadores.

3.6.2 Modelos de Detección de jugadores

Se utilizará Detectron2 como fue mencionado en el marco teórico. Es importante mencionar que los conjuntos de modelos de segmentación de máscaras y los de keypoints tienen como base los cuadros delimitadores, por lo que si bien es posible usarlos en los módulos posteriores, estos no funcionan de manera óptima para el seguimiento de instancias, a diferencia de los modelos entrenados específicamente para la detección de objetos. Como tal, Detectron2 ofrece 10 modelos de detección basados en Fast-RCNN [24], 3 en RetinaNet [25] y 3 en RPN con Fast-RCNN. Estos últimos fueron descartados por no contar con un indicador de puntaje (score) sobre qué tan acertada es la predicción y por no tener una configuración fija del umbral de confianza mínimo esperado de las predicciones, lo que genera un montón de identificaciones a diferencia de los otros 2 grupos de modelos. Para analizar los modelos, se escogieron 4 clips de un partido que contenía características distintivas entre sí:

- Try (600 fotogramas): En este Clip hay 1 colisión casual entre 2 jugadores, hay mucho solapamiento por el movimiento de quien lleva el balón y un leve acoplamiento al comienzo del clip
- Scrum (599 fotogramas): Presencia de un acoplamiento largo de jugadores y por ende mucho solapamiento. El scrum ocurre lejano a la cámara
- Lineout (599 fotogramas): Varias situaciones de solapamiento y acople, pero a diferencia del scrum, aquí todo ocurre cercano a la cámara

Modelo	Promedio de identificaciónes por fotograma	Promedio de puntaje en las identifiicaciónes	Desviación Estándar de Detecciones	Desviación Estándar promedio de puntaje	Promedio de Solapamiento (IoU)	Detecciones Totales	Tiempo Ejecución [seg]
faster_rcnn R_50_C4_1x	21.68	0.85	4.09	0.15	0.0071	13011	286
faster_rcnn R_50_DC5_1x	22.12	0.86	4.24	0.14	0.0074	13275	156
faster_rcnn R_50_FPN_1x	18.75	0.86	3.34	0.13	0.0034	11248	94
faster_rcnn R_50_C4_3x	21.87	0.86	3.92	0.14	0.0065	13123	279
faster_rcnn R_50_DC5_3x	21.88	0.87	4.48	0.14	0.0072	13127	161
faster_rcnn R_50_FPN_3x	19.09	0.86	3.25	0.13	0.0036	11453	93
faster_rcnn R_101_C4_3x	21.29	0.88	4.28	0.14	0.0061	12772	297
faster_rcnn R_101_DC5_3x	21.43	0.88	4.38	0.14	0.0065	12859	176
faster_rcnn R_101_FPN_3x	19.05	0.88	3.24	0.12	0.0031	11428	117
faster_rcnn X_101_32x8d_FPN_3x	18.82	0.88	3.06	0.12	0.0027	11294	187
retinanet R_50_FPN_1x	11.50	0.66	2.74	0.11	0.0014	6901	69
retinanet R_50_FPN_3x	11.29	0.68	2.81	0.11	0.0021	6775	100
retinanet R_101_FPN_3x	13.68	0.68	3.32	0.11	0.0024	8206	129

Tabla 3.3: Métricas de desempeño en Try

- Half Start (899 fotogramas): Combinación de las 3 situaciones de cada clip anterior, junto con varios choques, caídas y tackles cercanos a la cámara.

Y para evaluar cada modelo además del renderizado de las identificaciones para ver que tan bien es su desempeño, se midieron diversas métricas: Número promedio de identificaciones por fotograma, Confianza (puntaje) promedio en identificaciones, Desviación estándar en número de detecciones, Desviación estándar promedio de los puntajes, Promedio de solapamiento entre cajas (IoU) y tiempo de ejecución (en este último para su uso en el resto del algoritmo se debe considerar entre 10 a 20 segundos menos, ya que no se realizan los cálculos de IoU, ni el número de identificaciones). Los números de cada modelo para Try, Scrum, Lineout y Half Start se encuentran en las tablas 3.3, 3.4 , 3.5 y 3.6 respectivamente.

Se puede observar un mejor desempeño en los modelos de faster rcnn en comparación con los de

Modelo	Promedio de identificaciónes por fotograma	Promedio de puntaje en las identifiaciónes	Desviación Estándar de Detecciones	Desviación Estándar promedio de puntaje	Promedio de Solapamiento (IoU)	Detecciones Totales	Tiempo Ejecución [seg]
faster_rcnn R_50_C4_1x	17.76	0.83	6.61	0.15	0.0033	9437	301
faster_rcnn R_50_DC5_1x	18.13	0.84	6.07	0.14	0.0076	10857	151
faster_rcnn R_50_FPN_1x	15.26	0.83	4.77	0.14	0.0034	9142	95
faster_rcnn R_50_C4_3x	17.95	0.85	6.51	0.14	0.0053	10752	299
faster_rcnn R_50_DC5_3x	18.91	0.84	5.85	0.15	0.0077	11327	155
faster_rcnn R_50_FPN_3x	15.55	0.82	4.32	0.15	0.0032	9315	94
faster_rcnn R_101_C4_3x	18.22	0.85	6.36	0.14	0.0052	10913	324
faster_rcnn R_101_DC5_3x	18.92	0.84	5.25	0.15	0.0072	11336	181
faster_rcnn R_101_FPN_3x	15.93	0.85	4.52	0.14	0.0030	9541	121
faster_rcnn X_101_32x8d_FPN_3x	15.96	0.86	3.80	0.13	0.0029	9562	202
retinanet R_50_FPN_1x	7.10	0.67	1.87	0.12	0.0016	4251	97
retinanet R_50_FPN_3x	6.85	0.70	1.43	0.12	0.0027	4102	97
retinanet R_101_FPN_3x	7.97	0.70	1.53	0.12	0.0020	4773	125

Tabla 3.4: Métricas de desempeño en Scrum

Modelo	**Promedio de identificaciónes por fotograma**	**Promedio de puntaje en las identifiifaciónes**	**Desviación Estándar de Detecciones**	**Desviación Estándar promedio de puntaje**	**Promedio de Solapamiento (IoU)**	**Detecciones Totales**	**Tiempo Ejecución [seg]**
faster_rcnn R_50_C4_1x	25.42	0.86	2.78	0.15	0.0132	15225	298
faster_rcnn R_50_DC5_1x	26.08	0.86	2.88	0.15	0.0117	15623	154
faster_rcnn R_50_FPN_1x	21.97	0.85	2.07	0.15	0.0076	13163	95
faster_rcnn R_50_C4_3x	27.11	0.86	2.90	0.15	0.0116	16238	297
faster_rcnn R_50_DC5_3x	26.97	0.86	2.96	0.15	0.0124	16156	154
faster_rcnn R_50_FPN_3x	22.52	0.87	2.11	0.14	0.0082	13487	94
faster_rcnn R_101_C4_3x	24.49	0.88	2.45	0.14	0.0087	14669	311
faster_rcnn R_101_DC5_3x	24.24	0.88	2.32	0.14	0.0107	14520	180
faster_rcnn R_101_FPN_3x	20.90	0.88	1.52	0.13	0.0076	12520	121
faster_rcnn X_101_32x8d_FPN_3x	20.95	0.89	1.81	0.12	0.0070	12551	196
retinanet R_50_FPN_1x	13.17	0.70	2.05	0.12	0.0064	7888	100
retinanet R_50_FPN_3x	14.20	0.72	2.03	0.12	0.0096	8508	97
retinanet R_101_FPN_3x	13.99	0.73	1.89	0.11	0.0093	8382	123

Tabla 3.5: Métricas de desempeño en Lineout

Modelo	Promedio de identificaciónes por fotograma	Promedio de puntaje en las identifiaciónes	Desviación Estándar de Detecciones	Desviación Estándar promedio de puntaje	Promedio de Solapamiento (IoU)	Detecciones Totales	Tiempo Ejecución [seg]
faster_rcnn R_50_C4_1x	25.80	0.84	3.29	0.15	0.0061	23195	447
faster_rcnn R_50_DC5_1x	26.50	0.84	3.12	0.15	0.0063	23825	233
faster_rcnn R_50_FPN_1x	21.74	0.83	2.66	0.15	0.0032	19545	141
faster_rcnn R_50_C4_3x	26.85	0.85	3.25	0.15	0.0052	24135	447
faster_rcnn R_50_DC5_3x	26.29	0.85	3.12	0.15	0.0057	23634	229
faster_rcnn R_50_FPN_3x	22.30	0.83	2.56	0.15	0.0030	20046	143
faster_rcnn R_101_C4_3x	25.28	0.85	3.26	0.15	0.0048	22726	486
faster_rcnn R_101_DC5_3x	26.10	0.84	2.90	0.15	0.0054	23466	262
faster_rcnn R_101_FPN_3x	22.91	0.84	2.57	0.15	0.0026	20599	181
faster_rcnn X_101_32x8d_FPN_3x	21.89	0.84	2.96	0.15	0.0025	19680	292
retinanet R_50_FPN_1x	10.61	0.68	2.02	0.12	0.0026	9538	148
retinanet R_50_FPN_3x	11.45	0.69	2.02	0.12	0.0031	10290	148
retinanet R_101_FPN_3x	10.98	0.69	1.86	0.12	0.0026	9874	184

Tabla 3.6: Métricas de desempeño en Half Start

retinanet, principalmente en lo que respecta a los puntajes. Ahora para seleccionar los modelos a usarse en los análisis posteriores, se va a considerar la cantidad de solapamiento promedio que hay (ya que será útil en la identificación de colisiones), las detecciones totales y por fotograma (ya que en la gran mayoría de casos sobrepasa el 0.84 de puntaje) y la desviación estándar en el número de detecciones por fotograma, siempre buscando concordancia entre los parámetros (es decir, de que sirve que la desviación de detecciones sea la más baja si hay pocas detecciones totales en comparación al resto, por ejemplo). Considerando lo anteriormente mencionado, junto con lo que se aprecia al representar visualmente las detecciones, se destacan los siguientes modelos:

1. faster_rcnn_R 101_DC5_3x: El punto medio entre rendimiento y tiempo de ejecución. Para imágenes lejanas es donde más destaca.

2. faster_rcnn_R_50_DC5_1x: Uno de los más rápidos en ejecutarse con mejor desempeño, en contextos cercanos y en general es el que posee mayor solapamiento y más detecciones cuando no hay acoplamiento y solo solapamiento por movimiento de jugadores

3. faster_rcnn_R_50_C4_3x: A pesar de ser el que más demora en su ejecución, en el contexto del modelo anterior es quien realiza más detecciones.

4. faster_rcnn_R_50_DC5_3x: Rápido, un punto medio en el desempeño para todos los contextos, aunque no da para superar al primer modelo

Todos estos modelos se considerarán para los análisis de seguimiento, teniendo como prioridad el uso de faster_rcnn_R 101_DC5_3x por ser un intermedio para nuestros propósitos

3.6.3 Modelos de Segmentación de jugadores

Luego se procede a encontrar los mejores modelos de segmentación para nuestro contexto, aunque a diferencia del modelo de detección, este se hizo únicamente mirando los resultados de renderizado, ya que los modelos no poseen métricas de puntaje y tampoco de identificación de instancias, simplemente encuentra las máscaras sin tener un indicativo de su cantidad. De los 10 modelos ofrecidos por Detectron2, los 3 mejores modelos de segmentación fueron:

1. mask_rcnn_R_101_DC5_3x: Para situaciones como el scrum y el lineout, donde varios jugadores se amontonan funciona de muy buena forma. Además, es capaz de realizar buen enmascaramiento para jugadores que se encuentran en vista parcial por el solapamiento.

2. mask_rcnn_R_50_C4_3x: Otro modelo que funciona bien en situaciones de amontonamiento, pero no tan bien como el anterior para el solapamiento

3. mask_rcnn_R_50_DC5_3x: De todos los modelos, es el más pesado, sin embargo, ello lo compensa con su rendimiento. Si el tiempo de ejecución no es determinante, es una opción que en ocasiones llega a ser mejor que el primero modelo mencionado.

Se decide para usos posteriores, usar el primero de la lista si el tiempo es condicionante, o el tercero si es que no es factor.

3.6.4 Modelos de articulado por keypoints en jugadores

Finalmente se procede con el análisis para los modelos que ilustran las articulaciones a partir de keypoints, bajo las mismas condiciones que los modelos de segmentación de mascara. De estos modelos que son solo 4, los 3 destacados fueron:

1. keypoint_rcnn_X_101_32x8d_FPN_3x: Aunque es el más pesado, es el mejor por diferencia, ya que en instancia de choques no pierde la detección de keypoints, aunque las instancias se encuentren alejadas de la cámara.

2. keypoint_rcnn_R_50_FPN_3x: Para casos como el scrum, la identificación era aceptable, a pesar de que se perdiesen instancias, estas se reencontraban inmediatamente, generando una especie de tintineo en el video con la detección de las instancias en el scrum.

3. keypoint_rcnn_R_50_FPN_1x: Para choques funciona basta bien, sin embargo es algo sensible al solapamiento, por lo que en condiciones como caídas o tackles puede no ser útil

Para usos posteriores, se decide continuar con keypoint_rcnn_X_101_32x8d_FPN_3x

3.7 Seguimiento de Jugadores

- Propósito: Trazar las líneas de seguimiento de cada jugador y calcular la velocidad y la aceleración de los jugadores.

- Alcance: Es uno de los componentes centrales de la solución, ya que además de usarse para calcular las velocidades y aceleraciones por jugador (a partir de los pixeles recorridos por instancia), también es utilizado como punto de partida para la identificación de Choques

- Dependencias: Modulo de detección de jugadores

- Supuestos: En las grabaciones, las tomas de los jugadores se encuentran en constante solapamiento, debido al ángulo en que se instala la cámara para realizar las grabaciones, por lo que la pérdida de instancias va a condicionar la calidad del seguimiento

- Restricciones: La precisión de los cálculos puede verse afectada por la calidad de las grabaciones (solapamiento). También las limitaciones en la capacidad de cómputo pueden afectar la velocidad de procesamiento y análisis.

- : Estructura General: Recibiendo las identificaciones de un determinado clip, encontrar que modelo hace mejores seguimientos (con menos ReID). Luego, se realiza el trazado de línea que representa el recorrido de los jugadores como base para luego, calcular la velocidad y la aceleración por jugador

3.7.1 Modelos de seguimiento en BoxMot

La importancia de realizar buenas detecciones, es que estas influyen directamente en un buen seguimiento de instancias, aunque en los 4 modelos descartados de BoxMot, a pesar de tener identificaciones con buen puntaje, el seguimiento no era acorde. También cabe mencionar que no se notó una diferencia muy significativa en los modelos cuando se utilizaban los diferentes modelos de re identificación que ofrece BoxMot. De los 6 modelos ofrecidos por el framework, se destacan ampliamente 2:

1. BotSort [30]: Buen desempeño en seguimiento cuando las identificaciones son cercanas a la cámara, bueno en retomar la identificación post solapamiento, pero tiene algunas dificultades en instancias lejanas a la cámara. Aun así, es el de mejor desempeño.

2. StrongSort [31]: Como su nombre lo indica, el seguimiento en este modelo es fuerte y robusto, incluso para instancias lejanas a la cámara. Sin embargo, para solapamientos en multitud de gente (como en choque múltiple o scrum) y en cambios de pose, el modelo tiende fácilmente a la re asignación de ID's para una misma instancia, generando un montón de ID's lo que dificulta el seguimiento. Si las detecciones en las que se basan los algoritmos de seguimiento fuesen de mejor calidad, StrongSort podría funcionar muchísimo mejor que BotSort.

Resulta interesante mencionar lo ocurrido al probar ByteTrack [32], el cual es casi un estándar para el seguimiento en deporte. Primero, es el modelo menos personalizable en su configuración (no admite modelos de ReID) y segundo, su desempeño era mucho más bajo en comparación a los 2 modelos mencionados: muchos ID's (más que StrongSort) y dificultad en identificar instancias luego del solapamiento: a casi todas les asignaba un nuevo ID, sobre todo a las instancias alejadas de la cámara. Teniendo estos modelos seleccionados, se opta por continuar con BotSort y su configuración con mobilenet (reid_weights="mobilenetv2_x1_4"), ya que se sufre menos de ReID masiva.

En lo que respecta al trazado de la línea de trayectoria, BoxMot ofrece la opción de realizar este trazado de forma automática, al momento de dibujar los resultados del seguimiento, definiendo verdadero la opción show_trajectories, la cual habilita la ejecución de la función plot_trackers_trajectories.

3.7.2 Cálculo de velocidad y aceleración

En las grabaciones que se disponen, se sigue todo el tiempo el movimiento del balón, generando la sensación de giro de cámara que uno observa cuando ve cualquier partido de rugby, futbol, o basquetbol. Es importante tener en cuenta este punto, ya que de obviarse se tendrían valores de velocidad y aceleración bastante alejados de lo que realmente está sucediendo en cancha, por ende se hace necesario compensar este movimiento para realizar cálculos lo más exactos posibles, más aún al no lograr realizar una transformación geométrica para tener una vista plana de la cancha (no es posible tener referencias constantes, como esquinas para generar áreas a transformar) y también la detección de las líneas delimitadoras de la cancha no es constante debido a lo mal dibujadas que se encuentran. El movimiento de la cámara en esta ocasión es encontrado al determinar cuánto varían las 10 columnas superiores de pixeles y las 150 columnas inferiores, y como el interés es ver la variación de la velocidad (sin importar que no sea 100% exacta), es que se idea la siguiente rutina para encontrar la velocidad y aceleración:

1. Encontrar, luego de convertir en gris el fotograma original, las características más relevantes de los límites superiores e inferiores previamente definidos.

2. Encontrar el flujo óptico por el método de Lucas-Kanade para extraer las nuevas características que definirán cuánto se ha movido la cámara en las coordenadas bidimensionales (x,y), donde x será positivo al moverse a la derecha, negativo a la izquierda, e y será positivo al moverse hacia arriba y negativo hacia abajo.

3. Se encuentran las coordenadas del centro de las identificaciones de los jugadores, junto con el área de estas. La razón entre el área y un factor de escalado, determinan la contribución del movimiento del centro para el cálculo de la velocidad y aceleración.

4. Cuando el centro de las identificaciones de los jugadores cambia su posición, sus valores en X e Y se restan con los valores en X e Y del movimiento de la cámara.

Así, la velocidad se define como el cambio en la posición dividido por el tiempo entre dos fotogramas consecutivos. Para las componentes x e y, las ecuaciones son:

$$v_x = \frac{x_t - x_{t-1}}{\Delta t}, \quad v_y = \frac{y_t - y_{t-1}}{\Delta t}$$

donde:

- x_t e y_t: Posición del centro de la caja englobante del jugador identificado en el fotograma actual;

- x_{t-1} e y_{t-1}: Posición del centro de la caja englobante del jugador identificado en el fotograma anterior;
- $\Delta t = \frac{1}{\text{fps}}$: Intervalo de tiempo entre dos fotogramas consecutivos;

y la aceleración se define como el cambio en la velocidad dividido por el tiempo entre dos fotogramas consecutivos. Las ecuaciones para las componentes x e y son:

$$a_x = \frac{v_{x,t} - v_{x,t-1}}{\Delta t}; \quad a_y = \frac{v_{y,t} - v_{y,t-1}}{\Delta t}$$

,

donde:

- $v_{x,t}$ y $v_{y,t}$: Velocidades en el fotograma actual;
- $v_{x,t-1}$ y $v_{y,t-1}$: Velocidades en el fotograma anterior.

3.8 Detección de Choques

- Propósito: Identificar jugadas y momentos en las grabaciones que podrían inducir lesiones, a partir de la identificación de 4 eventos en los fotogramas pasados: colisión, caída, tackle, o no_choque, a partir de la prueba de diversas arquitecturas IA pre entrenadas con un vasto dataset.
- Alcance: Además de la identificación de los 4 eventos mencionados en el propósito, el módulo también genera clips de 2 segundos a partir de estos fotogramas, para identificar mediante los keypoints las articulaciones afectadas en los choques y la gravedad de los mismos.
- Dependencias: Detección de jugadores (por los modelos de keypoints), el Bot para descarga de videos (de forma indirecta) y de seguimiento de jugadores
- Supuestos: Se cuenta con un conjunto de datos de entrenamiento adecuado para la estimación de poses en contextos de rugby. Este conjunto de datos es lo suficientemente global y de alta definición para ser aplicado en nuestro contexto
- Restricciones: La precisión de las predicciones puede verse afectada por la calidad y cantidad de los datos de entrenamiento.
- Estructura General: Recibiendo el dataset en primera instancia, se procede a realizar el pre procesamiento de las imágenes para adaptarlas al tamaño de las grabaciones, encontrar los keypoints de estas imágenes y en segunda instancia las máscaras de segmentación. Luego,

se crean 4 arquitecturas con PyTorch para ser entrenadas con las imágenes del dataset y analizando sus puntajes de desempeño. Posteriormente, se pasa un video para correr un proceso diferente, en el que se extraen fotogramas con posibles colisiones, caídas o tackles, los cuales son clasificados con alguno de los modelos entrenados. Dependiendo el fotograma clasificado que sea de interés, de este finalmente se crea un video de 2 segundos para analizar las partes del cuerpo involucradas en el impacto, para posteriormente medir su gravedad

3.8.1 Pre procesamiento del Dataset

El dataset para el entrenamiento de las arquitecturas consiste en 4 carpetas con 293 imágenes cada una de alta calidad, las cuales contienen imágenes de colisiones, caídas, tackles y no_choques, puesto a que hay poses repetitivas en las 3 primeras clases mencionadas. Debido al solapamiento y de la acción misma del choque, es posible que algunos keypoints se pierdan en las imágenes a analizar y del entrenamiento, lo que podría condicionar el desempeño de los modelos al relacionar la ausencia de keypoints con la ocurrencia de determinadas clases. Por lo que las métricas a analizar para predecir los eventos de choque serán:

- Distancias hombros-rodillas
- Angulo del Torso
- Ángulos de ambos brazos

Los keypoints del Dataset son adaptados para tener las dimensiones del vídeo (1920 x 1080), aunque esto variará en pruebas posteriores. Luego, al momento de generar los keypoints, se observa que a pesar de tener la misma cantidad de imágenes en las 4 clases, la clase no_choques es la que termina generando más muestras (6167 para ser exactos), por lo que para evitar desigualar el número de imágenes por clase, se opta por generar duplicados para las otras 3 clases, haciendo muy pequeñas variaciones en el escalado de la distancia brazo/hombro, ángulos de los brazos y rotación en el ángulo del torso (esto explicará los resultados obtenidos en el desempeño de los modelos).

3.8.2 Arquitectura Transformer

El primer modelo es un Transformer [33] diseñado con PyTorch. Esta arquitectura tiene la capacidad de identificar patrones no lineales en su input, siendo esa la razón de su implementación en este caso. En la figura 3.5 se representan las capas del modelo, el cual es simple puesto que el Transformer se usa para clasificación, lo cual hace que se use un encoder y no un decoder.

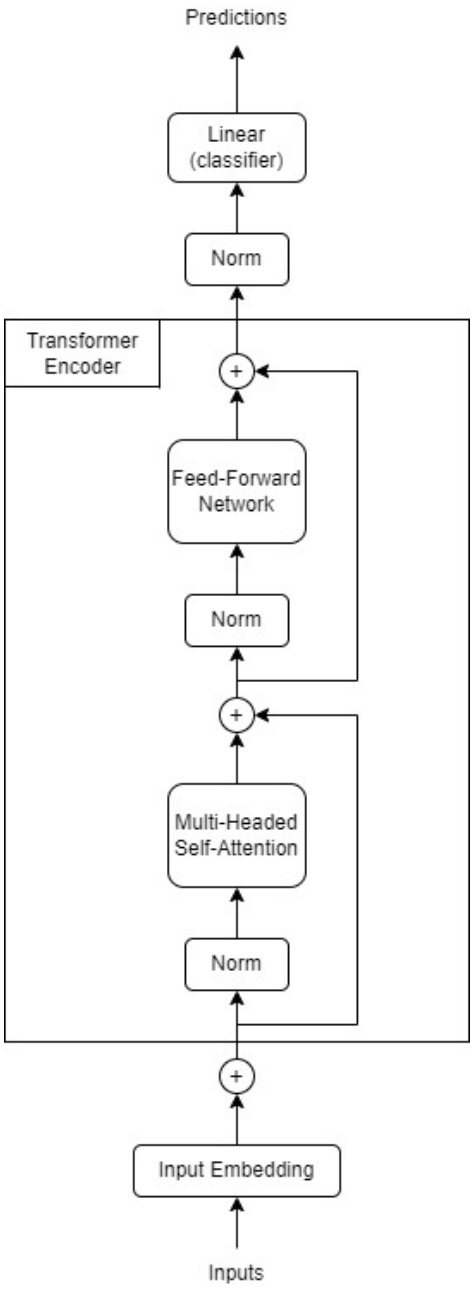


Figura 3.5: Arquitectura del modelo Transformer

Paralelo a este modelo, se crea una versión la cual incluye una capa de codificación posicional, justo antes de ingresar a la capa del Transformer, con el fin de asignar mejor los pesos de cada característica de keypoints, especialmente con la posición a través de los senos y cosenos de la misma. Para ello, es necesario saber el número total de muestras. Luego, la capa de transformers permanece igual, pero previo a la última capa de normalización se agrega una capa lineal como grupo de atención. Su arquitectura se representa en la figura 3.6

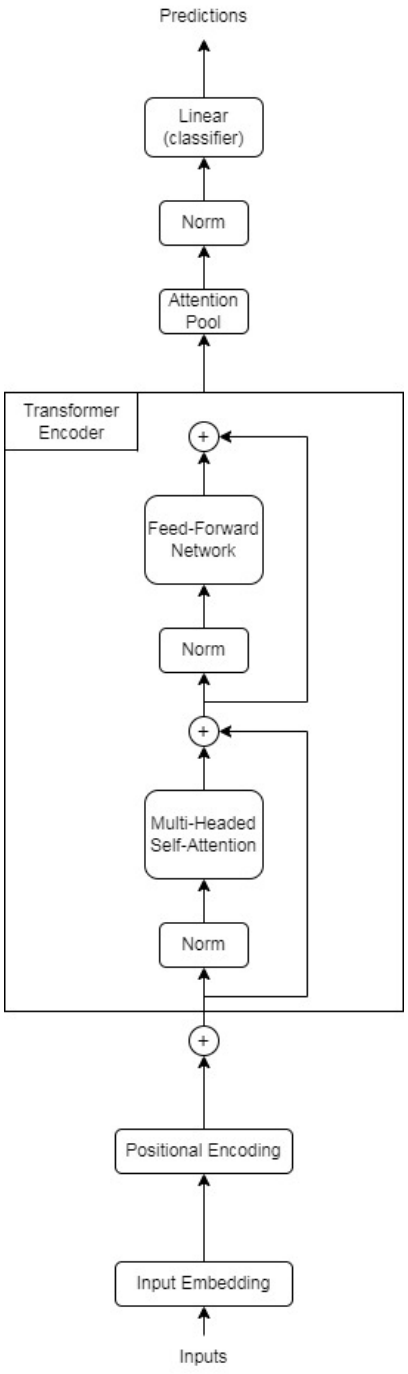


Figura 3.6: Arquitectura Transformer con codificación posicional

3.8.3 Arquitectura CNN "Simple"

En esta primera arquitectura, como los datos de ingreso son cálculos obtenidos de los keypoints, se consideran todas las capas unidimensionales. Además, cabe mencionar que todas las redes neuronales convolucionales tendrán batch_size 32. La arquitectura de esta primera CNN se ilustra

en la figura 3.7.

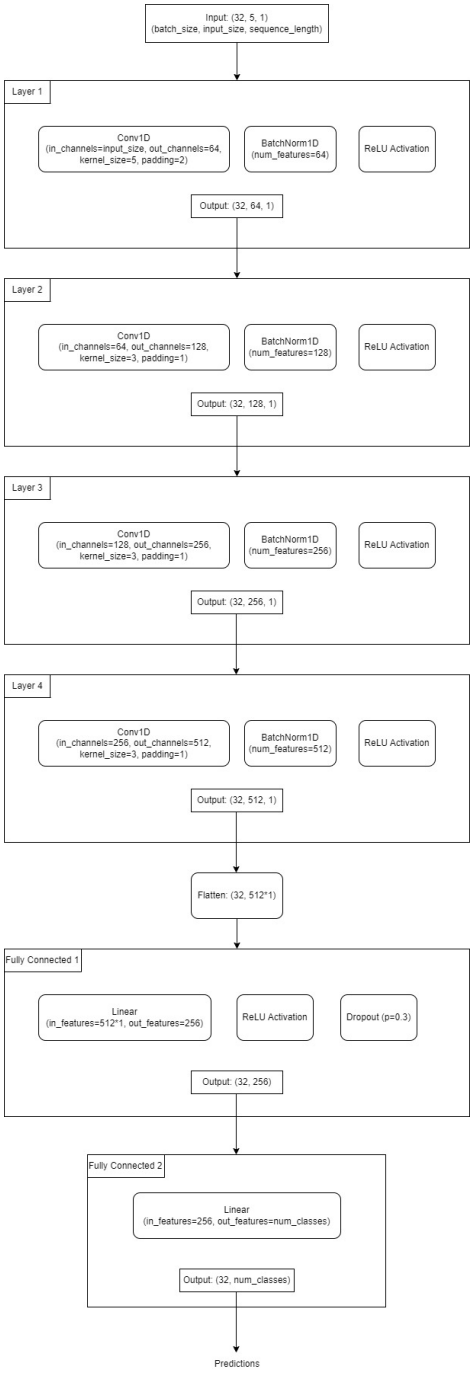


Figura 3.7: Arquitectura del primer CNN

3.8.4 Arquitectura CNN con aumento artificial de dimensiones

Si bien esta segunda red es bastante parecida a la primera, la gran diferencia es que al momento de iniciarla se define un sequence_lenght de 4, para simular que la red tiene 4 dimensiones iniciales por

interpolación lineal. Estas dimensiones decrecen hasta llegar a 1, como se ilustra en la arquitectura de la figura 3.8

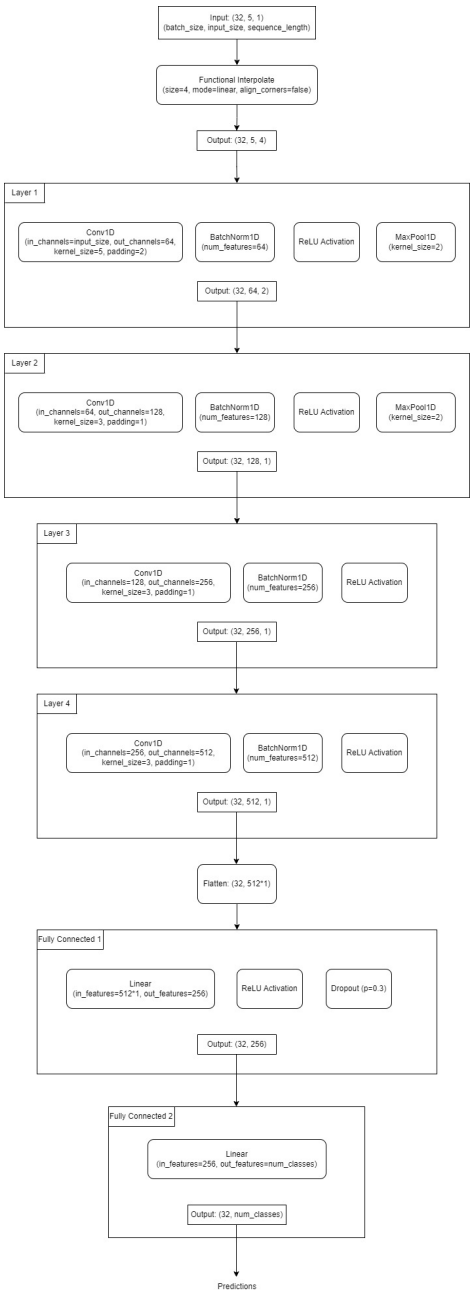


Figura 3.8: Arquitectura del primer CNN con aumento artificial de dimensiones

3.8.5 Arquitectura MLP

Esta vendría a ser la arquitectura más simple, ya que consiste únicamente en 3 capas completamente conectadas con dropouts entremedio antes del final, lo cual hace que no sea una CNN como tal. Su

diagrama de arquitectura se encuentra en la figura 3.9

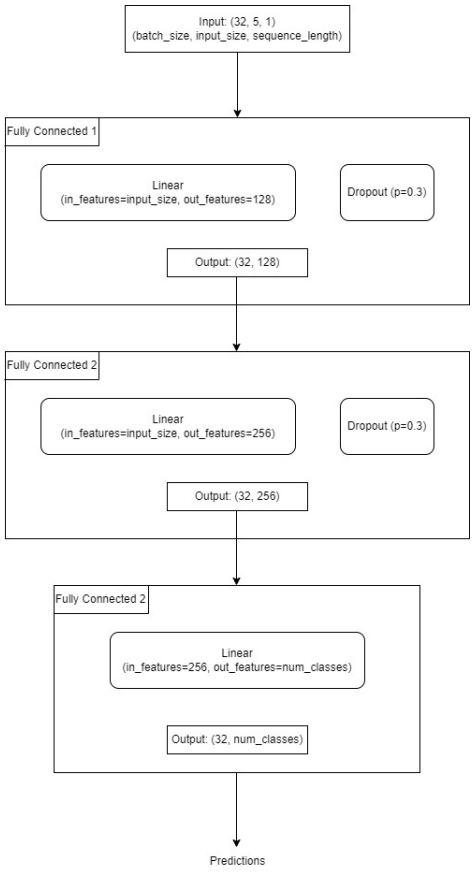


Figura 3.9: Arquitectura MLP con 3 capas fc

3.8.6 Entrenamiento y resultados

Para el entrenamiento, de ahora en adelante las mediciones encontradas en los keypoints por el proceso descrito en la sección 3.8.1 se llamarán "features". Una vez formada la lista de features para cada clase, se hizo la separación entre entrenamiento y validación (train y valid en inglés y usado globalmente para referirse a la separación de datos) para los features y labels (etiquetas) usando el 80% del tamaño en ambos y se procedió a igualar la cantidad de características para las 3 clases con menos muestras que no_choques.

El optimizador usado fue Adam, el criterio de pérdida fue una versión suavizada de la entropía cruzada, basada en probabilidad logarítmica calculada a partir del log_softmax ofrecido por PyTorch y se utilizó un planificador para ajustar el aprendizaje cuando la pérdida del puntaje de validación no progresara significativamente después de 5 ciclos. Entrenando todos los modelos a 50 ciclos, los resultados obtenidos fueron los siguientes:

1. Transformers: Val Accuracy=0,4137, demás puntajes en la tabla 3.7

	precision	**recall**	**f1-score**
colisión	0.29	0.22	0.25
caída	0.25	0.38	0.30
tackle	0.29	0.32	0.30
no_choque	0.60	0.54	0.57
promedio macro	0.36	0.36	0.36
peso promedio	0.43	0.41	0.42

Tabla 3.7: Desempeño Transformers

	precision	**recall**	**f1-score**
colisión	0.27	0.26	0.27
caída	0.25	0.38	0.30
tackle	0.29	0.32	0.30
no_choque	0.60	0.52	0.55
promedio macro	0.34	0.35	0.34
peso promedio	0.42	0.40	0.41

Tabla 3.8: Desempeño CNN Simple

2. CNN "Simple": Val Accuracy=0,3968, demás puntajes en la tabla 3.8
3. CNN con aumento de dimensión: Val Accuracy=0,4098, demás puntajes en la tabla 3.9
4. MLP: Val Accuracy=0,3791, demás puntajes en la tabla 3.10

De los resultados obtenidos, rápidamente se pueden sacar algunas ideas:

- La generación de muestras duplicadas ligeramente diferentes a las originales no logra equilibrar el sesgo que tiene el sistema por identificar la clase con mayores muestras

	precision	**recall**	**f1-score**
colisión	0.28	0.21	0.24
caída	0.26	0.36	0.31
tackle	0.26	0.32	0.28
no_choque	0.60	0.55	0.57
promedio macro	0.35	0.36	0.35
peso promedio	0.43	0.41	0.41

Tabla 3.9: Desempeño CNN simple con aumento de dimensión

	precision	**recall**	**f1-score**
colisión	0.29	0.27	0.28
caída	0.25	0.38	0.29
tackle	0.28	0.24	0.26
no_choque	0.61	0.50	0.55
promedio macro	0.34	0.35	0.34
peso promedio	0.42	0.38	0.39

Tabla 3.10: Desempeño MLP

- El uso o no de convoluciones en los modelos mejora en algunas centésimas solamente el desempeño
- El uso solo de datos de keypoints no resulta suficiente para un buen entrenamiento de los modelos

Luego, se procede a incluir las máscaras generadas por Detectron2 (sin re-dimensionar las imágenes), en el modelo de detección de choques, con el fin de enriquecer el proceso de entrenamiento. Las máscaras fueron tratadas como contornos a los cuales se les extraían datos geométricos de interés, los cuales fueron:

- **Transformación de Fourier discreta**, para comprimir los contornos conservando su forma original;
- **Momentos Hu** para extraer características invariantes a rotaciones, traslaciones y escalados del contorno;
- **HOG (Histograma de Gradientes Orientados)** para generalizar cada clase al capturar las direcciones principales de los bordes;
- **Embeddings con ResNet18 [34] pre-entrenado** para obtener vectores de características.

. Al usar las máscaras en los modelos de Transformer, su rendimiento en el entrenamiento fue muy bajo, pudiendo clasificar solo 1 clase y con un puntaje de exactitud bajo el 10%. Con las CNN no hubieron diferencias significativas en su entrenamiento, sin embargo en MLP si hubo una mejora considerable: Para los momentos Hu y la transformada discreta de Fourier, el rendimiento decayó como en los otros modelos, pero al introducir únicamente los HOG, se obtuvieron los puntajes ilustrados en la tabla 3.11, los que se traducen en un Val Accuracy = 0.5593.

Con estos resultados, se esperaba que al juntar los HOG con los embeddings de ResNet18 se pudiera alcanzar aun una mejor clasificación, sin embargo esto no ocurrió, ya que se obtuvo un Val Accuracy = 0.2213, y los puntajes de clasificación de la tabla 3.12

	precision	**recall**	**f1-score**
colisión	0.56	0.42	0.48
caída	0.61	0.80	0.69
tackle	0.51	0.56	0.53
no_choque	0.55	0.46	0.50
promedio macro	0.56	0.56	0.55
peso promedio	0.56	0.56	0.55

Tabla 3.11: Desempeño MLP entrenando con HOG de Mascaras

	precision	**recall**	**f1-score**
colisión	0.22	0.20	0.21
caída	0.33	0.02	0.03
tackle	0.14	0.02	0.03
no_choque	0.22	0.70	0.34
promedio macro	0.23	0.24	0.15
peso promedio	0.23	0.22	0.15

Tabla 3.12: Desempeño MLP entrenando con HOG y embeddings de Mascaras

Finalmente, hay que mencionar que hubo una diferencia en el número de muestras respecto a los generados para keypoints, ya que al ser contornos de la imagen, no se consideró la separación entre instancias de personas, por lo que el tamaño de las muestras para cada clase terminó siendo el número de imágenes del dataset.

Otra variante probada para encontrar mejores puntajes de clasificación fue la de no realizar re-dimensionado de las imágenes y keypoints del dataset (en transformers). Para el primer transformer se tuvo un Val Accuracy=0.4348 y para el segundo un Val Accuracy=0.4363. El detalle de ambos modelos se muestra en las tablas 3.13 y 3.14 respectivamente.

Es importante mencionar que a diferencia de los modelos entrenados con las imágenes re-dimensio-

	precision	**recall**	**f1-score**
colisión	0.33	0.24	0.28
caída	0.25	0.35	0.29
tackle	0.30	0.27	0.29
no_choque	0.58	0.51	0.59
promedio macro	0.37	0.37	0.36
peso promedio	0.43	0.43	0.43

Tabla 3.13: Desempeño Transformers sin re-dimensionado de keypoints

	precision	**recall**	**f1-score**
colisión	0.32	0.18	0.23
caída	0.27	0.32	0.29
tackle	0.31	0.33	0.32
no_choque	0.57	0.62	0.59
promedio macro	0.37	0.36	0.36
peso promedio	0.43	0.44	0.43

Tabla 3.14: Desempeño Transformers codificación posicional sin re-dimensionado de keypoints

adas, donde el rendimiento siempre fue oscilante, en los 2 casos anteriores el mejoramiento del puntaje de validación siempre fue constante. A veces mejoraba unas pocas centésimas, pero siempre fue mejor que el ciclo anterior, por lo que se queda con la sensación de que quizás, si se dedicasen el triple o aún más ciclos de los que se dieron en este trabajo, los modelos pudiesen tener un mejor desempeño. No se pudo realizar este aumento en los ciclos debido a limitaciones de tiempo de uso en el entorno en que fue ejecutado.

3.8.7 Encontrando fotogramas sospechosos

El fin de crear estas arquitecturas, era reducir el número de falsos positivos que se obtienen con la rutina que se describe a continuación, en la cual se extraían fotogramas sospechosos de choque basándose en situaciones que suceden con el modelo de seguimiento. Las situaciones son:

- Cuando habiendo 2 instancias cercanas, esta luego es confundida como 1 única y posteriormente se pierde o las 2 anteriores en un lapso de fotograma considerable (para efectos prácticos, 2 o 3 fotogramas)
- Cuando 2 cajas se solapan parcialmente, teniendo estas un tamaño similar
- Cuando una caja se solapa con otra, habiendo una diferencia de tamaño máximo de un tercio (generalmente esto sucede cuando hay un tackleo debido a la inclinación)
- Cuando una caja se encuentra completamente dentro de otra
- Cuando una caja se encuentra dentro de otras 2 diferentes (indicativo de posible caída)

Para ilustrar lo caótico del proceso descrito, tan solo con la primera situación en el clip de Try, donde hay 2 colisiones, se encuentran 600 fotograma posibles de choques. Lo sorprendente es, a pesar de los malos puntajes de los modelos, se logran reducir los falsos positivos de los posibles choques, teniendo además mucho más ordenados para su visualización. Por ejemplo, para el clip

de Half Start donde la rutina encuentra 1296 fotogramas sospechosos (algunos de ellos duplicados por encontrar diversas pérdidas o solapamientos en un mismo fotograma), el modelo transformers encuentra 50 colisiones, 160 caídas y 78 tackles, mientras que el CNN con aumento de dimensión encontró 28 colisiones, 71 caídas y 82 tackles. Todos ellos contienen todavía falsos positivos, pero logran contener los 3 eventos de interés descritos en un espacio mucho más reducido de muestra.

Además de esta rutina, se considera para la extracción de fotogramas los valores instantáneos de ambas aceleraciones por jugador, ya que indican cambios por fotograma de velocidad. Este valor fue definido como C, y es simplemente la raíz cuadrada de la suma de las aceleraciones del jugador al cuadrado:

$$C = \sqrt{a_x^2 + a_y^2}$$

Debido a que se considera el movimiento de la cámara para el seguimiento, además de definirse un valor absoluto mínimo de aceleración para sospechar de algún evento (en este caso, $C > 40$), se define un máximo, el cual es 110 puesto a que cuando se sobrepasa este valor y hay jugadores corriendo, es un indicativo de que van en el mismo sentido de giro de la cámara. Con esto, para el clip de Try se encuentran 300 momentos sospechosos. Es importante mencionar que se hablan de momentos, ya que como son variaciones de cada jugador, puede haber varias sospechas de algún evento en un mismo fotograma.

Capítulo 4

Resultados

A continuación se muestran los resultados de usar todo lo descrito previamente en las grabaciones

4.1 Pruebas Modelos Detectron2

Se obviará en primera instancia las detecciones, ya que su desempeño se verá plasmado en el seguimiento.

Partiendo por la segmentación, debido al solapamiento era de esperarse que esta no fuese perfecta y en los modelos que tuvieran peor desempeño se observase una intermitencia constante entre instancias. Esta intermitencia es constante cuando hay situaciones de acoplamiento como en el scrum, sean cercanos o lejanos estos momentos. Aun así para los 3 mejores modelos se logran resultados aceptables si de limpiar la grabación se trata. Todo lo comentado se aprecia en las figuras 4.1, 4.2, 4.3 y 4.4, donde se empleó mask_rcnn_R_101_DC5_3x para la segmentación.

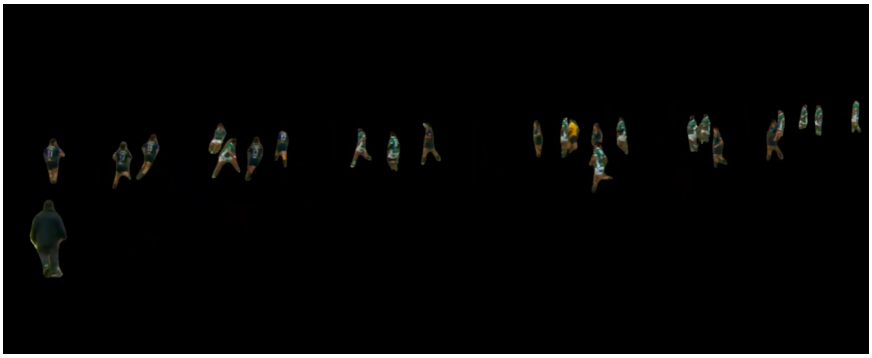


Figura 4.1: Segmentación de instancias en un fotograma de Try

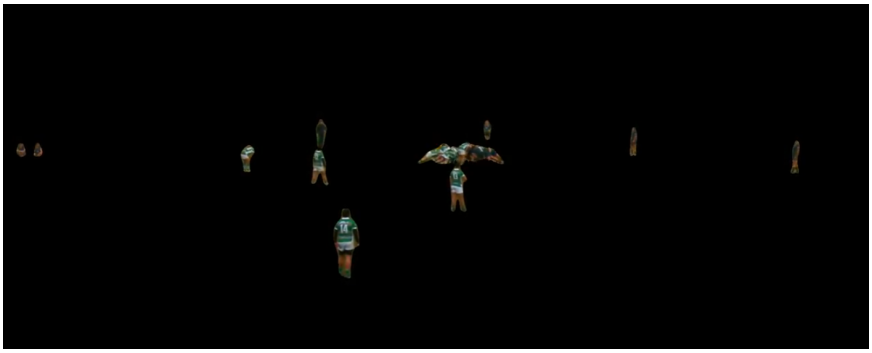


Figura 4.2: Segmentación de instancias en un fotograma de Scrum

Figura 4.3: Segmentación de instancias en un fotograma de Lineout

Figura 4.4: Segmentación de instancias en Lineout con jugadores acoplados

Por otro lado en los keypoints para el mejor modelo, los resultados son excepcionales, puesto que las pérdidas en instancias cercanas a la cámara son bajas (1 o 2 para 30 jugadores), aún en casos de amontonamiento de jugadores. Esta es la razón de por que posteriormente para el análisis de los choques se hace un zoom a los fotogramas sospechosos. En las siguientes 4 imágenes se

aprecia cómo es la representación de los keypoints, Cabe notar como en acoplamientos lejanos a veces hay pérdidas.

Figura 4.5: Keypoints en un fotograma de Try

Figura 4.6: Keypoints en otro fotograma de Try

Figura 4.7: Keypoints en un fotograma de Lineout

Figura 4.8: Keypoints en otro fotograma de Lineout

4.2 Trayectorias, Velocidades, y Aceleración

Ahora, para el trazado de trayectoria como se mencionó previamente, es una opción que viene ya diseñada por BoxMot y aunque logre el trazado, este no se adapta al área de los jugadores, lo que para el contexto de solapamiento y acoplamiento de jugadores al que estamos inmersos, tiende a ser muy caótica su vista y puede llegar a ser algo confuso para el usuario. Este trazado se aprecia en la figura 4.9

Figura 4.9: Arquitectura del modelo Transformer

A continuación se realiza la representación de las velocidades y aceleraciones calculadas en los clips de interés, denotadas por las letras V y A respectivamente. De la misma manera que el movimiento de cámara, las velocidades obedecerán a su signo según se mueven en el eje de coordenadas, es decir, v_x será negativa cuando se mueva el jugador a la izquierda, positiva cuando se mueva a la derecha y v_y será negativa cuando se mueva hacia abajo y positiva hacia arriba. Por otra parte, el signo en las aceleraciones obedecerá al cambio de sentido que tenga el movimiento: Un a_x negativo indica una desaceleración si el jugador se estaba moviendo hacia la derecha, o acelerando hacia la izquierda si su v_x es negativa y viceversa. Por otro lado, a_y sugiere una desaceleración si se estaba moviendo hacia abajo con v_y positiva, o acelerando si en caso contrario dicha velocidad es negativa.

Partiendo con el análisis en scrum, se ve claramente cómo la inestabilidad de la bounding box influye en la obtención del área, partiendo en la figura 4.10 donde se ve como a pesar de estar varios jugadores quietos al igual que la cámara, se indican velocidades y aceleraciones superiores a 0 en su magnitud absoluta, al igual que cuando la cámara induce movimiento estos valores aumentan más drásticamente (ilustrado en la figura 4.11)

Figura 4.10: Velocidades y Aceleración en Scrum cámara fija

Figura 4.11: Velocidades y Aceleración en Scrum cámara en movimiento

En el Lineout (figura 4.12), la variación de velocidad tiene más sentido en los jugadores que se encuentran disputando el balón, pero también se aprecia leves velocidades en los jugadores quietos (aunque con valores menos elevados que en el caso del Scrum).

Figura 4.12: Velocidades y Aceleración en Lineout

Pero sin dudas cuando la pelota se encuentra en movimiento es cuando las estimaciones de velocidad y aceleración tienen valores más consistentes, precisamente en los momentos en que nos interesa más ver sus variaciones (aunque no sean perfectas, como se ve en la figura 4.13). Cabe mencionar, que para el Half start las mediciones también tienen mucho sentido, principalmente a que el movimiento de la cámara queda más denotado al estar la pelota mucho más en movimiento, como se observa en la figura 4.14

Figura 4.13: Velocidades y Aceleración en Try movimiento normal

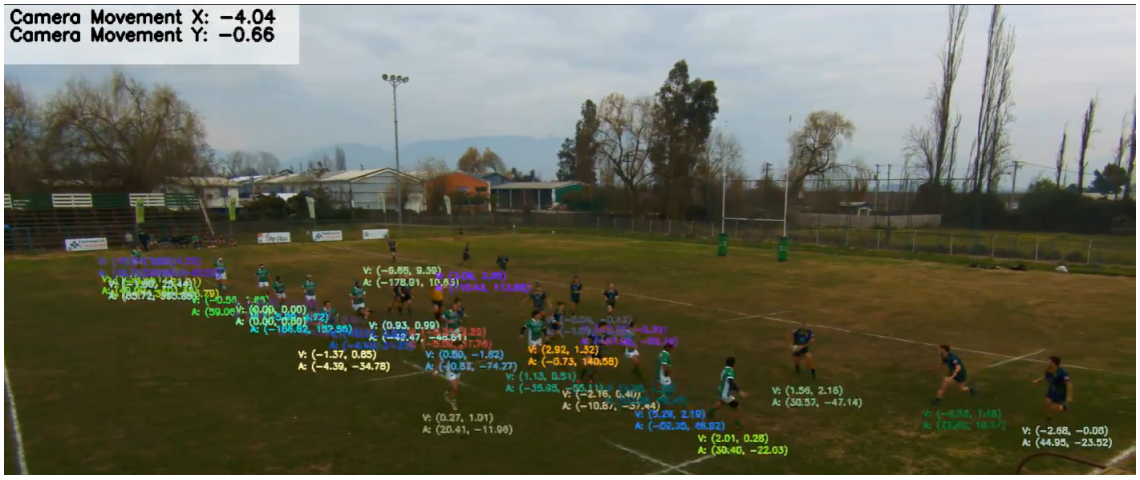
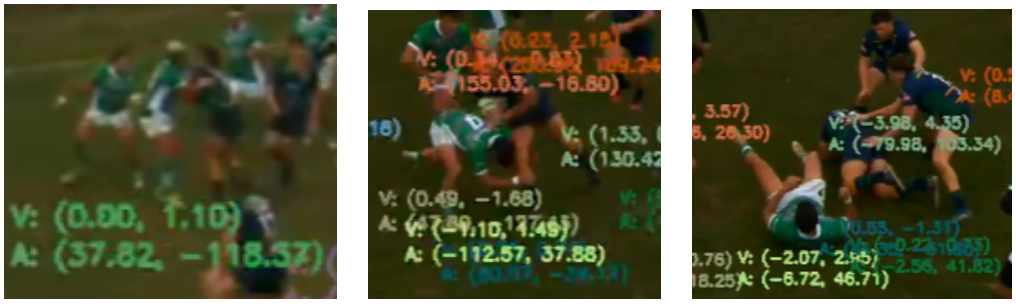


Figura 4.14: Velocidades y Aceleración en Half Start movimiento normal

Es de suma importancia mencionar lo identificado en los últimos 2 clips mencionados, ya que se logra el objetivo de encontrar variaciones de velocidad y aceleración lo suficientemente repentinas para inducir que a ocurrido algún tipo de evento, como se aprecia en los 3 casos de la figura 4.15. Esta variación se toma no solo por los valores inmediatos, sino por los previamente observados (en el inmediato de la medición, 1 fotograma; en el observado en primera instancia, 3 o 5 fotogramas seguidos).



(colisión) (tackle) (caída)

Figura 4.15: Velocidad y Aceleración inmediata al momento de ocurrir los 3 eventos de interés a encontrar

4.3 Detección de Choques

En lo que respecta la detección de los posibles fotogramas con choques, su visualización se hizo por matplotlib. Cabe mencionar que el número que aparece en la parte superior de la imagen no representa el número del fotograma en el video (ya que varios se repiten pero en diferentes acercamientos), sino que es la posición en la lista generada por la rutina.

4.3.1 Keypoints con imágenes re-dimensionadas

Partiendo por el tackle, si bien encuentran los momentos en que suceden estos (o similares, debido a que hay colisiones que pueden identificarse como tackles por la posición de los brazos como aparece en la figura 4.16), también se encontraron en esta lista algunas caídas, como el de la figura 4.17.

Figura 4.16: Detecciones de tackle

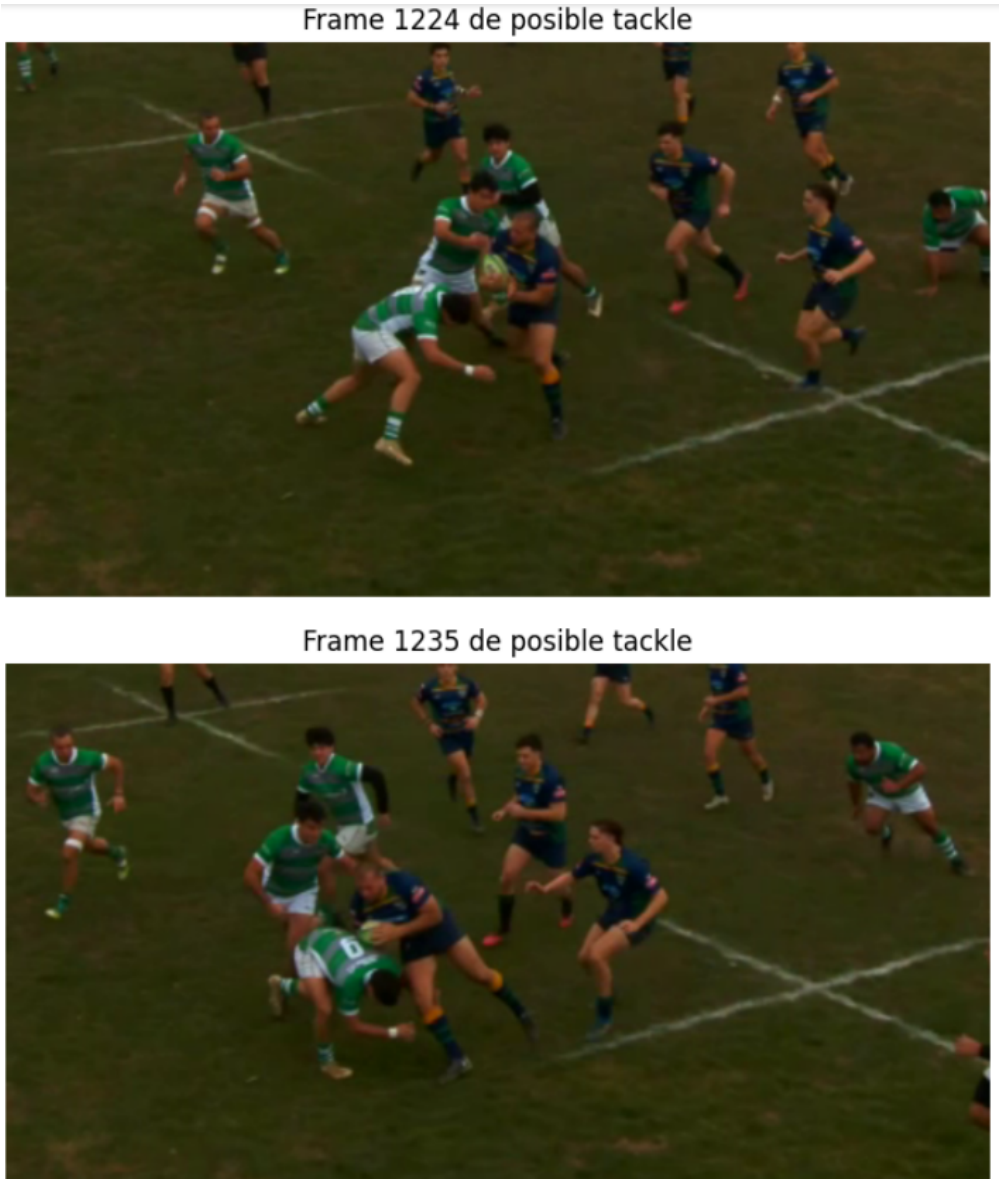


Figura 4.17: Detecciones de tackle

Siguiendo con las colisiones, a simple vista en Half Start no hay presencia de estas como en Try, por lo que era de esperarse que clasificase el modelo algunas de las otras 2 clases como colisión, como se aprecia en las figuras 4.18 y 4.19.

Figura 4.18: Detección de tackle y caída como colisión

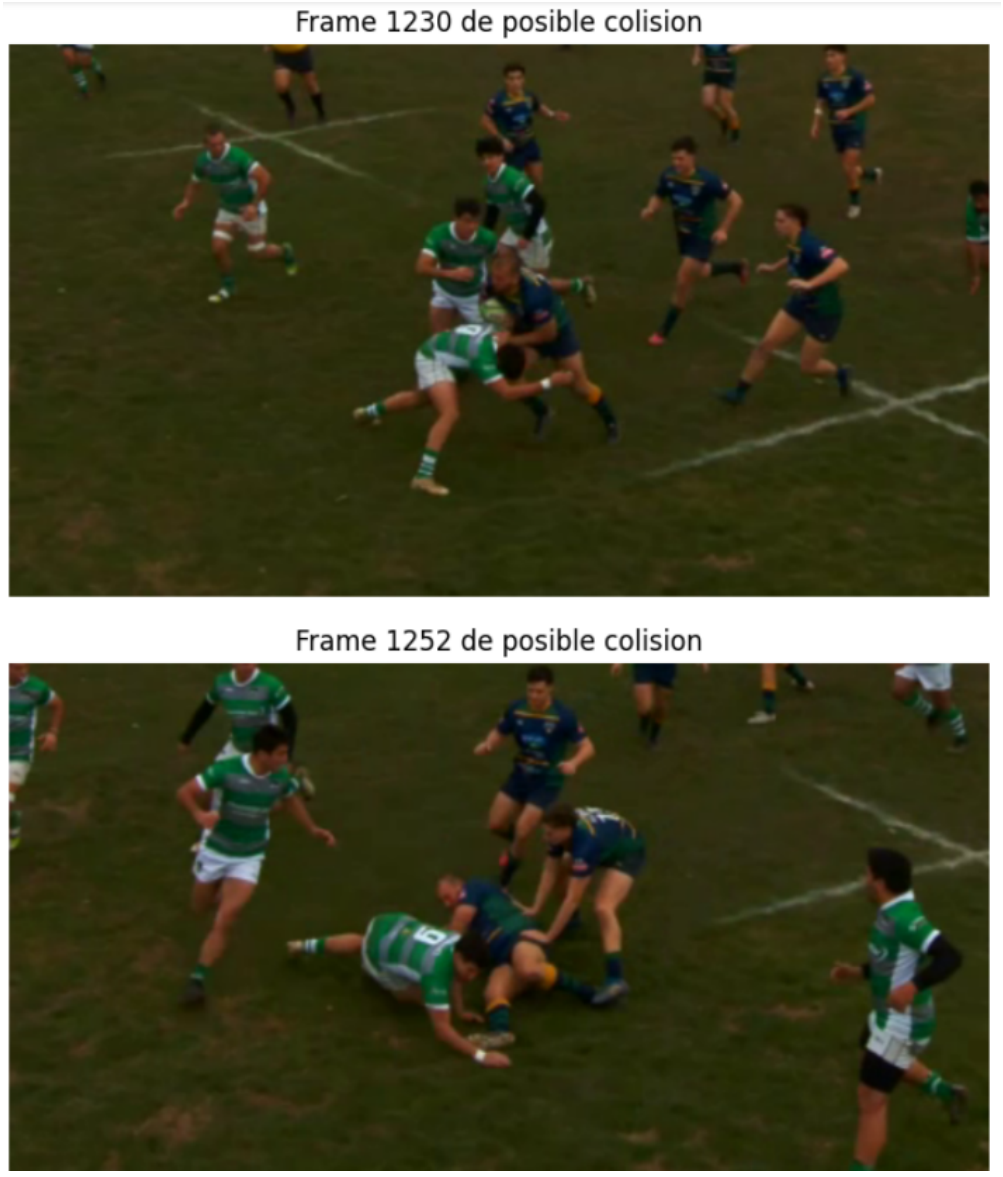


Figura 4.19: Detección de tackle y caída cercanas como colisión

Lo más curioso viene a suceder con las caídas, puesto que si bien son contenidas, estas no entran por el criterio de posible choque, ya que el centro del zoom ocurre en partes cercanas de la caída. Lo otro curioso es que, varias veces se confunde la caída por la posición de los brazos (si están levantados o como si tratasen de afirmarse), como se aprecia en las figuras 4.20 y 4.21.

Figura 4.20: Detecciones de caídas

Frame 604 de posible caida

Figura 4.21: Detecciones de caídas por sesgo

De lo observado en las clasificaciones para Transformers con las imágenes re dimensionadas, se puede confirmar de que hay posibles sesgos que suceden por las posiciones de los brazos principalmente, y ademas viene a cuestionar la efectividad para encontrar posibles choques usando la rutina de solapamiento, sobretodo por la ultima figura mencionada donde fue por mera casualidad que se obtuvieron jugadores cayéndose, por lo que se usan las variaciones de aceleración en los 2 análisis de choques posteriores.

4.3.2 MLP con máscara de segmentación

Aún cuando se obtuvieron buenos puntajes en el entrenamiento, el desempeño deja mucho que desear, ya que además de solo clasificar 1 imagen por lista de posibles fotogramas pasados, esta siempre tenía un sesgo, puesto a que como se aprecia en la figura 4.22, bastaba con que hubiese un

cruce de brazos de un jugador para clasificar una colisión, o una levantada de brazos para clasificar una caída.

Figura 4.22: MLP teniendo sesgo para clasificar colisión y caída

4.3.3 Keypoints sin re-dimensionar imágenes

En este caso, si bien se sigue esperando categorizar todos los eventos que tienen algo que ver con choque aún si no es acertado, aquí se producen más errores en dicha categorización. Por ejemplo en el caso del transformer con codificación posicional, para las poses del fotograma de la figura 4.23

tuvo un sesgo por los ángulos, y confundió una clara colisión con una caída.

Figura 4.23: Transformer 2 clasificando una colisión como caida

No obstante por este mismo sesgo, fue capaz de clasificar de forma correcta 2 caídas (o momentos en que sucedió, ya que el jugador se encontraba parándose) de manera correcta, aún estando el jugador levemente solapado, como se aprecia en las figura 4.24.

Figura 4.24: Transformer 2 clasificando una caída correctamente

En cuanto el primer transformer (sin codificación posicional), este sufre más del problema de no clasificar de forma correcta determinados eventos. En la figura 4.25 se muestra cómo clasificó una supuesta caída únicamente por la pose del jugador, como si se estuviese cayendo levemente de espalda cuando en realidad, el ángulo de su torso no se condice con el de una persona cayéndose, ya que se encuentra más recto que inclinado.

Figura 4.25: Transformer 1 falso positivo de caida

Del mismo modo, la misma caída clasificada por el Transformer 2, este lo identificó primero como colisión, y posteriormente como tackle, como se aprecia en la figura 4.26.

Figura 4.26: Transformer 1 clasificando una caída como colisión y tackle

Una vez analizados los 3 casos, se observa que la mejor forma de realizar la identificación de caídas, choques, y tackles, es utilizando las variaciones de aceleración para identificar posibles eventos, para luego clasificar estos fotogramas mediante el transformer con codificación posicional entrenado sin re-dimensionar las imágenes del dataset

Luego, la generación del clip corto con los keypoints del fotograma seleccionado resulta bastante simple y su representación como era de esperarse es de muy buena calidad gracias al zoom, como se demuestra en la figura 4.27

Figura 4.27: Fotograma del clip de 2 segundos generado

De este video, se obtiene que las partes involucradas en la colisión son el pecho y los hombros del jugador de los All Brads involucrado, esto debido a la cercanía de los keypoints del rival con el que colisiona.

Finalmente, para comparar cuantitativamente del rendimiento de los 3 modelos con mejor rendimiento, se crea una tabla con el detalle del número de las clasificaciones en los modelos utilizados, según cada criterio para identificar posibles choques, y si se encuentran cubiertos todos los eventos ocurridos en el clip. Cuando en este último apartado se mencione "Si por casualidad", se hace referencia a que algunos de los eventos contenidos no forman parte del centro del fotograma, indicando que la combinación de modelo y criterio no es acertada en su totalidad, aún cuando el fotograma posee un evento. En la tabla 4.1 se encuentran las clasificaciones con el criterio de variación de velocidad.

En la tabla 4.2 se encuentran las clasificaciones usando el criterio de solapamiento. Este criterio tuvo un peor rendimiento si se considera la cobertura de los eventos. Es importante mencionar que solo 1 pudo contener todos los eventos, por lo que se puede concluir que también los modelos utilizados en este contexto no son adecuados, y también que la CNN en solapamiento resultó ser la combinación con peor rendimiento.

También se encuentra que de los 3 modelos, el que mejor rendimiento obtuvo para los 2 criterios fue el Transformer Normal, a diferencia de lo mencionado previamente sobre el Transformer con Codificación Posicional.

Clip	**Modelo**	**colisión**	**caída**	**tackle**	**no_choque**	**¿Contuvo todos los eventos ocurridos?**
Try	Trans. Codificación Posicional	10	69	17	205	Si
	Trans. Normal	5	31	31	234	Si (por casualidad)
	CNN Aumento Dimensional	5	23	19	254	Si (por casualidad)
Half_start	Trans. Codificación Posicional	8	57	40	446	No
	Trans. Normal	5	60	48	438	Si
	CNN Aumento Dimensional	3	9	16	523	No

Tabla 4.1: Clasificación de posibles choques extraídos usando criterio de aceleración

Clip	**Modelo**	**colisión**	**caída**	**tackle**	**no_choque**	**¿Contuvo todos los eventos ocurridos?**
Try	Trans. Codificación Posicional	29	95	54	461	No
	Trans. Normal	21	97	50	471	No
	CNN Aumento Dimensional	93	57	61	428	No
Half_start	Trans. Codificación Posicional	59	169	80	988	No
	Trans. Normal	50	160	78	1008	Si
	CNN Aumento Dimensional	28	71	82	1115	No

Tabla 4.2: Clasificación de posibles choques extraídos usando criterio de solapamiento

Capítulo 5

Conclusiones y Trabajos Futuros

5.1 Conclusiones

A grandes rasgos la hipótesis definida se cumple, puesto que sin usar un datataset para hacer las identificaciones de los jugadores y sin crear un dataset explícito de los partidos de los All Brads, se logra encontrar momentos de colisión, caída y tackle que tomaría esfuerzo por parte de los usuarios si se quisiesen identificar. Eso sí, el desempeño no fue el esperado, ya que siguen habiendo falsos positivos generados por cierto sesgo que tienen los modelos seleccionados. Respecto a cada modelo usado y creado en el desarrollo de esta memoria:

- En identificación, si bien el puntaje de cada modelo siempre fue superior a 83%, aún es posible lograr mayor exactitud, puesto que Detectron2 da la posibilidad de agregar un dataset personalizado que sea subclase de personas.
- En seguimiento de los jugadores, se demostró que ByteTrack no siempre es una buena opción y que para casos de solapamiento sea necesario potenciar los modelos pre entrenados para tener una mejor re identificación de instancias cuando se pierdan (lo que escapa al enfoque de esta memoria).
- En detección de choques, se demuestra que la creación de un dataset para este propósito no es una tarea trivial y de que lograr una generalización de esto tampoco lo es, y que aún cuando las imágenes son de alta calidad y detalle, en las arquitecturas simples (al menos) la clasificación sufre de errores de clasificación de colisión, caída o choque, pero funciona bien para identificar cuando no hay choque.

5.2 Trabajos Futuros

- Crear un dataset para tener la subclase All Brads y Rivales en Detectron2.
- Achicar la línea generadora de la trayectoria de BoxMot para que sea en concordancia con el área de la caja
- Considerar las mediciones de los sensores para realizar el cálculo de la velocidad en conjunto con el seguimiento de los jugadores.
- Agregar o Modificar el Dataset de las clases de choques, considerar agregar los keypoints generados, las imágenes sin limpiar e imágenes de grabaciones de partidos.
- Diseñar o combinar diversas arquitecturas para la identificación de los choques, aumentándoles su complejidad de ser necesario (a pesar de que esto vaya en contra de lo indicado por la literatura), hasta alcanzar un mejor porcentaje de exactitud en el entrenamiento.

Capítulo 6

Referencias

[1] J. van Dis B.Sc. , dr. ir. R.W. Poppe, dr. F.P.M. Dignum, "Extracting ground plane location data of rugby 7s players, from a zooming and rotating camera" Master of Science Thesis, Faculty of Science, Universiteit Utrecht (2016)

[2] Federica Baldi, Dr. Luca Marchesotti, Fabrizio Falchi, Claudio Gennaro, "Action Detection in rugby video sequences in both professional and amateur settings" Master of Science in AI and Data Engineering Thesis, School of Engineering, Department of Information Engineering, Università Di Pisa

[3] Manju Rana, Vikas Mittal, "Wearable Sensors for Real-Time Kinematics Analysis in Sports: A Review", IEEE Sensors Journal, Vol. 21, no. 2 (2021)

[4] Mareike Roell 1, Hubert Mahler, Johannes Lienhard, Dominic Gehring, Albert Gollhofer, Kai Roecker, "Validation of Wearable Sensors during Team Sport-Specific Movements in Indoor Environments", Article, Institute of Sports and Sports Science (IfSS), Albert-Ludwigs-University Freiburg, Germany (2019)

[5] Ryan M. Chambers, Tim J. Gabbett3, Michael H. Cole, "Validity of a Microsensor-based Algorithm for Detecting Scrum Events in Rugby Union", International Journal of Sports Physiology and Performance (2018)

[6] Jagreet Kaur, Suryakant, Akash Pandey, Arshika, "Sports Analytics using computer vision Performance analytics using Computer vision and analysis techniques"

[7] Graham Thomas, Thomas B. Moeslund, Adrian Hilton "Computer Vision in Sports", book (2014)

[8] Banoth Thulasya Naik, Mohammad Farukh Hashmi, Neeraj Dhanraj Bokde “A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions”, review (2022)

[9] Dongdong Zhu, Honglei Zhang,Yulong Sun, Haijie Qi, “Injury Risk Prediction of Aerobics Athletes Based on Big Data and Computer Vision”, Institute of Physical Education, Dezhou University, Shandong, China (2021)

[10] Zubair Martin, Sharief Hendricks, Amir Patel, “Automated Tackle Injury Risk Assessment in Contact-Based Sports - A Rugby Union Example”, African Robotics Unit, University of Cape Town (2021)

[11] Qin Jiang, Ruanming Huang, Yichao Huang, Shujuan Chen, Yuqing He, Li Lan, Cong Liu, “Application of BP Neural Network Based on Genetic Algorithm Optimization in Evaluation of Power Grid Investment Risk”, College of Electrical and Information Engineering, Hunan University, Changsha, China (2019)

[12] Zhiqiang Zhang, Li Gao, Yangyang Xiang, “Application of Optimized BP Neural Network Based on Genetic Algorithm in Rugby Tackle Action Recognition”, Beijing University of Posts and Telecommunications Beijing, China (2020)

[13] Kristina Host, Marina Ivašic-Kos, “An overview of Human Action Recognition in sports based on Computer Vision”, Faculty of Informatics and Digital Technologies, University of Rijeka, Croatia (2022)

[14] Naoki Nonaka, Ryo Fukihira, Monami Nishio, Hidetaka Murakami, Takuya Tajima, Mutsuo Yamada, Akira Maeda, Jun Seita, “End-to-End High-Risk Tackle Detection System for Rugby”, Faculty of Health and Sport Sciences, Ryutsu Keizai University

[15] Rina Ichige, Yoshimitsu Aoki, “Action Recognition in Sports Video Considering Location Information”, Keio University, Japan (2020)

[16] Yasushi Akiyama, Rodolfo Garcia, Tyson Hynes, “Video Scene Extraction Tool for Soccer Goalkeeper Performance Data Analysis”, IUI Workshops’19, Los Angeles(2019)

[17] Rahmad, N.A.; As'ari, M.A.; Ghazali, N.F.; Shahar, N.; Sufri, N.A.J. "A survey of video based action recognition in sports". Indones. J. Electr. Eng. Comput. Sci. (2018)

[18] Tan, S.; Yang, R. "Learning similarity: Feature-aligning network for few-shot action recognition". In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary (2019)

[19] Ullah, A.; Ahmad, J.; Muhammad, K.; Sajjad, M.; Baik, S.W. "Action recognition in video sequences using deep bi-directional LSTM with CNN features". IEEE Access (2017)

[20] Ke Sun, Bin Xiao, Dong Liu, Jingdong Wang, "Deep High-Resolution Representation Learning for Human Pose Estimation", University of Science and Technology of China, Microsoft Research Asia, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[21] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", Microsoft Research (2016)

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, "Focal Loss for Dense Object Detection", Facebook AI Research (FAIR) (2018)

[26] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, "Mask R-CNN", Facebook AI Research (FAIR) (2018)

[27] Siyuan Li, Li Sun, Qingli Li, "CLIP-ReID: Exploiting Vision-Language Model for Image Re-Identification without Concrete Text Labels", Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, Shanghai, China (2023)

[28] Fabian Herzog, Xunbo Ji, Torben Teepe, Stefan Hörmann, Johannes Gilg, Gerhard Rigoll, "Lightweight Multi-Branch Network for Person Re-Identification", IEEE International Conference on Image Processing (ICIP) (2021)

[29] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, Tao Xiang, "Omni-Scale Feature Learning for Person Re-Identification", University of Surrey, Queen Mary University of London, Samsung AI Center, Cambridge (2019)

[30] Nir Aharon, Roy Orfaig, Ben-Zion Bobrovsky, "BoT-SORT: Robust Associations Multi-Pedestrian

Tracking", School of Electrical Engineering, Tel-Aviv University (2022)

[31] Yunhao Du, Zhicheng Zhao, Yang Song Yanyun Zhao, Fei Su, Tao Gong, Hongying Meng, "StrongSORT: Make DeepSORT Great Again" (2023)

[32] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, Xinggang Wang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box", Huazhong University of Science and Technology, The University of Hong Kong, ByteDance Inc. (2022)

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need", Google Research, University of Toronto (2017)

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", Microsoft Research (2015)

[35] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, Lu Yuan, "TinyViT: Fast Pretraining Distillation for Small Vision Transformers", Sun Yat-sen University, University of Wisconsin-Madison, Microsoft Research, Microsoft Cloud+AI (2022)