ANÁLISIS DEPORTIVO A TRAVÉS DE LA VISIÓN POR COMPUTADOR

DANIEL LEÓN PINEDA
LAURA NATALIA MOTTA CADENA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2021

ANÁLISIS DEPORTIVO A TRAVÉS DE LA VISIÓN POR COMPUTADOR

DANIEL LEÓN PINEDA
LAURA NATALIA MOTTA CADENA

Trabajo de grado presentado como requisito para obtener el título de Ingeniero Electrónico

Director:

Julián Adolfo Ramírez Gutiérrez

MSc en Teoría de la señal y Comunicaciones

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA, COLOMBIA
2021

Nota De Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Neiva, Abril 19 de 2021

Agradezco primeramente a Dios por permitirme alcanzar este logro, a mi familia por su apoyo y confianza, principalmente a mi madre, aquella que siempre estuvo desde que inicié hasta que culminé, la que me motivó y me dio fuerzas. Mi padre que en todo momento me recalco la importancia del estudio. A mi hermano mayor, aquel que en esos momentos difíciles fue clave para decirme que no me rindiera e impulsándome a cumplir mis anhelos. A mi hermanito aquel que siempre me estaba preguntando "que era eso", "para que sirve", el que en su inocencia me daba su apoyo. A mi compañera Laura Natalia aquella que, pese a todas esas dificultades presentes, siempre estuvo en alto, persistiendo y logrando terminar este gran trabajo. Quiero agradecer a mis amigos y demás familiares que siempre estuvieron de una u otra forma brindándome su compañía, apoyo, motivación y cariño. Y no menos importante a mis profesores de carrera que fueron pieza clave para este gran momento, muchas gracias a cada una de las personas que estuvieron ahí.

Daniel León Pineda

Agradezco en primer lugar a Dios por haberme concedido llegar a este punto de mi vida, a mis padres, Luis Antonio y Sandra Milena, a quienes debo todo lo que soy y tengo, por su amor, compañía, dedicación y constante trabajo a lo largo de todos estos años, por los valores inculcados y por las enseñanzas impartidas, este trabajo está dedicado a ellos. A mi familia, por su apoyo y consejo. A mis amigos, por sus palabras de aliento y motivación en los días difíciles. A mi compañero de Tesis, por su trabajo y paciencia. A mis profesores, por las lecciones dadas y el tiempo invertido. A todas y cada una de las personas que me han rodeado los últimos años y que de alguna u otra forma me han ayudado a ser mejor mujer y persona. Gracias a todos ellos, porque me han permitido alcanzar esta meta, una de tantas por venir.

Laura Natalia Motta Cadena

AGRADECIMIENTOS

Un agradecimiento especial a nuestro asesor de Tesis, el ingeniero Julián Adolfo Ramírez, por su ayuda y mentoría durante el desarrollo de este proyecto de grado y especialmente, por su acompañamiento durante todo nuestro ciclo formativo. A Diana María García Cardona y en general a los miembros del Grupo de Investigación en Fisiología de la Actividad Física y la Salud (GIFAS) de la Universidad del Quindío por su constante colaboración y apoyo técnico durante las grabaciones de los test deportivos. Al ingeniero Carlos Pardo por sus asesorías referentes al tema de nuestro proyecto. A nuestros compañeros de estudio por su compañía y ayuda a lo largo de la carrera. Y finalmente, pero no menos importante, a cada uno de los miembros, ingenieros y administrativos, del programa de Ingeniería Electrónica, por sus enseñanzas y consejos.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

La visión artificial o visión por computador es una rama de la inteligencia artificial que se ocupa del estudio del procesamiento, análisis e interpretación de imágenes digitales de forma automática[1]. En los últimos años, el auge de esta disciplina ha sido tal, que ha dado lugar al surgimiento de un gran número de aplicaciones en torno a este campo de estudio, una de ellas es la estimación de la pose humana[2], la cual a su vez consiste, en la detección e identificación de las principales articulaciones del cuerpo humano de una o varias personas. El estudio del análisis del movimiento humano y la estimación de la pose humana ha estado bastante ligado a las investigaciones en las áreas de la biomecánica y de las ciencias del deporte, en esta última, los avances han permitido obtener sistemas capaces de automatizar y mejorar los análisis deportivos, al proporcionar información que los métodos de análisis tradicionales no pueden aportar.

Con el fin de evaluar la contribución de esta aplicación de la visión artificial a la automatización de ciertas prácticas deportivas, y partiendo de una actividad en específico, como lo es en este caso el Test de Course – Navette, se diseñó un programa de automatización capaz de obtener las evaluaciones deportivas, producto del análisis manual de la ejecución del Test, en base a un video pre – grabado del mismo. El programa creado se realizó en base al lenguaje de programación de Python y cuenta con una estructura de tipo modular, es decir, se encuentra divido en tres etapas de desarrollo.

La primera etapa del programa consiste en la detección de los jugadores presentes en el video a evaluar, a través de algoritmos de estimación de la pose humana e identificación de objetos; debido a que la aplicación requiere tener conocimiento de la ubicación de los deportistas a lo largo de todo el video, se hizo necesario implementar una etapa de seguimiento, complementaria a la primera, que fuera capaz de identificar a los jugadores y rastrearlos durante toda la ejecución del test. Finalmente, para obtener las evaluaciones correspondientes a cada uno de los deportistas, se realizó una etapa de transformación de perspectiva y mapeo de coordenadas, la cual permitió analizar de forma más simple y clara el recorrido hecho por los jugadores.

Palabras Clave: Análisis Deportivo, Estimación de la Pose Humana, Seguimiento de Personas, Visión por Computador, Transformación de Perspectiva.

[1] Visión por computadora - Libro online de IAAR.”. {En línea}. {18 marzo de 2021} disponible en: (https://iaarbook.github.io/vision-por-computadora/).

[2] “A 2019 guide to Human Pose Estimation with Deep Learning.”. {En línea}. {18 marzo de 2021} disponible en: (https://nanonets.com/blog/human-pose-estimation-2d-guide/)

ABSTRACT

Computer vision is a branch of artificial intelligence that deals with the study of automatic processing, analysis, and interpretation of digital images. In recent years, the rise of this discipline has allowed the emergence of many applications around this field of study, as an example, human pose estimation, which consists in the detection and identification of the human body's main joints for one or more people. The field of study for human movement analysis and human pose estimation has been closely linked to research activities in biomechanics and sports science's areas, advances in the last one have made possible to obtain systems capable of automating and improving sports analysis by providing information that traditional analysis methods cannot.

To evaluate the contribution of this computer vision application to the automation of certain sports practices and starting from a specific activity, such as the Course-Navette Test, a capable automation algorithm was designed to obtain the sports evaluations from the Test, the above based on a pre-recorded video of the test. The program created is based on Python programming language and has a modular type of structure, which means that it is divided into three development stages.

Namely the first stage of the program consisted in the detection of the players present in the video to be evaluated through human pose estimation algorithms; since the application requires to know the location of the athletes throughout the video, it was necessary to implement a follow-up stage, complementary to the initial stage, that could identify the players and tracked them throughout the execution of the test. Finally, to obtain the evaluations corresponding to each one of the athletes, a perspective transformation and mapping stage was carried out, which allowed a simple and clear analysis of the path taken by the players.

Keywords: Computer Vision, Human Pose Estimation, People Tracking, Perspective Transformation, Sports Analysis.

1. INTRODUCCIÓN

El análisis deportivo es un concepto que ha existido por muchos años y que, tradicionalmente, puede ser definido como el uso de datos históricos en conjunto con la aplicación de modelos matemáticos y estadísticos avanzados, orientados a la obtención de métricas que permitan medir el rendimiento, tomar decisiones y hacer predicciones sobre los resultados de un deporte en cuestión, con el fin de obtener cierto tipo de ventaja táctica sobre los otros equipos o jugadores[3]. En las últimas décadas, esta área de estudio ha tomado un nuevo impulso y ha ampliado de forma significativa sus alcances, gracias a los aportes que ha obtenido del desarrollo y continuo avance que ha tenido la inteligencia artificial y sus distintas ramas; es común ver entonces, que los principales clubs deportivos en la actualidad inviertan significativas cifras en modelos y algoritmos que les ayuden a identificar las principales debilidades de sus equipos, así como de sus rivales[4] [5] [6] [7] [8].

La repercusión que ha tenido esta nueva perspectiva del análisis deportivo ha permitido el desarrollo de diversos estudios enfocados a diferentes prácticas deportivas y a variadas formas de análisis; los principales deportes que han sido foco de investigación en esta área han sido el Futbol, el Baloncesto y el Tenis, alrededor de ellos han surgido un gran número de trabajos que han estado enfocados principalmente en dos ramas, complementarias la una a la otra, la primera se relaciona con la creación de algoritmos para la detección y el rastreo de uno o varios jugadores, tal y como se presenta en[9] para el Baloncesto, en[10] para el Tenis y en[11] para deportes realizados en cancha en general; por otro lado, la segunda rama se ha orientado al área de las predicciones deportivas, las estimaciones del rendimiento y los análisis de jugadas, algunos ejemplos de estos

[3] "What is Sports Analytics? Victor Holman Explains - Agile Sports Analytics.". {En línea}. {10 marzo de 2021} disponible en: (https://www.agilesportsanalytics.com/what-is-sports-analytics/)
[4]"Hudl: We Help Teams and Athletes Win.". {En línea}. {18 marzo de 2021} disponible en: (https://www.hudl.com/).
[5] "Kitman Labs - Powering Human Performance.". {En línea}. {18 marzo de 2021} disponible en: (https://www.kitmanlabs.com/).
[6] "SAS: Analytics, Artificial Intelligence and Data Management | SAS.". {En línea}. {18 marzo de 2021} disponible en: (https://www.sas.com/en_ae/home.html).
[7] "The analytics database | Exasol.". {En línea}. {18 marzo de 2021} disponible en: (https://www.exasol.com/).
[8] "SportVU - Stats Perform.". {En línea}. {21 marzo de 2021} disponible en: (https://www.statsperform.com/team-performance/football-performance/optical-tracking/).
[9] E. Cheshire, C. Halasz, and J. K. Perin, "Player Tracking and Analysis of Basketball Plays," (2015).
[10] R. M. Nieto and J. M. M. Sanchez, "An automatic system for sports analytics in multi-camera tennis videos," En: IEEE Int. Conf. Adv. Video Signal Based Surveill., (2013); p. 438–442.
[11] V. Vovk, S. Skuratovskyi, P. Vyplavin, and I. Gorovyi, "Light-Weight Tracker for Sports Applications," Signal Process. Symp. SPSympo, (2019), p. 251–255.

trabajos son los sistemas presentados en[12] [13] [14] para el Futbol y en[15] para escenarios deportivos en general. De igual forma, es posible encontrar enfoques basados en el seguimiento y análisis de relación de los objetos comúnmente utilizados en las prácticas deportivas, tales como las pelotas, ejemplos de esto son los trabajos en[16] [17].

Siguiendo el modelo de los trabajos que se han realizado en la parte de la detección y el seguimiento o rastreo de jugadores, este proyecto de grado explica el funcionamiento de un programa que se encarga de realizar las evaluaciones deportivas del Test de Course – Navette de forma automática, y en base a un video pregrabado de la ejecución de este. Para lograr este objetivo se realizó, en primer lugar, una exploración a la documentación oficial del Test de Course – Navette[18], con el fin de identificar los parámetros a obtener del proceso de evaluación de la ejecución del Test, como resultado de dicha búsqueda se encontró que el test evalúa la potencia aeróbica máxima o el consumo máximo de oxígeno en los jugadores, dicha variable se utiliza como un indicador de la cantidad máxima de oxígeno que un individuo puede absorber del ambiente por unidad de tiempo, durante la realización de una actividad que aumenta de intensidad progresivamente[19]. En base a esta información se definió la metodología a implementar para llevar a cabo el desarrollo del algoritmo que permitiera obtener el parámetro mencionado anteriormente; de acuerdo con los requerimientos de la aplicación, se implementaron tres etapas de desarrollo: Etapa de Detección, Etapa de Seguimiento y Etapa de Evaluación; a través de cada una de estas etapas se evaluaron diferentes alternativas de software y diferentes perspectivas de grabación del escenario deportivo, esto para poder obtener, respectivamente, un programa que arrojara las evaluaciones buscadas, y por otro lado, el mejor arreglo de cámaras que permitiera una captura eficiente de la ejecución del test. Para la comprobación del funcionamiento del programa, se realizó la comparación de los

[12] K. Apostolou and C. Tjortjis, “Sports Analytics algorithms for performance prediction,” 10th Int. Conf. Information, Intell. Syst. Appl. IISA, (2019), p. 1–4.
[13] V. C. Pantzalis and C. Tjortjis, “Sports Analytics for Football League Table and Player Performance Prediction,” (Octu 2020), p. 9.
[14] M. Stein et al., “Revealing the Invisible: Visual Analytics and Explanatory Storytelling for Advanced Team Sport Analysis,” Symp. Big Data Vis. Immersive Anal. BDVA, (2018), pp. 148–156.
[15] L. Sha et al., “Interactive Sports Analytics: An Intelligent Interface for Utilizing Trajectories for Interactive Sports Play Retrieval and Analytics,” ACM Trans. Comput. Interact., vol. 29, (2010), p. 34.
[16] M. Gowda et al., “Bringing IoT to sports analytics,” Proc. 14th USENIX Symp. Networked Syst. Des. Implementation, (2017), p. 499–513.
[17] Y. Xu and Y. Peng, “Real-Time Possessing Relationship Detection for Sports Analytics,” Chinese Control Conf. (Jul 2020), p. 7373–7378.
[18] G. C. García and J. D. Secchi, “Test Course Navette de 20 metros con etapas de un minuto. Una idea original que perdura hace 30 años,” Apunt. Med. l’Esport, vol. 49, no. 183, (2014), p. 93–103.
[19] M. Firinola G, “Pruebas de Campo para la Valoración del Consumo Máximo de Oxígeno, La Velocidad Aeróbica Máxima, y La Resistencia Intermitente,” Rev. electrónica Ciencias Apl. al Deport., vol. 2, no. 5, {En línea}. {2019} disponible en: http://www.romerobrest.edu.ar/ojs/index.php/ReCAD%0APruebas

resultados productos de la evaluación automática frente a los resultados producto de la evaluación manual.

El desarrollo del trabajo se expone en el texto de la siguiente forma, en el capítulo 3 se presenta un marco contextual de las herramientas utilizadas y discutidas durante el desarrollo del programa, seguido a esto, en el capítulo 4, se introduce la metodología de desarrollo utilizada para dar cumplimiento a los objetivos planteados y se explica, mediante diagramas de flujo, la estructura general de la herramienta en cuestión, de igual forma, se exponen los resultados obtenidos durante el desarrollo de cada una de las etapas planteadas para el algoritmo de automatización, y se evalúan las alternativas implementadas para cada caso. En el capítulo 5, se realiza una discusión de los resultados obtenidos, la cual está centrada en las limitaciones del programa, las recomendaciones de uso y la validación de los resultados frente a un evaluador humano. En el capítulo 6, se realizan las conclusiones correspondientes y finalmente, en el capítulo 7, se exploran los posibles trabajos a futuro haciendo uso de la implementación realizada.

2. OBJETIVOS

2.1. OBJETIVO ESPECIFICO

- Evaluar una estrategia de automatización para el test deportivo de Course-Navette haciendo uso de OpenPose como herramienta de visión artificial.

2.2. OBJETIVOS GENERALES

- Identificar los parámetros y variables claves involucrados en la ejecución de la prueba, que permitan realizar de forma sencilla y eficaz la automatización de este.
- Determinar el mejor arreglo de cámaras (posición, altura, tipo y número) que permita obtener una captura eficiente del total de la ejecución del test deportivo.
- Diseñar un algoritmo que permita obtener de forma automática las evaluaciones deportivas producto del desarrollo del test
- Evaluar el funcionamiento del algoritmo de automatización realizado a través de la comparación con los resultados obtenidos de la evaluación deportiva manual

3. MARCO DE TRABAJO

3.1. ANÁLISIS DEPORTIVO.

El análisis deportivo, como se mencionó anteriormente, es un campo que aplica técnicas de análisis de datos al estudio de diversos componentes, escenarios y/o actores de la industria del deporte. En los últimos años, esta disciplina ha ampliado su enfoque y se ha convertido en el principal aliado de los grandes clubs y ligas deportivas; actualmente, una de las principales compañías en ofrecer este tipo de tecnología es SportVU[20], su software se encarga de entregar estadísticas de rendimiento al extraer las coordenadas de los jugadores y el balón, y utilizar técnicas de inteligencia artificial con ellas, grandes equipos como el Juventus o el Bayern Múnich han adoptado esta herramienta para el análisis de sus juegos.

En el escenario global, deportes como el Futbol, el Baloncesto y el Tenis han sido el principal foco de atención de esta área de estudio, alrededor de ellos han surgido un gran número de trabajos, los cuales han sido encaminados por dos enfoques complementarios, el primero centrado en la creación de programas para la detección y seguimiento de uno o varios jugadores[21] [22] [23] y el segundo orientado al área de las predicciones deportivas, las estimaciones del rendimiento y los análisis de jugadas[24] [25] [26] [27]. Sin embargo, con respecto al ámbito local, las prácticas deportivas realizadas en ambientes educativos o los test ejecutados por adultos mayores, han sido, en su gran mayoría, apartados de los alcances de esta disciplina; dentro de dichas prácticas deportivas locales se encuentra el Test de Course – Navette o Test de Leger, comúnmente utilizado por estudiantes o personas del común.

3.1.1 Test de Course – Navette. El Test de Course Navette o también conocido como Test de Leger fue desarrollado por el doctor Luc Leger, profesor de la Universidad de Montreal, en la década de 1980. Conocido principalmente como la carrera de 20 metros, el funcionamiento de este test, tal y como lo describe Curilem y sus compañeros[28], se

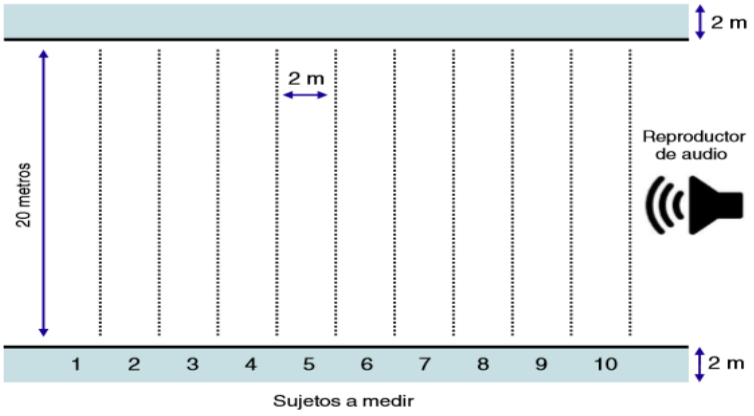
[20] SportVU - Stats Perform. Op.cit.
[21] E. Cheshire, C. Halasz. Op. cit.
[22] R. M. Nieto and J. M. M. Sanchez. Op. cit. p. 438–442.
[23] V. Vovk, S. Skuratovskyi. Op. cit. p. 251–255.
[24] K. Apostolou and C. Tjortjis. Op. cit. p. 1-4.
[25] V. C. Pantzalis. Op. cit. p. 9.
[26] M. Stein et al. Op. cit. p. 148-156.
[27] L. Sha et al. Op. cit. p. 34.
[28] C. Curilem Gatica, A. Almagiá Flores, and Y. F. Tuillang, “Aplicación del Test Course Navette en Escolares,” Rev. Mot. Humana, Edición, vol. 16, (2015), p. 95–99.

basa principalmente en una carrera de ida y vuelta (20 metros en cada sentido) en la que el sujeto se desplaza de un punto a otro del escenario, al ritmo indicado por una señal sonora que aumenta su ocurrencia de forma progresiva.

La velocidad inicial del test es de $8.5Km/h$, a medida que transcurre el tiempo, esta aumenta $0.5Km/h$ cada minuto hasta alcanzar los $15.5Km/h$, que corresponde a la velocidad de la etapa número quince. Existen dos versiones del test, una de 15 etapas de 1 minuto u otra de 20 etapas de igual tiempo, la más practicada suele ser la primera, en ambas versiones, las primeras etapas son de velocidad baja y tienen como objetivo permitirle al jugador familiarizarse con el test. García y compañero[29] presentan la representación gráfica del escenario que se utiliza en el test y que se observa en la Figura 1.

Existen dos motivos principales que representan la finalización del test para el deportista, tal y como lo menciona García en[30], la primera de ellas es cuando el jugador se detiene porque alcanzó la fatiga y la segunda, cuando por 2 veces consecutivas no llega a pisar detrás de la línea al sonido del «beep».

Figura 1. Representación gráfica del escenario utilizado en la ejecución del Test de Course – Navette.



Fuente: G. C. García and J. D. Secchi, “Test Course Navette de 20 metros con etapas de un minuto. Una idea original que perdura hace 30 años,” Apunt. Med. l’Esport, vol. 49, no. 183, (2014), p. 93–103.

El programa de automatización presentado en este trabajo fue diseñado para arrojar las evaluaciones correspondientes a la ejecución del Test de Course - Navette de 15 etapas.

[29] G. C. García and J. D. Secchi. Op. cit. p. 193-102.
[30] Libd., p.193-102.

Para la obtención de las evaluaciones deportivas correspondientes, el test maneja una plantilla o tabla en la cual se van registrando las etapas y los tramos que realiza el deportista; de acuerdo con esta información, y en base a la velocidad alcanzada en la última etapa completada (VFA - velocidad final alcanzada), que puede ser consultada en la Tabla N° 1, se calcula la capacidad pulmonar en términos del consumo máximo de oxígeno ($VO_{2màx}$) de cada jugador.

El consumo máximo de oxígeno puede ser calculado con la fórmula 1:

$$VO_{2màx} = (5.857 \times VFA) - 19.458 \qquad 1$$

Tabla 1. VFA para Test de Course – Navette de 15 etapas.

Rutina de Tiempos		
Etapas	**Tramos**	**VFA (Km/h)**
1	7	8,5
2	7	9
3	7	9,5
4	8	10
5	8	10,5
6	9	11
7	9	11,5
8	9	12
9	10	12,5
10	10	13
11	11	13,5
12	11	14
13	12	14,5
14	12	15
15	12	15,5

3.2. VISIÓN ARTIFICIAL.

La principal responsable del avance que han tenido las técnicas de análisis deportivo en los últimos años ha sido la Inteligencia Artificial y sus distintas ramas, entre las cuales destaca la Visión Artificial o Visión por computador, esta, como fue previamente definida, se encarga principalmente del procesamiento, análisis e interpretación de imágenes digitales. Actualmente, existen una gran cantidad de herramientas a nivel de software para la implementación de diversas aplicaciones de Visión Artificial, una de las principales es OpenCV.

3.2.1. OpenCV. OpenCV (Open Source Computer Visión Library)[31] es la librería de código abierto de Visión por Computador más grande que existe en la actualidad, ya que contiene implementaciones de más de 2500 algoritmos y se encuentra disponible en múltiples lenguajes de programación tales como Python, Java y C++. OpenCV fue desarrollada originalmente por Intel en el año 2000.

OpenCV se basa en una estructura de tipo modular, es decir, dentro del paquete completo del programa se encuentran múltiples librerías compartidas o estáticas para diferentes aplicaciones. Los principales módulos que se encuentran disponibles son los siguientes:

- Módulo Principal (Núcleo): Este módulo define las estructuras de datos básicas y las funciones base utilizadas por los demás módulos.
- Módulo de Procesamiento de Imágenes (imgproc): Este módulo contiene funciones para el filtrado de imágenes, transformaciones geométricas de imágenes, conversión de espacios de color, histogramas, entre otras.
- Módulo de Análisis de Video (video): Este módulo contiene algoritmos de estimación de movimiento, substracción del fondo y seguimiento de objetos.
- Módulo de Marco de Características 2D (features2D): Este módulo contiene algoritmos de detectores de características, descriptores y comparadores de descriptores.
- Módulo de Detección de Objetos (objdetect): Este módulo contiene algoritmos de detección de objetos de instancias de las siguientes clases predefinidas (caras, ojos, objetos, personas, carros, etc.)

Todas las clases y funciones de los distintos módulos de OpenCV se encuentran designadas dentro del espacio de nombres **cv,** esto significa que para acceder a ellas es necesario llamarlas con el especificador mencionado.

3.2.2. Python. Python[32] es un lenguaje de programación interpretado multiparadigma orientado a objetos que soporta la programación imperativa y, en cierta medida, la programación funcional. Python fue diseñado por Guido Van Rossum en el año 1991 y cuenta con licencia de código abierto.

Una de las principales ventajas de Python es su sintaxis robusta, escalable y de fácil entendimiento, que permite desarrollar algoritmos de gran complejidad en tan solo unas cuantas líneas de código; esta característica ha hecho de este lenguaje de programación el predilecto para variadas aplicaciones de Inteligencia Artificial tales como la visión por computador, big data, data science, entre otras, en el primer campo, el uso de Python en

[31] "OpenCV". {En línea}. {31 marzo de 2021} disponible en: (https://opencv.org/)
[32] "Python.org". {En línea}. {31 marzo de 2021} disponible en: (https://www.python.org/)

conjunto con OpenCV, conforman la API (Application Programming Interface) de mayor uso para la solución de problemas de Visión Artificial.

3.2.3. Estimación de la Pose Humana. Alrededor del campo de estudio de la Visión por Computador han surgido un gran número de aplicaciones, una de ellas es la estimación de la pose humana, la cual a su vez consiste, en la detección e identificación de las principales articulaciones del cuerpo humano de una o varias personas. Según lo mencionado en[33], los algoritmos de estimación de la pose humana en 2D para múltiples personas pueden agruparse en dos categorías dependiendo del enfoque:

1. Enfoque descendente (Top Down): Este enfoque se basa en el uso de una etapa inicial de detección, la cual utiliza recuadros delimitadores (Bounding Boxes) o segmentación, para luego, correr un detector de personas u objetos sobre los datos arrojados por estos métodos. Los principales sistemas que hacen uso de este enfoque son Detectron2, el cual se basa en el modelo Mask RCNN, LCRNet, entre otros.

2. Enfoque ascendente (Bottom Up): Este enfoque, contrario al anterior, utiliza una etapa inicial de detección para determinar la información estructural, es decir, todas las articulaciones y parámetros de asociación. Seguido a esto, utiliza un algoritmo de emparejamiento, el cual, en base a la información arrojada por la primera etapa, se encarga de ensamblar las poses de cuerpo completo. Uno de los principales sistemas que hace uso de este enfoque es OpenPose.

3.2.3.1. OpenPose. OpenPose[34] representa el primer sistema capaz de detectar en tiempo real puntos clave del torso humano, del rostro, de las manos y los pies (135 puntos en total) a partir de imágenes o videos para múltiples personas o para individuos solos. El funcionamiento de este programa está basado en un modelo entrenado en una red neuronal convolucional (CNN) para la estimación de la pose humana, la cual surgió de un trabajo realizado por investigadores de la universidad de Carnegie Mellon en Pensilvania. OpenPose hace uso de dos distintos tipos de bases de datos (datasets) o

[33] R. Jena, “Out of the Box: A combined approach for handling occlusions in Human Pose Estimation,” (2019).
[34] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, (2021), pp. 172–186.

modelos de entrenamiento para su detección, estos son, COCO KeyPoints[35] y MPII Human Pose[36].

En la Figura 2 se observan las salidas de ambos datasets ploteados sobre una misma persona.

El formato de salida de los puntos clave arrojados por el modelo de COCO es el siguiente:

Nariz – 0, Cuello – 1, Hombro derecho – 2, Codo derecho – 3, Muñeca derecha – 4, Hombro izquierdo – 5, Codo izquierdo – 6, Muñeca izquierda – 7, Punto derecho de cadera – 8, Rodilla derecha – 9, Tobillo derecho – 10, Punto izquierdo de cadera – 11, Rodilla izquierda – 12, Tobillo izquierdo – 13, Ojo derecho – 14, Ojo izquierdo – 15, Oreja derecha – 16, Oreja izquierda – 17, Fondo – 18.

Figura 2. Salidas del modelo COCO y modelo MPII.

COCO KeyPoints MPII KeyPoints

Fuente*:* Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, (2021), pp. 172–186.

El formato de salida de los puntos clave arrojados por el modelo MPII es el siguiente:
Cabeza – 0, Cuello – 1, Hombro derecho – 2, Codo derecho – 3, Muñeca derecha – 4, Hombro izquierdo – 5, Codo izquierdo – 6, Muñeca izquierda – 7, Punto derecho de

[35] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., p. 3686–3693. {En línea}. { 2015} disponible en: (http://arxiv.org/abs/1405.0312).
[36] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2D Human Pose Estimation: New Benchmark and State of the Art Analysis,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., no. December, (2014), p. 3686–3693.

cadera – 8, Rodilla derecha – 9, Tobillo derecho – 10, Punto izquierdo de cadera – 11, Rodilla izquierda – 12, Tobillo izquierdo – 13, Pecho – 14, Fondo – 15.

3.2.3.2. Detectron2. Detectron2[37] es un software de detección de objetos producto del trabajo del grupo de investigación en AI (Artificial Intelligence) de Facebook (FAIR)[38]; esta segunda versión corresponde a una reescritura desde cero de la versión inicial del programa, Detectron[39], la cual funcionaba bajo el modelo de Deep Learning, Caffe2. El nuevo sistema esta implementado en PyTorch.

Detectron2 incluye implementaciones de algoritmos de detección de última generación, entre los que se encuentran DensePose[40], Panoptic Feature Pyramid Networks[41], TensorMask[42] y numerosas variantes de la familia de algoritmos Mask R-CNN[43], desarrollada también por FAIR.

Modelos de Detectron2. Detectron2 cuenta con una larga lista de modelos pre-entrenados[44] que pueden ser fácilmente acoplados al sistema implementado; entre los principales modelos destacan aquellos que han sido entrenados en el dataset de COCO, estos son: COCO Instance Segmentation Baselines with Mask R-CNN, COCO Person Keypoint Detection Baselines with Keypoint R-CNN y COCO Panoptic Segmentation Baselines with Panoptic FPN. Para el desarrollo del algoritmo en mención se hizo uso únicamente del modelo Keypoint Detection Baselines with Keypoint R-CNN.

Person KeyPoint Detection. El algoritmo de este modelo pre entrenado hace parte del enfoque ascendente, mencionado anteriormente, y permite identificar los principales puntos clave de las personas detectadas por el programa. La salida que se obtiene de Detectron2 al utilizar este modelo se observa en la Figura 3.

---

[37] "GitHub - Detectron2". {En línea}. {31 marzo de 2021} disponible en: (https://github.com/facebookresearch/detectron2).
[38] "Facebook AI". {En línea}. {31 marzo de 2021} disponible en:(https://ai.facebook.com/).
[39] "GitHub - facebookresearch/Detectron". {En línea}. {10 marzo de 2021} disponible en:(https://github.com/facebookresearch/Detectron/).
[40] R. A. G, N. Neverova, and I. Kokkinos, "DensePose: Dense Human Pose Estimation In The Wild", (2018), p. 1–12.
[41] A. Kirillov, P. Dollár, R. Girshick, and K. He, "Panoptic Feature Pyramid Networks," (2019), p. 10.
[42] X. Chen, P. Dollár, R. Girshick, and K. He, "TensorMask: A Foundation for Dense Object Segmentation," (2019), p. 12.
[43] K. He, G. Gkioxari, R. Girshick, and P. Dollár, "Mask R-CNN," p. 10.
[44] "Model Zoo Detectron2". {En línea}. {10 marzo de 2021} disponible en: (https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md).

Figura 3. Salida de Detectron2 al utilizar el modelo de detección de Keypoints.

Fuente: "GitHub - Detectron2". {En línea}. {31 marzo de 2021} disponible en: (https://github.com/facebookresearch/detectron2).

Las principales funciones y líneas de código de las que hace uso este software de detección son:

- $cfg = get\_cfg()$: El comando get_cfg () se utiliza para obtener la configuración por defecto del detector, la cual se asigna a la variable cfg.
- $predictor = DefaultPredictor(cfg)$: A través de la clase DefaultPredictor () se crea un predictor simple con la configuración dada (cfg), el cual se ejecuta en un solo dispositivo para una sola imagen de entrada.
- $outputs = predictor(frame)$: Se llama al objeto del predictor previamente configurado y se le pasa como argumento el frame actual, de esta forma, el programa hace la detección de las personas sobre esa imagen de entrada. Los resultados de la detección se almacenan en una variable (outputs) en forma de diccionario.
- $instances = outputs['instances']$: Se accede a una instancia del diccionario generado ('instances'), la cual almacena las coordenadas de los recuadros que delimitan los objetos encontrados, los keypoints y los confidence scores, estos últimos corresponden a valores de umbral asignados a las detecciones realizadas. Para acceder a las coordenadas de los recuadros y los Keypoints se utilizan los campos pred_boxes y pred_keypoints, respectivamente.

3.2.4. Seguimiento de Objetos. El termino de seguimiento de objetos hace referencia a la localización de un objeto en los frames sucesivos de un video. Tal y como se menciona

en[45], un algoritmo de seguimiento se inicializa con un frame de una secuencia de vídeo y un recuadro delimitador que indica la ubicación del objeto al que se va a hacer seguimiento. A partir de dichos parámetros, el algoritmo de seguimiento se encarga de generar, para el objeto en cuestión, un recuadro delimitador que identifique al objeto en los frames posteriores.

La detección y el seguimiento de objetos u personas son dos algoritmos distintos que se complementan el uno al otro, según[46] las principales razones por las que es aconsejable implementar ambas etapas en un programa de detección son las siguientes:

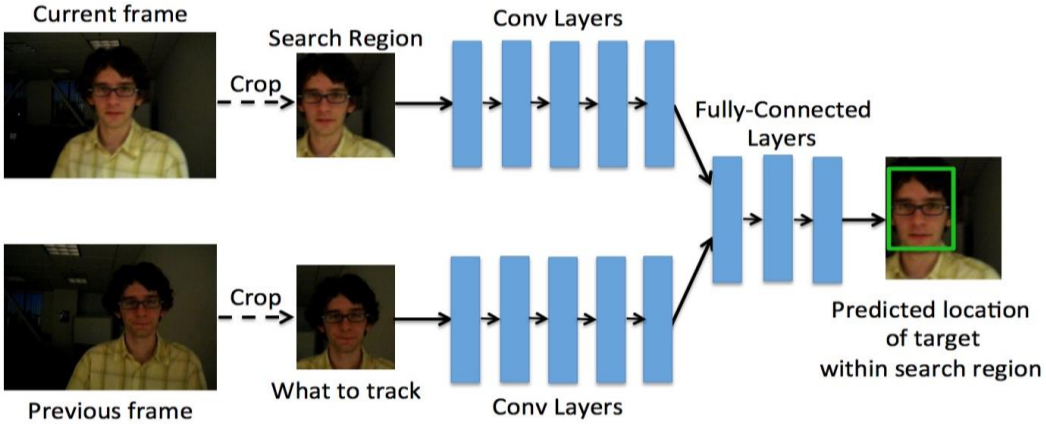
- El algoritmo de seguimiento es más rápido que el algoritmo de detección. Esto se debe a que un algoritmo de seguimiento utiliza toda la información que ha recolectado de un objeto en frames anteriores para predecir su ubicación en el frame posterior, contrario a lo que hace un algoritmo de detección, el cual siempre inicia desde cero. Por este motivo, los sistemas de detección y seguimiento se suelen diseñar para que el algoritmo de detección se ejecute cada n^{th} frame mientras que el algoritmo de seguimiento se ejecuta en los frames n -1 intermedios.
- El algoritmo de seguimiento puede mantener la detección de un objeto incluso si el algoritmo de detección falla debido a la oclusión.
- El algoritmo de seguimiento preserva la identidad del objeto u persona en cuestión. Cuando se ejecuta un algoritmo de detección, la salida que se obtiene corresponde a un array de coordenadas de rectángulos delimitadores sin ningún tipo de información que las conecte al objeto que representan.

Actualmente existen muchos programas de seguimiento, unos son más robustos que otros, todo depende de la aplicación en la que se vaya a utilizar. Algunos de los más comunes son:

3.2.4.1. GOTURN. GOTURN (Generic Object Tracking Using Regression Networks) es un algoritmo de seguimiento que aprende a rastrear objetos en tiempo real en videos fuera de línea. El modelo de GOTURN realiza el seguimiento de los objetos genéricos mediante redes de regresión, las cuales están entrenadas a través de redes neuronales, lo que les permite aprender una relación genérica entre la apariencia y el movimiento de los objetos nuevos de una manera rápida, sólida y precisa.

[45] "GOTURN : Deep Learning based Object Tracking | Learn OpenCV". {10 marzo de 2021} disponible en: (https://learnopencv.com/goturn-deep-learning-based-object-tracking/).
[46] "Object Tracking using OpenCV (C++/Python)". {10 marzo de 2021} disponible en: (https://learnopencv.com/object-tracking-using-opencv-cpp-python/).

Figura 4. Funcionamiento de la Arquitectura de GOTURN.



Fuente: D. Held, S. Thrun, and S. Savarese, "Learning to Track at 100 FPS with Deep Regressions Networks," Comput. Vis. – ECCV 2016 Lect. Notes Comput, p. 749–765, [Online]. Available: {10 marzo de 2021} disponible en:(http://davheld.github.io/GOTURN/GOTURN.html).

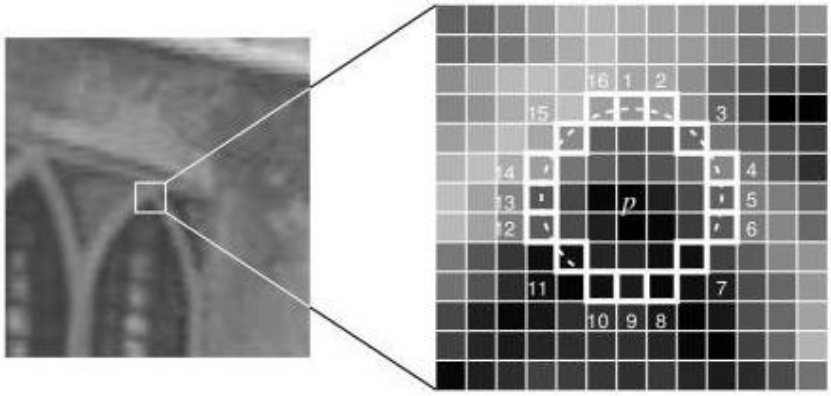
GOTURN fue presentado por David Held, Sebastian Thrun y Silvio Savarese en su artículo[47], en el cual presentan la arquitectura de la red para el seguimiento. Como se observa en la Figura 4, a la red se ingresa una región de búsqueda tomada del frame actual y el objeto a rastrear tomado del frame anterior (también denominado primer frame). La red aprende a comparar estos frames para encontrar la ubicación del objeto en la imagen actual (también denominado segundo frame), en el momento en que el objeto es detectado, se realiza un recuadro delimitador sobre este.

3.2.4.2. ORB. ORB (Oriented FAST and Rotated BRIEF) fue desarrollado en los laboratorios de OpenCV por Ethan Rublee, Vincent Rabaud, Kurt Konolige y Gary R. Bradski en 2011[48], como una alternativa eficiente, viable y de acceso libre a los métodos basados en extracción de características SIFT y SURF[49]. ORB se compone de una fusión del detector de puntos FAST[50] y el descriptor BRIEF[51]. Ambas técnicas son atractivas por su buen desempeño y bajo costo**.**

[47] D. Held, S. Thrun, and S. Savarese, "Learning to Track at 100 FPS with Deep Regressions Networks," Comput. Vis. – ECCV 2016 Lect. Notes Comput. {10 marzo de 2021}, p. 749–765 disponible en:(http://davheld.github.io/GOTURN/GOTURN.html).
[48] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," Proc. IEEE Int. Conf. Comput. (2011) p. 2564–2571.
[49] Y. Sakai, T. Oda, M. Ikeda, and L. Barolli, "An Object Tracking System Based on SIFT and SURF Feature Extraction Methods," 18th Int. Conf. Network-Based Inf., vol. 18, (2015), pp. 561–565.
[50] D. G. Viswanathan, "Features from Accelerated Segment Test (FAST)", (2011), p. 5.
[51] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6314, (2010), p. 778–792.

El funcionamiento de ORB se basa en la clasificación de puntos clave; dado un píxel p en un array, el detector FAST se encarga de comparar el brillo del píxel p con 16 pixeles en una zona pequeña alrededor de él, tal y como se observa en la Figura 5. Los pixeles del círculo alrededor de p se clasifican en tres clases: más claros que p, más oscuros que p y similares a p; si dicha clasificación tiene más de 8 pixeles oscuros o brillantes, este es denominado un punto clave. Los puntos clave encontrados brindan información sobre la ubicación de los bordes en una imagen.

Figura 5. Clasificación de puntos clave realizada por el detector FAST.



Fuente: E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” Proc. IEEE Int. Conf. Comput. (2011) p. 2564–2571

3.2.4.3. CentroidTracker. El CentroidTracker[52] es un algoritmo de seguimiento de objetos que está basado en la distancia Euclidiana entre el centroide de un objeto ya existente (es decir, objetos que el centroidtracker ya ha visto) y el centroide de un nuevo objeto localizado en los frames posteriores del video analizado.

El funcionamiento de este algoritmo de seguimiento se resume en los pasos que se observan en la Figura 6. A continuación se describen cada uno de ellos y se mencionan las funciones utilizadas.

- *Acepta las coordenadas de los recuadros de los objetos detectados y computa el centroide de cada recuadro*. El algoritmo del CentroidTracker recibe un set de coordenadas (X1, Y1, X2, Y2) de los recuadros delimitadores para cada objeto detectado en cada frame, y a partir de esto, calcula el centroide (Cx, Cy) de cada uno de ellos utilizando la fórmula 2:

[52] “Simple object tracking with OpenCV - PyImageSearch”. {10 marzo de 2021} disponible en: (https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/)

$$C_X = \frac{(X_1 + X_2)}{2} \;\; ; \;\; C_Y = \frac{(Y_1 + Y_2)}{2} \qquad 2)$$

- *Calcula la distancia euclidiana entre los nuevos rectángulos y los objetos existentes*. Para cada frame posterior se repite el primer paso, sin embargo, en lugar de asignar un identificador nuevo a cada objeto, se hace un chequeo para determinar si se pueden relacionar esos nuevos centroides (inputCentroids) con los centroides ya computados (objectCentroids), puntos amarillos y morados, respectivamente, en la Figura 6. Para esto se obtiene la distancia euclidiana (Flechas Verdes) entre cada par de objetos existentes y de objetos nuevos haciendo uso de la expresión 3. Del ejemplo de la Figura 6 paso 2 se observan tres objetos detectados, los dos pares de puntos que están más cerca el uno del otro son dos objetos ya existentes.

$$D = dist.cdist(np.array(objectCentroids), inputCentroids) \qquad 3)$$

  La anterior línea de código utiliza la función dist.cdist del paquete *scipy.spatial*[53] de la librería SciPy[54]. A través de ella se calcula la distancia entre cada par de objetos de las dos listas de entrada (objectCentroids, inputCentroids).

- *Actualiza las coordenadas (X, Y) de los objetos existentes*. La principal función del algoritmo del CentroidTracker es que un objeto dado se moverá en frames posteriores del video, pero la distancia entre el centroide del frame actual F_t y el centroide del frame posterior F_{t+1} va a ser más pequeña que todas las otras distancias entre los objetos existentes; es de esta forma, asociando centroides con las distancias más pequeñas entre frames posteriores, como se construye este algoritmo de seguimiento. En la Figura 6 paso 3, se observa como el algoritmo elige asociar aquellos centroides cuya distancia euclidiana sea mínima.

- *Registra nuevos objetos*. En el caso de que el número de objetos detectados sea mayor que el número de objetos siendo rastreados, el algoritmo procede a registrar los nuevos objetos a través del método $register(centroid, rect)$, el cual acepta como parámetros de entrada el centroide calculado y las coordenadas del rectángulo. Luego de registrar el objeto se le asigna un nuevo identificador.
  En la Figura 6 paso 3, el centroide que no se pudo asociar a otro centroide del frame anterior, se registra como un nuevo objeto y se le asigna el ID 3, como se observa en la Figura 6 paso 4. En este punto el algoritmo vuelve al paso número

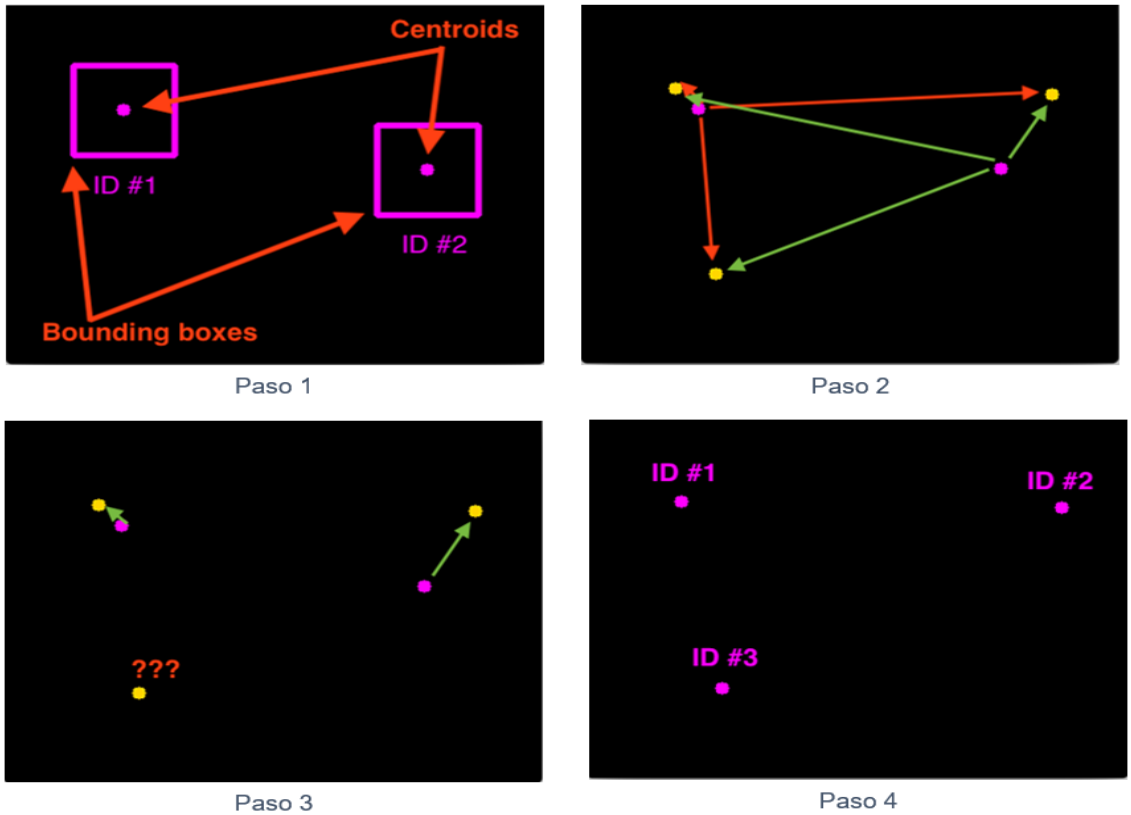
[53] "Spatial data structures and algorithms (scipy.spatial) — SciPy". {02 abril de 2021} disponible en: (https://scipy.github.io/devdocs/tutorial/spatial.html).
[54] "SciPy.org — SciPy.org". {02 abril de 2021} disponible en: (https://www.scipy.org/).

dos y repite la serie de pasos descrita anteriormente para cada frame del video que se está analizando.

- *Elimina objetos viejos*. El último paso que realiza el algoritmo es eliminar los identificadores de los objetos cuando estos se pierden de vista mediante el método $deregister(objectID)$; para esta implementación se eliminan objetos cuando no se puede hacer match con un objeto ya existente pasados 50 frames.

Todas estas funciones se ejecutan dentro del método principal del algoritmo, update, el cual se explica a profundidad en el documento *Explicación Códigos,* cuya ubicación se enuncia en el Anexo A del presente texto.

Figura 6. Etapas de Funcionamiento del Algoritmo del CentroidTracker.



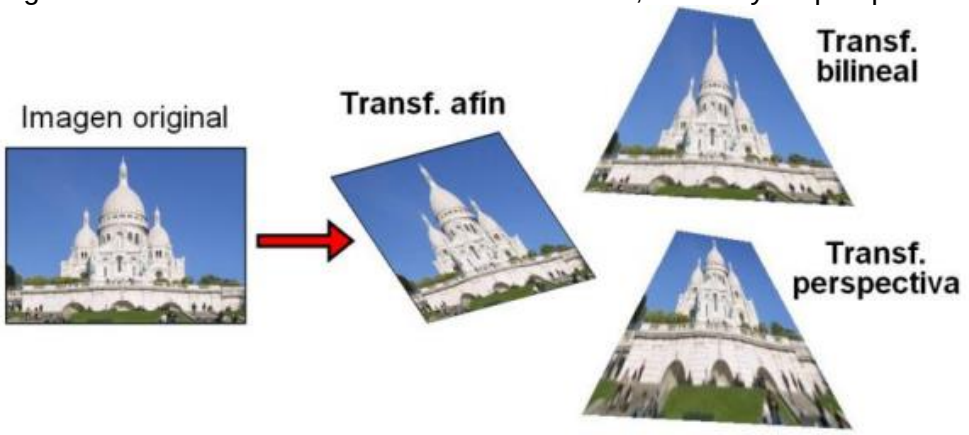
Fuente: “Simple object tracking with OpenCV - PyImageSearch”. {10 marzo de 2021} disponible en: (https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/)

3.2.5. Transformación de Perspectiva. Las transformadas geométricas están definidas como la relación existente entre los puntos de dos imágenes, y se suelen representar por

operaciones matriciales[55]. Existen diferentes categorías de transformadas geométricas, las principales son:

- Transformaciones Básicas: Dentro de las transformadas básicas se encuentran la traslación, la rotación, cambio de escala, simetría y deformación o inclinación.
- Transformadas Afines: Las transformadas afines son aquellas que conservan la rectitud y el paralelismo de la imagen. Una transformada afín se concibe como una transformada lineal de una coordenada, que incluye las transformaciones básicas.
- Transformación de Perspectiva y Bilineal: La transformación de perspectiva es aquella que transforma las líneas rectas en líneas rectas que no conservan, necesariamente, su paralelismo. La transformación bilineal es una simulación de la transformada de perspectiva y aunque su resultado no es exactamente igual, permite obtener una perspectiva de la imagen mapeada que varía de acuerdo con el efecto de perspectiva que se le dé. En la Figura 7 se observan las diferencias entre los tres tipos de transformadas.

Figura 7. Diferencias entre la transformación afín, bilineal y de perspectiva.



Fuente: D. F. Villamarín Zapata, “Estado del Arte, Herramientas y Aplicaciones para Transformaciones Geométricas 3D,” X Congr. Cienc. y Tecnol. (2015), p. 226–231.

Una de las aplicaciones habituales de la transformación de perspectiva, en conjunto con el mapeo de coordenadas, es la proyección en 2D de una representación en 3D, como lo suele ser un video deportivo; actualmente, en muchos ambientes deportivos tales como el basquetbol o el fútbol se realizan transformaciones de perspectiva para mapear las posiciones de los jugadores a un plano 2D, y de esta forma poder obtener las métricas del juego. En la Figura 8 se observa la proyección en 2D de un juego de basquetbol.

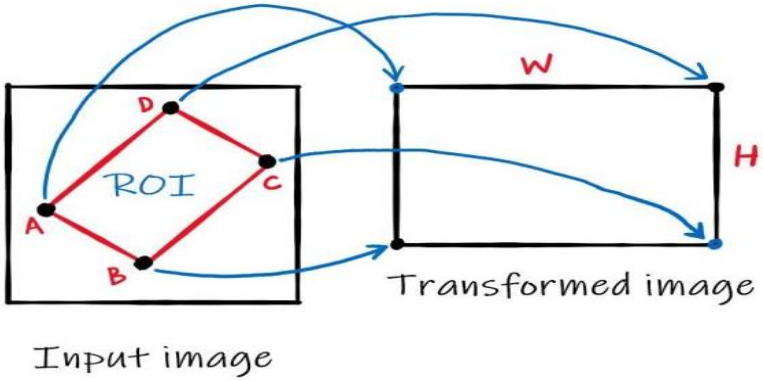
[55] D. F. Villamarín Zapata, “Estado del Arte, Herramientas y Aplicaciones para Transformaciones Geométricas 3D,” X Congr. Cienc. y Tecnol. (2015), p. 226–231.

Figura 8. Transformación de Perspectiva de un juego de Basquetbol

Fuente: E. Cheshire, C. Halasz, and J. K. Perin, “Player Tracking and Analysis of Basketball Plays,” (2015).

Para realizar una transformación de perspectiva es necesario obtener una matriz de transformación 3x3, para la cual es necesario definir los 4 puntos que delimitan la región de interés a transformar en la imagen de entrada y los puntos correspondientes en la imagen de salida. La Figura 9 resume la idea básica detrás del proceso de transformación.

Figura 9. Transformación de Perspectiva en una Imagen.



Fuente: “Perspective Transformation”. {11 abril de 2021} disponible en: (https://theailearner.com/tag/cv2-warpperspective/).

Para realizar la transformación de perspectiva se hace uso de las siguientes funciones, dos de ellas hacen parte del grupo de funciones que tiene OpenCV, dentro del Módulo de Procesamiento de Imágenes descrito anteriormente, para este tipo de operaciones geométricas[56].

[56] OpenCV: Geometric Image Transformations”. {31 marzo de 2021} disponible en: (https://docs.opencv.org/master/da/d54/group__imgproc__transform.html).

- get_mouse_points (event, x, y): Esta función realiza la selección de los cuatro puntos de origen de manera interactiva a través de la captura de los eventos del mouse.
- cv2.getPerspectiveTransform(src, dst): Esta función calcula la matrix de transformación 3x3 que será utilizada para aplicar la transformación de perspectiva a la imagen de entrada y obtener la imagen transformada final; los parámetros de entrada de esta función son:

    - src: Coordenadas de los cuatro puntos de la región de interés en la imagen de entrada.
    - dst: Coordenadas de los cuatro puntos destino definidos a partir del orden específico para los puntos de origen.

- cv2.warpPerspective(): Esta función aplica la transformación de perspectiva sobre la imagen de entrada haciendo uso de la matrix previamente calculada. Los parámetros de entrada son:

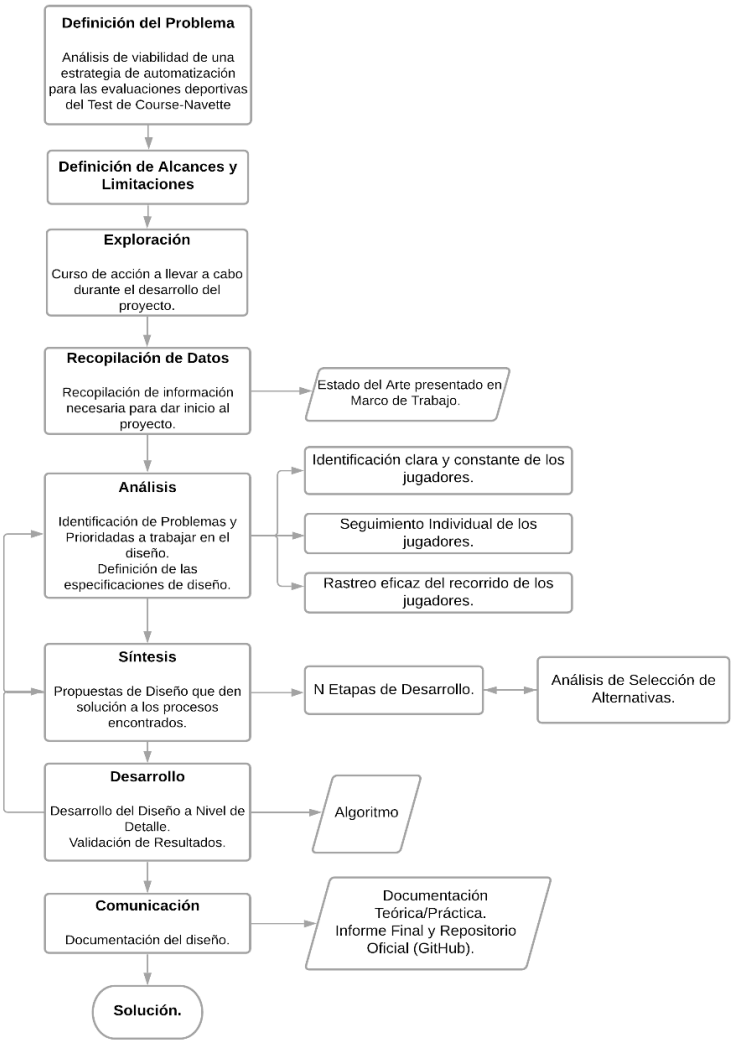
    - img: Imagen de entrada.
    - M: Matriz de transformación de 3x3.
    - dsize: tamaño de la imagen de salida (columnas, filas).

4. METODOLOGÍA Y RESULTADOS

La metodología utilizada en este proyecto está basada en el desarrollo de Bruce Archer, presentado por Cross en[57]. Según Archer, el proceso de diseño debe contener las etapas analíticas, creativa y de ejecución, las cuales a su vez se encuentran divididas en las fases que se mencionan a continuación. El diagrama metodológico implementado se observa en la Figura 10.

Etapa Analítica: Definición del problema, Alcances y Limitaciones del Proyecto, Programación y Recopilación de datos. Etapa Creativa: Análisis de problemas y prioridades, definición de especificaciones, síntesis de propuestas y desarrollo del diseño. Etapa Ejecutiva: Validación del diseño y comunicación de resultados.

Figura 10. Diagrama Metodológico.



[57] N. Cross, “Design Research: A Disciplined Conversation” Des. Issues, vol. 15, (abril de 1999), pp. 5–10.

De acuerdo con el diagrama planteado, la definición del problema y de los alcances y limitaciones del proyecto dan forma a las fases de diseño sucesivas; este último parámetro, previamente definido en el anteproyecto de este trabajo, enuncia que el desarrollo de este se encuentra limitado, en primer lugar, al cumplimiento de los objetivos y, en segundo lugar, a las condiciones de desarrollo impuestas por el elevado costo computacional generado durante la ejecución de algorimos de reconocimiento y automatización, es decir, no se realizaran evaluaciones en tiempo real.

En base a esto, el curso de exploración planeado para el desarrollo del proyecto, consiste, primero, en una indagación de los trabajos realizados en esta área de estudio, de los cuales, se identifican los posibles problemas y prioridades a trabajar durante el desarrollo del algoritmo y se definen, las especificaciones de diseño que permitan cubrir dichos factores encontrados. Luego de esto, se desarrollan las propuestas de diseño que cumplan con las especificaciones presentadas, estas propuestas se dividen en etapas de desarrollo que se encuentran sujetas a la valoración de diferentes alternativas, y que, en base al analisis de selección realizado, se implementan en forma modular dentro del programa de automatización; posterior a la obtención de este algoritmo, se realiza la verificación de resultados correspondiente y se presenta la documentación del diseño final realizado. Cabe recalcar que las fases de análisis, sintesis y desarrollo no son lineales sino que, por el contrario, se iteran de acuerdo a los resultados que se van obteniendo en cada una de ellas.

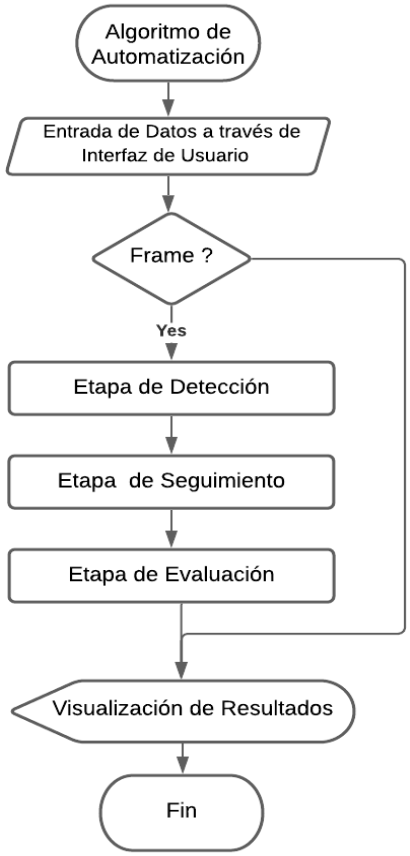
4.1. ETAPAS DE DESARROLLO

Las propuestas de diseño planteadas consisten en tres etapas de desarrollo: Etapa de Detección, Etapa de Seguimiento y Etapa de Evaluación; cada una de estas etapas se implementa de forma modular, de modo que terminan siendo importadas como funciones dentro del algoritmo general, este tipo de organización hace que el programa sea más legible y menos extenso. El funcionamiento que se espera obtener del programa principal se encuentra resumido en el diagrama de flujo de la Figura 11.

El algoritmo de automatización tiene como parámetros de entrada el archivo del video a evaluar, los puntos de la región de interés seleccionados en la etapa de evaluación y el tiempo de inicio en segundos, dentro del archivo de video, del test. La lectura del video se realiza frame a frame, para efectos de explicación se añadieron los bloques de lectura de video a cada uno de los diagramas presentados en las Figuras 12,13 y 14, sin embargo, y dada la estructura modular utilizada, solo se realiza una lectura del video dentro del algoritmo principal. Luego de la ejecución de las etapas propuestas, se efectúa

la visualización de resultados, de forma tal, que sea fácil de entender para el usuario y que permita realizar la posterior verificación de los resultados.

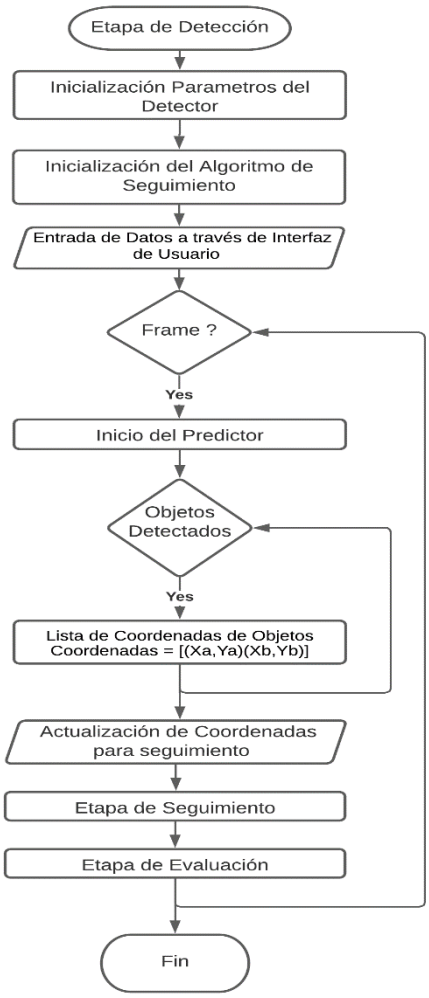
Figura 11. Diagrama de Flujo – Programa Principal.



4.1.1. Etapa 1: Detección. El primer paso para el desarrollo del programa de automatización es la obtención de las coordenadas de ubicación de los jugadores presentes en la prueba; esta etapa, que también se puede denominar etapa de Detección, es clave en el desarrollo del programa debido a que permite identificar los movimientos del jugador. El objetivo de esta etapa es lograr una identificación clara y constante del deportista.

El funcionamiento que se espera obtener del algoritmo planteado para la etapa de detección se encuentra resumido en el diagrama de flujo de la Figura 12.

Figura 12. Diagrama de Flujo – Etapa de Detección.



El primer paso en el algoritmo es la inicialización de las características de funcionamiento del detector seleccionado, posterior a esto, se realiza la invocación del algoritmo de seguimiento, el cual, toma las coordenadas de los objetos detectados para realizar el respectivo rastreo.

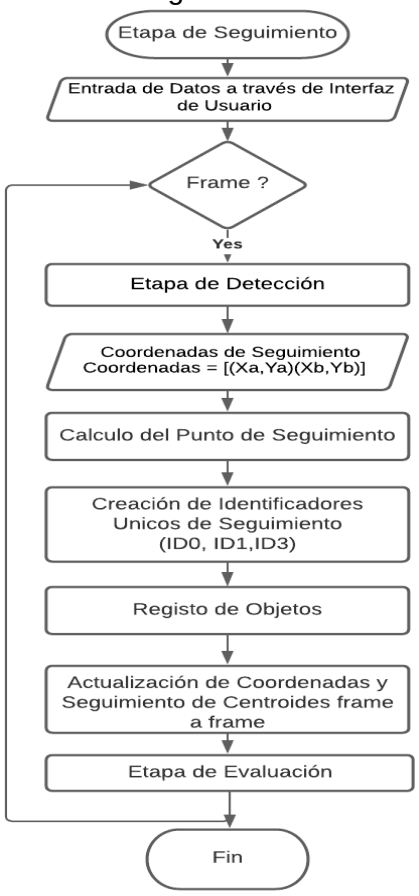
En cada uno de los frames leídos, el algoritmo de detección procede a correr el predictor, previamente inicializado, y a almacenar la información de los objetos detectados; inmediatamente después se implementa un ciclo, el cual recorre la lista de los objetos detectados para ir almacenando las coordenadas de los recuadros delimitadores.

El ciclo de detección finaliza cuando se termina la lectura de los frames del video o si el usuario decide interrumpir la ejecución.

4.1.2. Etapa 2: Seguimiento. Luego de la etapa de detección y con el fin de tener registro en todo momento de la ubicación y el recorrido de cada uno de los jugadores, se desarrolla una etapa complementaria a la anterior, cuyo objetivo principal es llevar el seguimiento individual de cada uno de los deportistas a lo largo de todo el video.

El funcionamiento que se espera obtener del algoritmo planteado para la etapa de seguimiento se encuentra resumido en el diagrama de flujo de la Figura 13.

Figura 13. Diagrama de Flujo – Etapa de Seguimiento.



En cada uno de los frames leídos, el algoritmo de seguimiento toma las coordenadas de los recuadros para los objetos identificados, provenientes del algoritmo de detección y procede a calcular el punto de seguimiento de cada uno de estos objetos; posterior a esto, realiza la creación y registro de los identificadores únicos de seguimiento para cada uno de ellos.

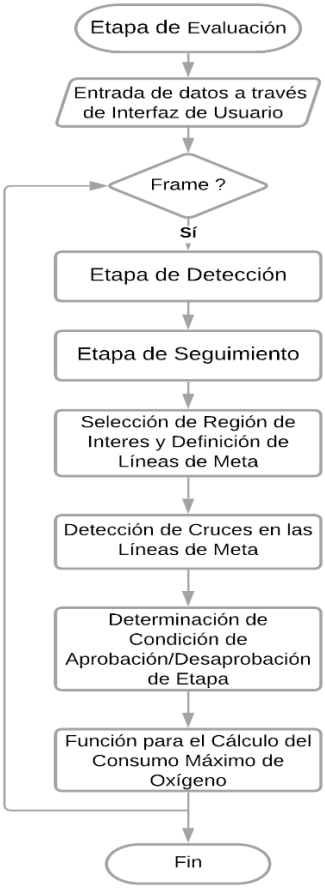
Finalizado el proceso de seguimiento en el frame actual, el algoritmo actualiza las coordenadas del punto de seguimiento para su posterior utilización en la siguiente etapa de desarrollo dentro del algoritmo principal.

4.1.3. Etapa 3: Evaluación. Una vez realizado el proceso de detección y seguimiento de los deportistas presentes en el video de prueba, se desarrolla una etapa que permita obtener la evaluación deportiva realizada a los jugadores durante la ejecución del Test de Course Navette. El principal resultado a obtener es el consumo máximo de oxígeno, a través del cual, se mide la capacidad pulmonar o resistencia cardiovascular del deportista que realiza el test, según Rodríguez en[58] esta medida se basa en la capacidad funcional del aparato circulatorio y respiratorio de ajustarse y recuperarse de los efectos del ejercicio muscular.

El funcionamiento que se espera obtener del algoritmo planteado para la etapa de evaluación se encuentra resumido en el diagrama de flujo de la Figura 14.

[58] F. Rodríguez Guisado, “Prescripción de Ejercicio para la Salud (I): Resistencia Cardiorrespiratoria,” Apunt. - Educ. Física y Deport., no. 39,(1995), p. 87–102.

Figura 14. Diagrama de Flujo – Etapa de Evaluación.



Luego de la lectura frame a frame del archivo de video, el algoritmo de evaluación implementa tres subetapas básicas. La primera se utiliza para seleccionar la región de interés dentro de la cual se llevará a cabo el análisis de evaluación y, también, para definir las líneas de meta del escenario deportivo. La segunda, debe identificar los cruces de los jugadores con las líneas de meta definidas y, la tercera, en base a la rutina de tiempos bajo la cual se desarrolla el test, debe determinar si el jugador aprobó o no el tramo de etapa en el que se encuentre en el momento.

En base a esa condición de aprobación/desaprobación se implementa una función que toma el valor de VFA (Velocidad Final Alcanzada) asignado a la última etapa realizada por el deportista y, haciendo uso de la fórmula 1, calcula el consumo máximo de oxígeno para el jugador.

4.2. ALTERNATIVAS ALGORÍTMICAS.

Las alternativas propuestas a continuación para las tres etapas de desarrollo son producto de la fase de recopilación de datos realizada y expuesta en el capítulo 4 de este documento. Los principales parámetros que se tuvieron en cuenta para su selección fueron, en primer lugar, que se tratara de programas que contaran con licencia de software libre, para que no existiera problema alguno con su implementación dentro del programa a desarrollar, y, en segundo lugar, que existiera documentación teórica y técnica disponible que sirviera de apoyo durante su puesta en funcionamiento.

4.2.1. Etapa 1: Detección. El análisis de selección de alternativas a evaluar para la etapa de detección se resume en la Tabla N°2.

Tabla 2. Alternativas Algorítmicas para la Etapa de Detección.

Alternativas Algorítmicas	Parámetros de Evaluación			
	Detección Múltiple de Personas	Software Libre	Documentación Disponible	Ejemplos Prácticos
OpenPose	Si	Si	Si	Si
Detectron2	Si	Si	Si	Si

En la sección 4.4.1.3 del documento se presenta el análisis de selección de la alternativa final a implementar en la etapa de detección, en base a los resultados obtenidos.

4.2.2. Etapa 2: Seguimiento. El análisis de selección de alternativas a evaluar para la etapa de seguimiento se resume en la Tabla N°3.

Tabla 3. Alternativas Algorítmicas para la Etapa de Seguimiento.

Alternativas Algorítmicas	Parámetros de Evaluación			
	Seguimiento de Objetos	Software Libre	Documentación Disponible	Ejemplos Prácticos
GOTURN	Si	Si	Si	Si
ORB	Si	Si	Si	Si
CentroidTracker	Si	Si	Si	Si

En la sección 4.4.2.4 del documento se presenta el análisis de selección de la alternativa final a implementar en la etapa de seguimiento, en base a los resultados obtenidos.

4.2.3. Etapa 3: Evaluación. El análisis de selección de alternativas a evaluar para la etapa de evaluación se resume en la Tabla N°4.

Tabla 4. Alternativas Algorítmicas para la Etapa de Evaluación.

Alternativas Algorítmicas	Parámetros de Evaluación			
	Detección de Cruces en líneas de Meta	Software Libre	Documentación Disponible	Ejemplos Prácticos
Métodos de Estimación de Distancia en Videos.	Si	Si	Si	No
Transformación de Perspectiva	Si	Si	Si	Si

En la Tabla N°4, la alternativa definida como métodos de estimación de distancia en videos, corresponde más que a un programa, como las demás opciones, a una técnica de software utilizada en algunos trabajos de detección de objetos, es por esta razón que no fue presentada como tal dentro del capítulo 4. Esta técnica consiste en el cálculo de

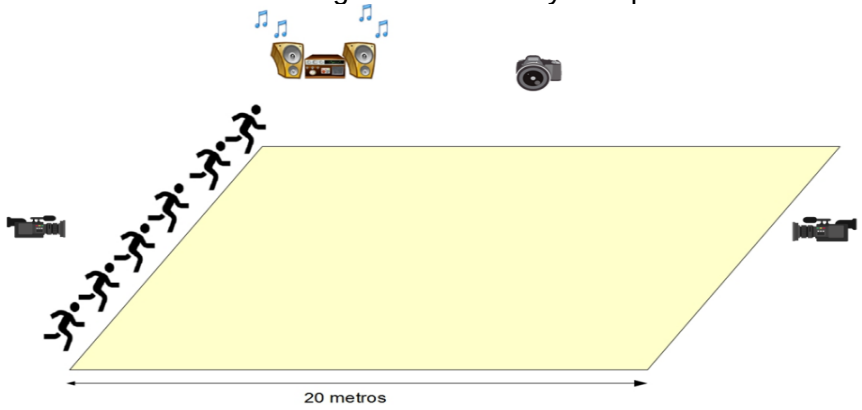
la distancia entre un objeto y la cámara utilizada para la grabación del video; Maga y Diamantas presentan en [59] y [60] respectivamente, dos técnicas de estimación en videos grabados con una sola cámara de video. Aunque esta alternativa no cuenta con ejemplos prácticos, fue tenida en cuenta durante el proceso de análisis de implementación del algoritmo de evaluación.

En la sección 4.4.4 del documento se presenta el análisis de selección de la alternativa final a implementar en la etapa de evaluación, en base a los resultados obtenidos.

4.3. ALTERNATIVAS DE ARREGLO DE CÁMARAS Y PERSPECTIVA DE GRABACIÓN.

Paralelo al proceso de evaluación de alternativas para las etapas de desarrollo, se analizaron diferentes arreglos de cámaras y perspectivas de grabación que permitieran optimizar los resultados obtenidos. En la Figura 15 se presenta el diagrama de la alternativa esbozada.

Figura 15. Alternativa de Arreglo de Cámaras y Perspectiva de Grabación.



Para la grabación del escenario deportivo se planteó la utilización de tres cámaras, dos ubicadas en las líneas de meta y una al lado del escenario, de tal forma que sea posible capturar la ejecución total del test desde diversas perspectivas de grabación. No se realizaron especificaciones técnicas con respecto al tipo de cámara y a la altura de su ubicación ya que se dejó como un parámetro a evaluar a partir de los resultados obtenidos y, sobre todo, de las herramientas y del espacio disponible por parte de los usuarios finales del programa.

[59] J. B. Maga, “Estimación de la Distancia a un Objeto con Visión Computacional,” Ingeniería, vol. 21, (2017), pp. 31–40.
[60] S. Diamantas, S. Astaras, and A. Pnevmatikakis, “Depth Estimation in Still Images and Videos Using a Motionless Monocular Camera”. (marzo de 2018), p. 129–134.

4.4. RESULTADOS

Para las pruebas realizadas durante la implementación del algoritmo de automatización se utilizaron cuatro videos de prueba, en la Tabla N°5 se resumen las características técnicas de cada uno de ellos. Cabe aclarar que la grabación de estos videos se realizó en base a los resultados que se iban obteniendo en las pruebas realizadas en cada etapa.

Tabla 5. Características Técnicas de los Videos de Prueba.

Videos de Prueba	Características				
	Duración (min: seg)	**Frames/seg (FPS)**	**Tamaño de Video**	**Perspectiva de Grabación**	**Nº Jugadores**
Video 1	10:24	30	640 x 352	Frontal	10
Video 2.1	10:24	30	1280 x 720	Frontal	10
Video 2.2	10:31	26,87	1280 x 720	Frontal Elevada	10
Video 2.3	10:31	29,97	1920 x 1080	Lateral Elevada	10
Video 3	10:45	30	1920 x 1080	Frontal	4
Video 3.1	2:30	30	1280 x 720	Lateral	4
Video 4	8:10	30	1920 x 1080	Frontal - Elevada	3

A continuación, se presentan los resultados obtenidos para las tres etapas de desarrollo con las alternativas algorítmicas planteadas en la anterior sección. De igual forma, se realiza la discusión correspondiente a las problemáticas encontradas y a la selección de alternativas a utilizar de forma definitiva en la implementación, y, se explica, resumidamente y de forma técnica, la funcionalidad de cada uno de los algoritmos. Junto a cada una de las imágenes resultado se adjunta un link de YouTube en el que se encuentra el video completo de cada prueba.

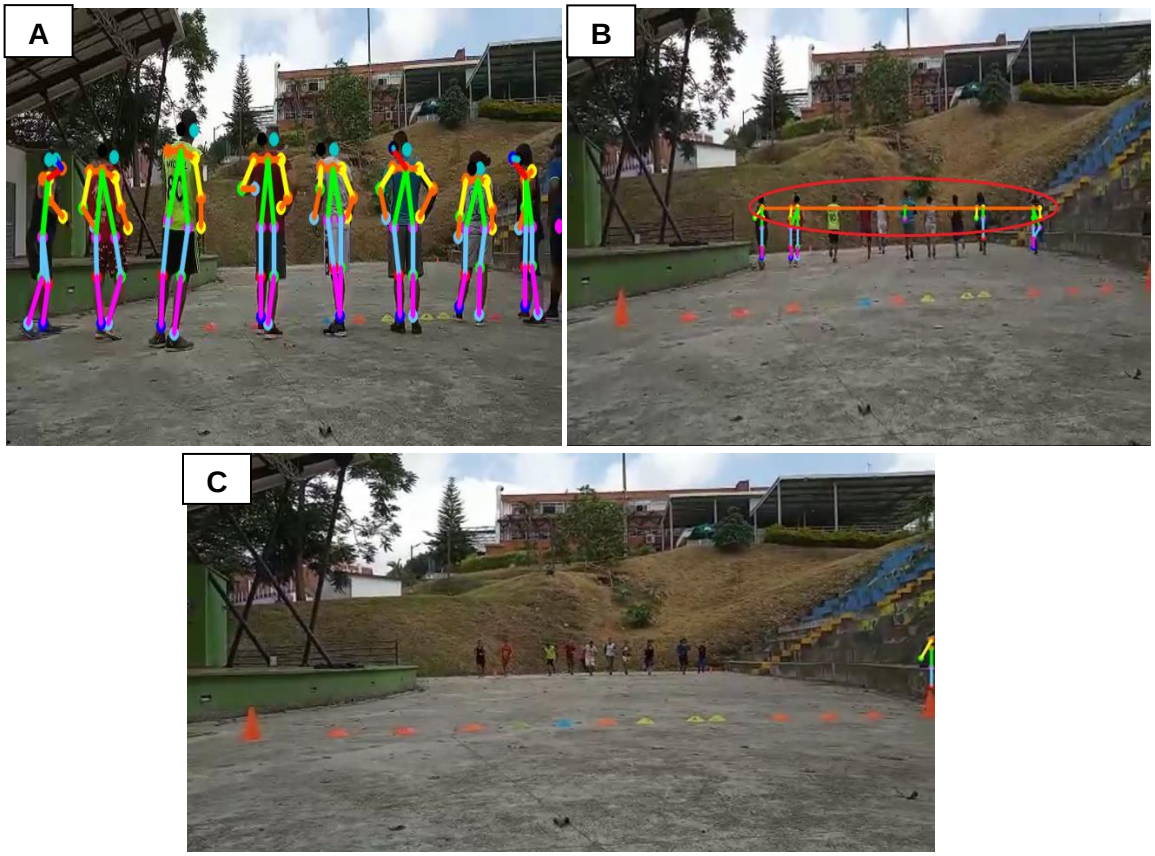
NOTA: En el Anexo A del documento se encuentra una guía para acceder al documento *Explicación Códigos*, el cual cuenta con la explicación completa y detallada de los algoritmos desarrollados para cada una de las etapas planteadas.

4.4.1. Etapa de Detección.

4.4.1.1. OpenPose. Para las pruebas realizadas con OpenPose se utilizó el dataset de COCO para la identificación de los Keypoints o puntos clave del cuerpo humano.

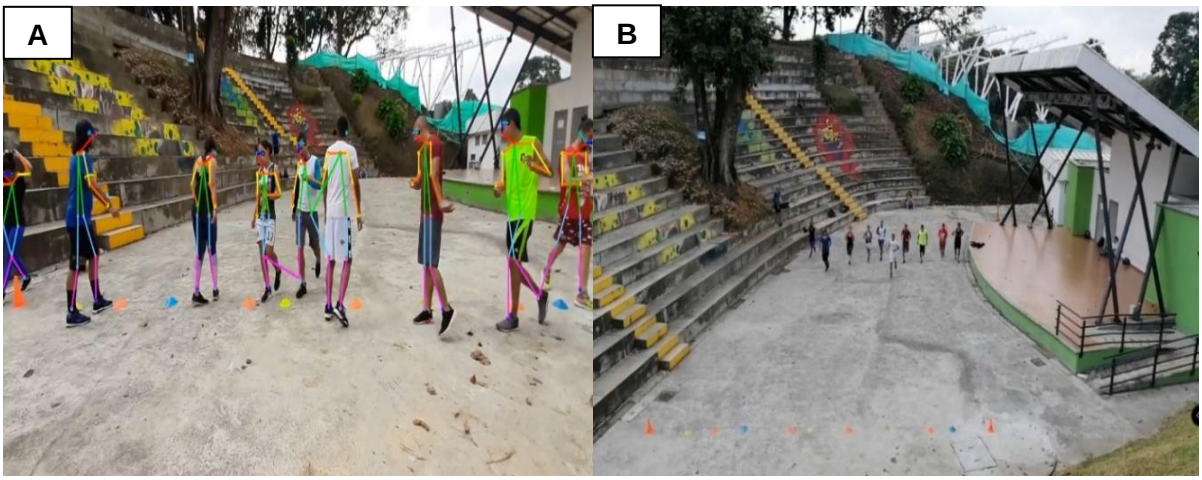
Video de Prueba N° 1.

Figura 16. Resultados de OpenPose para Video de Prueba N°1 en A) Punto Inicial, B) Punto Medio y C) Punto Final del Recorrido.



Video de Prueba N°2

Figura 17. Resultados de OpenPose para Videos de Prueba N°2.1,2.2 y 2.3.



C

Link de Youtube https://youtu.be/TRnzNK4WOCo

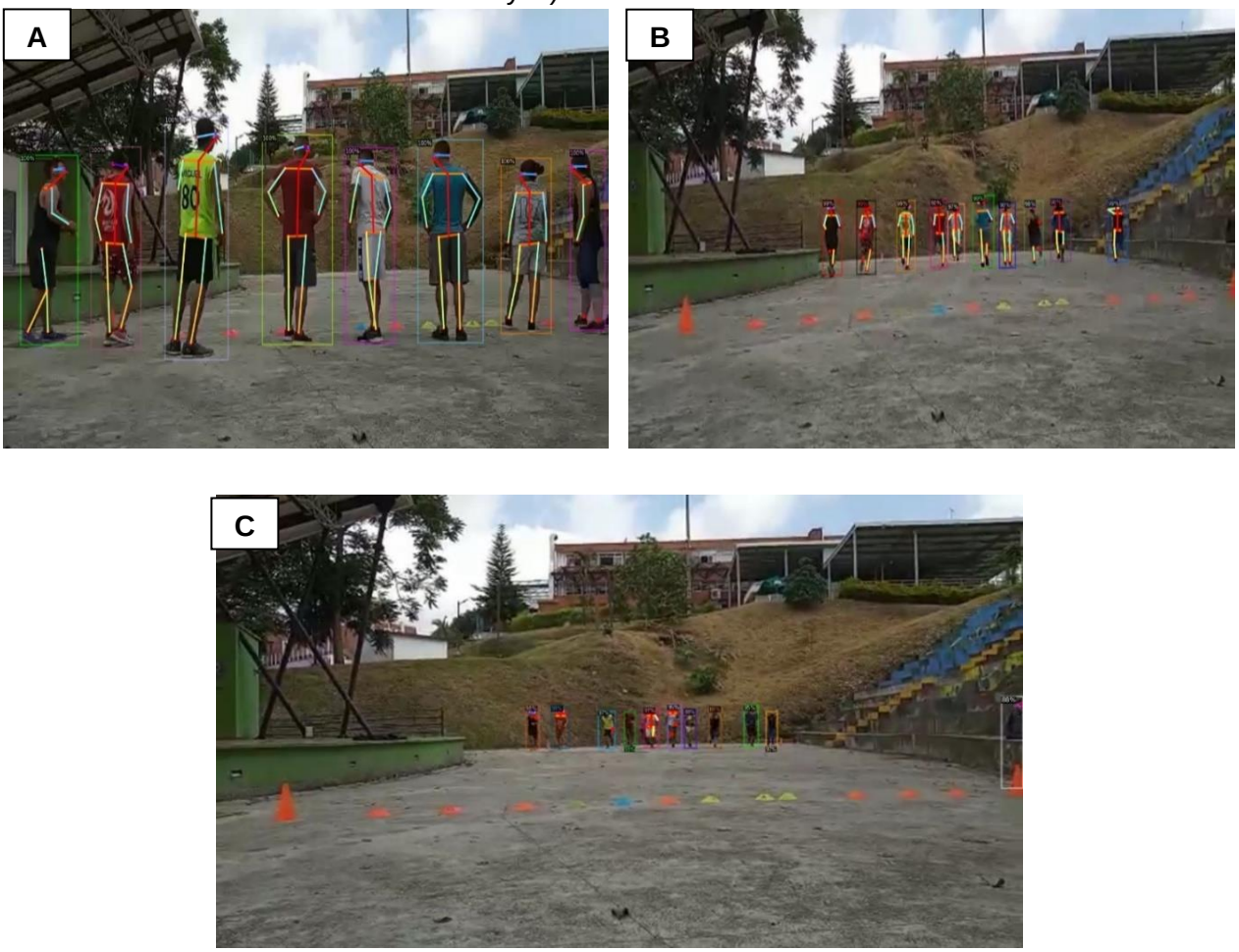
Figura 18. Resultados de OpenPose para Video de Prueba N°2.3 al variar parámetros de umbral.

Links de Youtube. A) https://youtu.be/nbsu3MhY0fk B) https://youtu.be/U4uEnCSdpCE C) https://youtu.be/SMpSd_dYbW8

4.4.1.2. Detectron2.

Video de Prueba N° .1

Figura 19. Resultados de Detectron2 para Video de Prueba N°1 en A) Punto Inicial, B) Punto Medio y C) Punto Final del Recorrido.

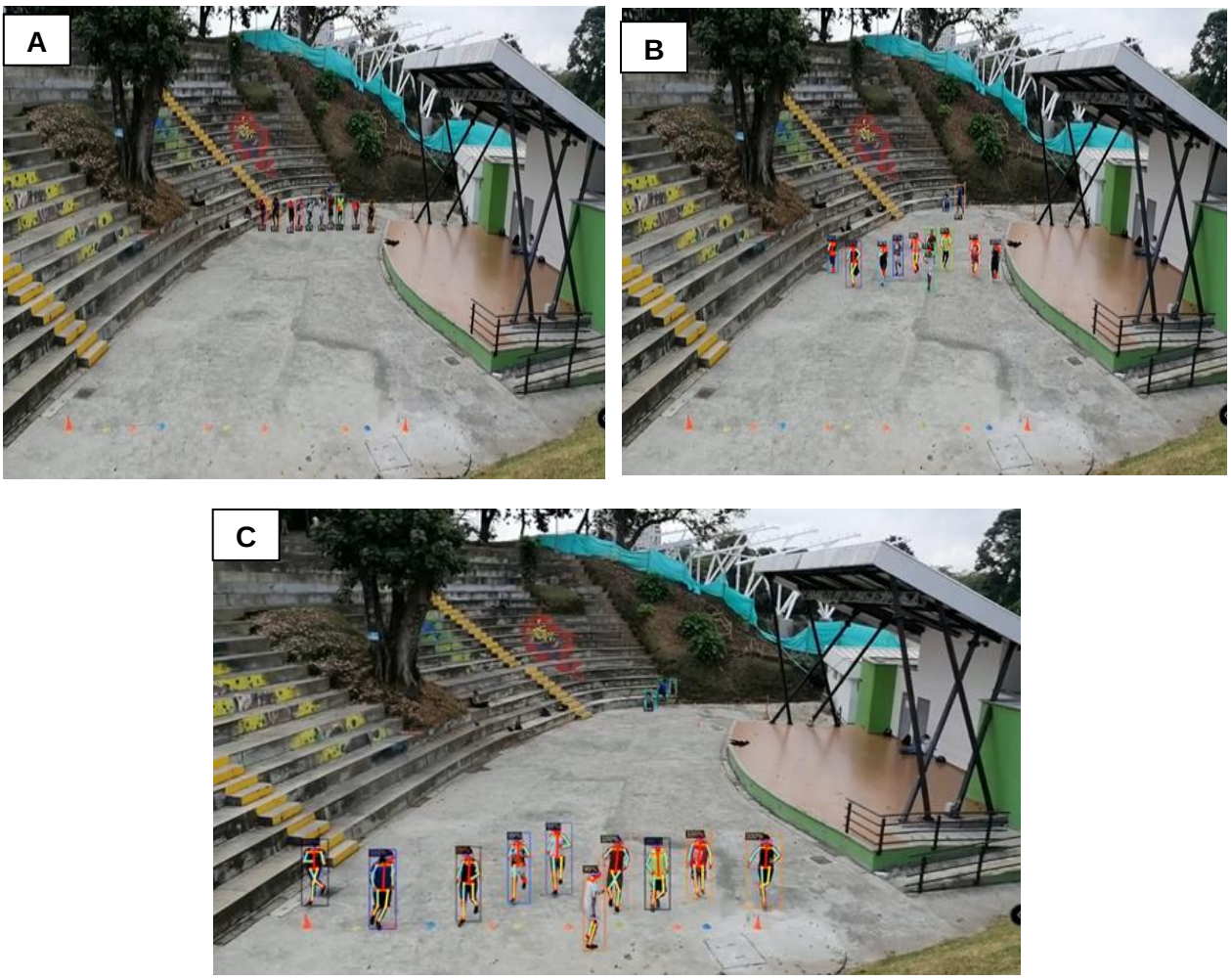


Nota: Para los resultados de Detectron2 con Video de Prueba N°1 no se tiene video ya que las pruebas se realizaron en base a imágenes.

Video de Prueba N° 2

Para las pruebas realizadas con la serie de videos de prueba 2 se tomaron en cuenta solo aquellos resultados obtenidos con la perspectiva elevada ya que para las otras dos perspectivas los resultados fueron los mismos.

Figura 20. Resultados de Detectron2 para Video de Prueba N°2.2 en A) Punto inicial, B) Punto medio y C) Punto final del recorrido.



Link de Youtube https://youtu.be/k4JBrqgI7BQ

4.4.1.3. Análisis de Resultados de Alternativas Evaluadas – Etapa de Detección. En la Tabla N° 6 se exponen los parámetros utilizados para evaluar cuál de las dos alternativas de software de detección probadas se adaptan mejor a las necesidades del programa.

En el primer parámetro se evalúa que el programa tenga la capacidad de detectar de forma clara a cada uno de los jugadores presentes en los videos de prueba y en el segundo, que pueda de mantener la continuidad en la detección realizada a cada uno de los deportistas frame a frame.

Tabla 6. Análisis de Resultados de Alternativas para Etapa de Detección.

Alternativas Algorítmicas.	Parámetros de Evaluación	
	Detección Clara	**Detección en Profundidad**
OpenPose	Si	No
Detectron2	Si	Si

Detección Clara: Para el caso de OpenPose, se hicieron evidentes dos errores en la detección realizada por el programa, el primero corresponde a las líneas que se cruzan entre los jugadores y que se pueden apreciar en la Figura 16 B para el video de prueba N° 1 y, el segundo, se relaciona con las extensiones entre jugadores de algunos segmentos de la pose estimada que se observa en la Figura 17 C para el video de prueba N° 2. Para tratar de contrarrestar estos errores se realizaron ciertas modificaciones en los parámetros de umbral del algoritmo de OpenPose, las cuales permitieron mejorar los resultados al punto de eliminar las extensiones de segmentos entre los jugadores, tal y como se observa en la Figura 18. Por otro lado, Detectron2 no presento ningún tipo de error en su identificación.

Detección en Profundidad: Para el caso de OpenPose, en las Figuras 16 y 17, A y B, para los videos de prueba N° 1 y 2, respectivamente, se puede observar como el algoritmo pierde la detección de los jugadores una vez se alejan de la ubicación de la cámara. Por otro lado, Detectron2 arrojo resultados positivos en este aspecto ya que mantuvo la detección de los deportistas aun cuando estos llegaron a la línea final de meta, tal y como se aprecia en la Figuras 19 y 20, A y B, para los videos de prueba N° 1 y 2.

En base a los resultados obtenidos, se descartó el uso de OpenPose como algoritmo de detección ya que las modificaciones realizadas al programa no permitieron corregir o mejorar el error correspondiente a la pérdida de detección por profundidad; por esta razón y dado que Detectron2 cumplió con ambos parámetros de evaluación, se eligió implementar esta herramienta en la etapa de detección.

4.4.1.4. Explicación Técnica del Algoritmo de Detección. A continuación, se presentan las principales funciones utilizadas para desarrollar el programa de detección de acuerdo con la estructura presentada en el diagrama de Flujo de la Figura 12. Estas funciones fueron previamente introducidas en la sección 3.2.3.2 de este documento.

El algoritmo de detección empieza con la configuración básica del Detector y la inicialización del algoritmo de seguimiento a través de 1), 2) y 3).

Inicialización Parámetros del Detector.

$$1)\ cfg\ =\ get\_cfg()$$

$$2)\ predictor\ =\ DefaultPredictor(cfg)$$

Inicialización del Algoritmo de Seguimiento.

$$3)\ ct\ =\ CentroidTracker()$$

Inicio del Predictor.

Luego de esto, se da inicio al predictor configurado 4) y se define el objeto que almacena las coordenadas de los recuadros delimitadores y los keypoints 5) de las personas detectadas.

$$4)\ outputs\ =\ predictor(frame)$$

$$5)\ instances\ =\ outputs['instances']$$

Verificación de la existencia de objetos detectados.

Para verificar la existencia de objetos detectados, se crea un ciclo 6) que itera sobre el rango de dichos objetos.

$$6)\ for\ i\ in\ range\big(0, len(instances)\big)$$

Almacenamiento de las Coordenadas de los Objetos Detectados.

Dentro de ese ciclo, y a partir del objeto de instances, se obtienen las coordenadas de los recuadros a través del campo pred_boxes, ya que el diccionario de outputs devuelve estos valores en forma de tensor, se convierten a un array numpy 7), seguido a esto, en 8), se itera sobre dichas coordenadas y se almacenan en la variable Box para luego realizar el cambio de tipo de datos a entero (‘int’) 9).

$$7)\ boxes\ =\ instances.pred\_boxes.tensor.cpu().numpy()$$

$$8)\ box\ =\ boxes[i]$$

$$9)\ rects.append\big(box.astype('int')\big)$$

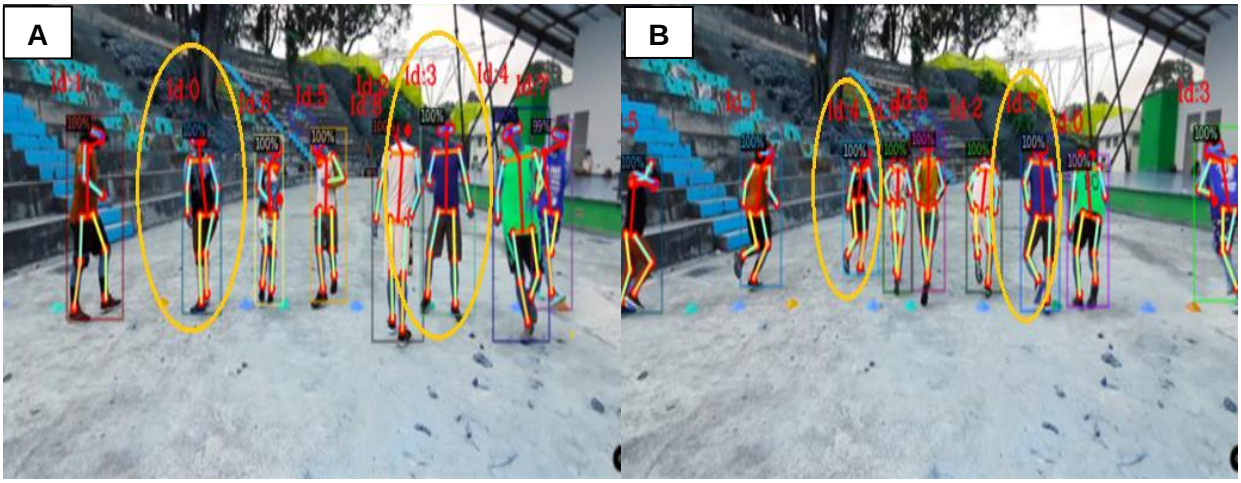
Actualización de Coordenadas para Seguimiento.

Finalmente, a través de la invocación del algoritmo de seguimiento, se utiliza el método update del CentroidTracker para actualizar el seguidor con las coordenadas de los recuadros de cada frame.

$$10)\ objects\ =\ ct.update(rects)$$

4.4.2. Etapa de Seguimiento. La primera prueba realizada en esta etapa consistió en asignar identificadores únicos a las coordenadas de los recuadros delimitadores dibujados por Detectron2, para luego, ejecutar el seguimiento durante frames consecutivos del video a dichos identificadores. Los resultados obtenidos se observan en la Figura 21.

Figura 21. Resultados de Detectron2 con Video de Prueba N° 2.1 en Etapa de Seguimiento para A) Punto 1, B) Punto 2.



Link de Youtube https://youtu.be/nQNUwU-Et0M

Según lo observado en la Figura 21, los identificadores asignados a los deportistas no se mantienen de un frame del video al otro, por ejemplo, la persona con el identificador número cero en la Figura 21A, pasa a tener el identificador número cuatro en la Figura 21B, de igual forma, la persona con el identificador número tres en la Figura 21A, pasa a tener el identificador número siete en la Figura 21B; estos resultados demuestran que el algoritmo de Detectron2 es incapaz, por sí solo, de realizar un seguimiento individual optimo a cada jugador. En base a esto, se hizo evidente la necesidad de implementar un algoritmo de seguimiento, complementario a Detectron2, que fuera capaz de identificar a los deportistas durante todo el video y bajo el mismo número de identificación asignado a cada uno en un principio.

Nota: Para los resultados de GOTURN y ORB con el Video de Prueba N°1 no se tienen links de videos ya que las pruebas se realizaron en base a imágenes.

4.4.2.1. GOTURN.

Video de Prueba N° 1

Para efectos de demostración de eficacia, el algoritmo se ejecutó para hacer el seguimiento de una sola persona.

Figura 22. Resultados de GOTURN con Video de Prueba N°1 en Etapa de seguimiento para A) Punto 1 y B) Punto 2.

4.4.2.2. ORB.

Video de Prueba N° 1.

Figura 23. Resultados de ORB con Video de Prueba N°1.

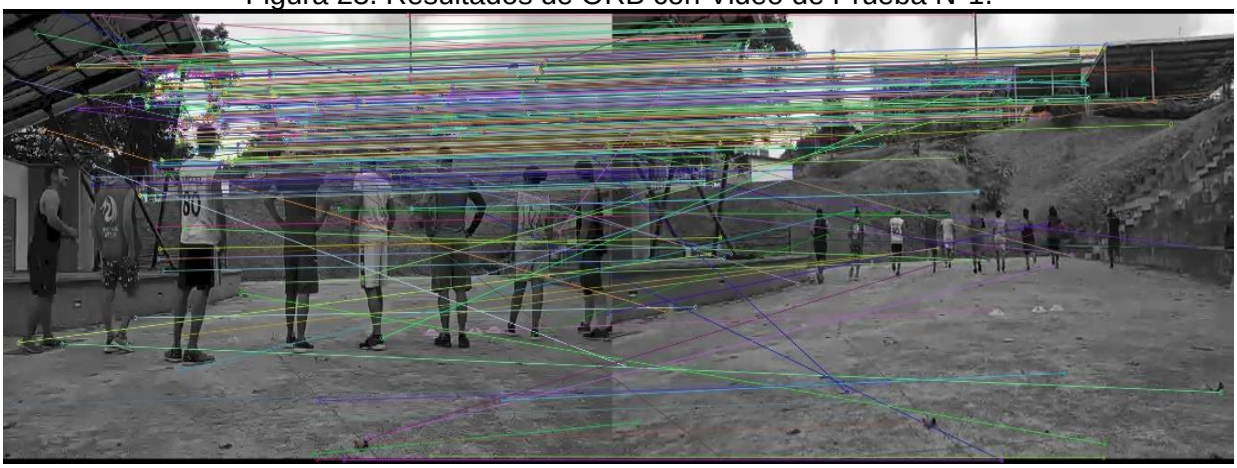
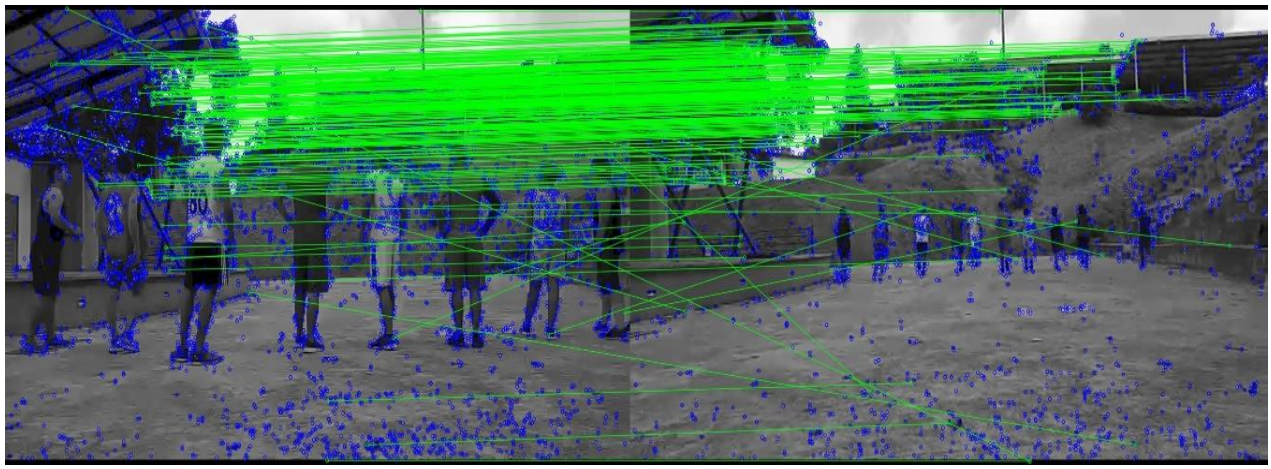


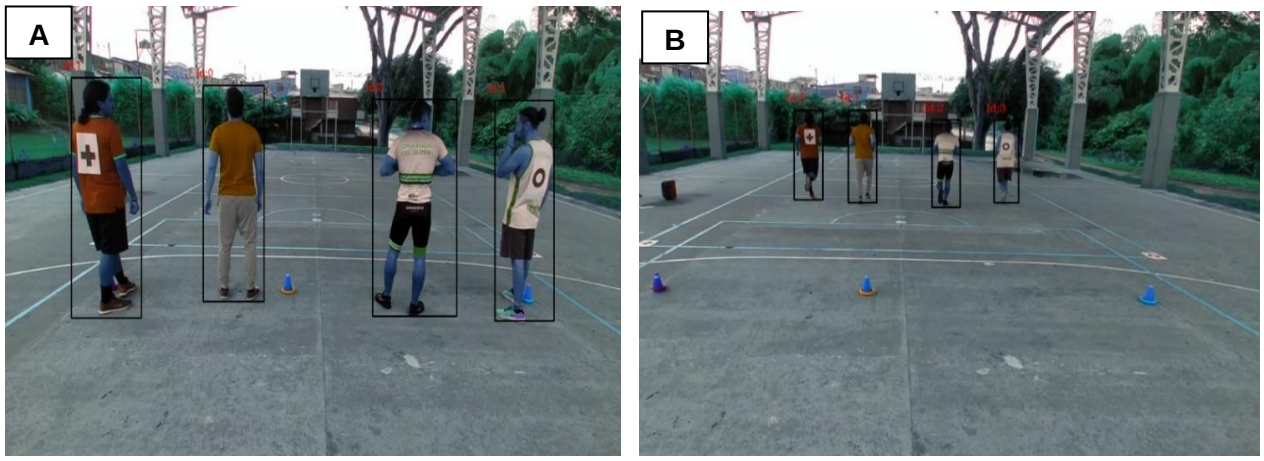
Figura 24. Resultados de ORB con Video de Prueba N°1 – Variando la Cantidad de Puntos Clave.



4.4.2.3. CentroidTracker

Video de Prueba N° 3.

Figura 25. Resultados de CentroidTracker con Video de Prueba N°3 en Etapa de Seguimiento para A) Punto Inicial, B) Punto Medio y C) Punto Final del recorrido.



Link de Youtube https://youtu.be/WvNy_YX4xW0

Los resultados obtenidos haciendo uso de Detectron2 y de CentroidTracker como las herramientas implementadas en las etapas de detección y seguimiento dependen en gran medida de la perspectiva utilizada para grabar el video y del número de jugadores presentes en el. Como se puede apreciar en las Figuras 26 y 27, al hacer uso de una perspectiva lateral, los constantes cruces entre los jugadores hacen que los identificadores se confundan, ocasionando que el algoritmo pierda el seguimiento de los deportistas.

Figura 26. Resultados de Detectron2 con CentroidTracker para Video de Prueba N°3.1 en Etapa de seguimiento Punto 1.

Link de Youtube https://youtu.be/LIzZ2uaxVKM

Figura 27. Resultados de Detectron2 con CentroidTracker para Video de Prueba N°3.1 en Etapa de seguimiento Punto 2.

En la Figura 26 el jugador con el identificador número 2 pasa a tener el identificador número 4 en la Figura 27, esto debido al cruce que hace con el jugador que está a su lado izquierdo, el cual también presenta un cambio de identificador.

4.4.2.4. Análisis de Resultados de Alternativas Evaluadas – Etapa de Seguimiento. En la Tabla N° 7 se exponen los parámetros utilizados para evaluar cuál de las tres alternativas de software de seguimiento probadas se adaptan mejor a las necesidades del programa.

En el primer parámetro se evalúa que el programa de seguimiento tenga la capacidad de identificar de forma individual a cada uno de los jugadores y de realizar su respectivo seguimiento frame a frame, y en el segundo, que pueda mantener un rastreo continuo de cada uno de los deportistas presentes en el video de prueba.

Tabla 7. Análisis de Resultados de Alternativas para Etapa de Seguimiento.

Alternativas Algorítmicas.	**Parámetros de Evaluación.**	
	Identificación/Seguimiento Individual	**Perdida de Seguimiento**
Goturn	No	Si
ORB	No	-
CentroidTracker	Si	Si

Identificación/Seguimiento Individual: Para el caso de Goturn, en la Figura 22 A se evidencia su mal funcionamiento ya que no es capaz de realizar el recuadro completo a todo el deportista, lo que resulta en datos erróneos a la hora de asignar los

identificadores; de igual forma, como se aprecia en la Figura 22 B, cuando el algoritmo presencia un cambio de frames tiene un mayor error en la asignación realizada.

El resultado que se esperaba obtener de ORB era que este algoritmo realizara un emparejamiento de los puntos clave detectados de un frame a otro y, en base a eso, hiciera el seguimiento de los objetos asociados; sin embargo, como se aprecia en la Figura 23, el seguimiento que se obtuvo fue nulo ya que el programa estaba realizando la asignación con todos los puntos detectados en el frame, incluyendo aquellos pertenecientes a la estructura física. Para tratar de mejorar este resultado, se aumentó la cantidad de puntos clave que estaba tomando el algoritmo, no obstante, y aunque es evidente en la Figura 24 que el programa empezó a emparejar algunos jugadores entre frames, el resultado no fue satisfactorio.

Por otro lado, y como se observa en la Figura 25, el CentroidTracker identifico de forma satisfactoria a cada uno de los jugadores a través de identificadores únicos que permitieron mantener el rastreo individual de los deportistas a lo largo de todo el video.

Perdida de Seguimiento: Según los resultados obtenidos, Goturn presenta un error de seguimiento ya que no mantiene la identificación realizada entre frames; por otro lado, y aunque en condiciones deseables de funcionamiento, CentroidTracker mantiene el seguimiento individual de cada uno de los jugadores con los identificadores asignados en el primer frame, se evidenciaron errores de cambio de identificador y perdida de seguimiento en situaciones que involucran cruces de deportistas (Figura 26 y 27). Para el caso de ORB no fue posible evaluar este parámetro ya que el algoritmo no fue capaz, en ningún momento, de realizar la identificación de los jugadores.

En base al análisis realizado y teniendo en cuenta las necesidades del programa se elige al CentroidTracker como la herramienta a implementar en la etapa de seguimiento.

4.4.2.5. Explicación Técnica del Algoritmo de Seguimiento. A continuación, se presentan las principales funciones utilizadas para desarrollar el programa de seguimiento de acuerdo con la estructura presentada en el diagrama de Flujo de la Figura 13. Estas funciones fueron previamente introducidas en la sección 3.2.4.3 del documento.

Cálculo del Punto de Seguimiento

El primer paso del programa es calcular el punto de seguimiento para cada uno de los recuadros detectados, para esto se toman las coordenadas que se traen del algoritmo de detección en 1) y haciendo uso de las fórmulas en 2) y 3), se calculan las coordenadas en X y Y del centroide.

$$1)\ (x, y, eX, eY) = rect$$
$$2)\ cX = ((x + eX) / 2.0)$$
$$3)\ cY = ((y + eY) / 2.0)$$

Almacenamiento de Centroides y Creación de su Identificador

Los centroides calculados y los identificadores asignados se almacenan en la variable Centroids y ObjectID.

$$4)\ Centroids = (cX, cY)$$
$$5)\ objectID = [ID0, ID1, ID2 \dots]$$

Registro y Eliminación de Centroides

A través de los métodos register y deregister se realiza la asignación de identificadores a los nuevos objetos detectados y la eliminación de estos mismos para los objetos que desaparecen del escenario, respectivamente.

$$6)\ register(centroid, rect)$$
$$7)\ deregister(objectID)$$

Seguimiento de Centroides

Para el seguimiento de los centroides se calcula la distancia euclidiana entre los elementos del frame actual y del frame anterior.

$$8)\ D = dist.cdist(np.array(objectCentroids), inputCentroids)$$

Actualización de Coordenadas

Finalmente, el algoritmo retorna dos diccionarios, el primero contiene las coordenadas de los centroides y los identificadores, y el segundo las coordenadas de los recuadros y los identificadores.

$$9)\ return\ objects, originRects$$

4.4.3. Análisis de Selección de Arreglo de Cámaras y Perspectiva de Grabación. En esta sección, se presenta el análisis realizado para la selección del arreglo de cámaras y perspectiva de grabación a utilizar, en la obtención de los videos a evaluar con el algoritmo de automatización desarrollado. Se eligió introducir esta información antes de terminar de exponer los resultados obtenidos para todas las etapas de desarrollo del algoritmo, ya que las opciones de alternativas escogidas en esta sección fueron las que se utilizaron para el desarrollo de la etapa de evaluación.

4.4.3.1. Análisis de Selección de Alternativa – Número y Tipo de Cámaras. En la Tabla N°8 se realiza el análisis de selección del tipo de cámara a utilizar para la grabación de la ejecución del Test.

Uno de los principales criterios dentro del desarrollo del programa es que los videos puedan ser grabados durante las pruebas realizadas, por las personas que las ejecutan, y haciendo uso de los medios que tengan a su disposición; en base a esto se definieron los dos primeros parámetros de la Tabla N°8, el tercer parámetro evalúa la complejidad de uso del dispositivo en la medida que requiera de configuraciones especiales para su correcto funcionamiento.

Tabla 8. Análisis de Selección de Alternativa – Tipos de Cámaras.

Tipos de Cámaras	**Parámetros de Evaluación**		
	Calidad de Video	**Disponibilidad de Recursos del Usuario**	**Configuraciones Técnicas y de Uso**
Dispositivo Móvil	Alta	Si	No
Cámara Profesional de Video	Alta	Depende de las condiciones económicas del usuario.	Depende del tipo de cámara profesional utilizada.

En lo que respecta a los dos primeros parámetros de evaluación, las cámaras de los celulares han ido evolucionando en los últimos años a tal punto de que, en la actualidad, presentan características técnicas que igualan o incluso sobrepasan las prestaciones que se pueden obtener con una cámara profesional de video, a esto se suma también que los precios de adquisición de estos dispositivos son mucho más asequibles para las personas del común, a comparación con los precios de la cámara profesional. Por otro lado, contrario a la cámara de un dispositivo móvil, que no necesita de ningún tipo de configuración especial para su utilización, las cámaras profesionales suelen requerir de la utilización de baterías de larga duración y de elementos de estabilización y de transporte para la misma.

De acuerdo con el análisis realizado en la Tabla N°8, y explicado a anteriormente, se concluyó que la mejor opción de grabación era el dispositivo móvil.

Paralelo a la selección del tipo de cámara, se evaluó el número de dispositivos a utilizar para la grabación; en la Tabla N° 9 se realiza el análisis de selección correspondiente a este aspecto. Los parámetros de evaluación utilizados en esta sección hacen referencia a la disponibilidad de recursos por parte del usuario final del programa, es decir, las personas que ejecutan el test, y a las configuraciones técnicas que conllevan el número de cámaras a utilizar.

Tabla 9. Análisis de Selección de Alternativa – Número de Cámaras.

Número de Cámaras	Parámetros de Evaluación	
	Disponibilidad de Recursos del Usuario	Configuraciones Técnicas
1 Cámara	Si	No
2 o más Cámaras	Depende de las condiciones económicas del usuario.	Si

Como es de suponer, es más factible que un jugador o entrenador posea un dispositivo móvil apto para la grabación de videos a que tenga a su disposición un esquema o estructura de múltiples cámaras. De igual forma, la utilización de una sola cámara facilita la captura y análisis de los videos ya que evita configuraciones relacionadas con la sincronización de las capturas realizadas.

La elección de alternativa realizada para el número y tipo de cámaras se refuerza en los resultados obtenidos y presentados en este capítulo, los cuales fueron logrados a partir de grabaciones hechas con un solo dispositivo móvil, que cambió de especificaciones técnicas, tales como la marca, resolución y frames por segundo, entre otras, a lo largo de la realización de los videos; durante estos cambios, fue posible advertir que aspectos como la calidad de grabación, no afectaban en gran medida los resultados obtenidos, mientras que la altura y la ubicación de las cámaras, presentaban las mismas dificultades de implementación independientemente del tipo de cámara utilizada.

4.4.3.2. Análisis de Selección de Alternativa – Perspectiva de Grabación. En la Tabla N° 10 se realiza el análisis de selección de perspectiva de grabación para definir cuál de las tres perspectivas evaluadas arroja mejores resultados en el funcionamiento del programa. El primer parámetro evalúa que la perspectiva de grabación permita realizar una identificación clara y un seguimiento constante a los jugadores que se encuentran desarrollando el test, mientras que el segundo, evalúa que la perspectiva de grabación permita identificar de forma clara las líneas de meta en las cuales se realiza el análisis de los cruces de los jugadores, a partir del cual, se define si el deportista aprobó o no el tramo correspondiente.

Tabla 10. Análisis de Selección de Alternativa – Perspectiva de Grabación.

Perspectiva de Grabación	Parámetros de Evaluación	
	Identificación y Seguimiento Constante	Identificación de Cruces en Líneas de Meta
Frontal	Si	No
Lateral	No	-
Frontal-Elevada	Si	Si

Identificación y Seguimiento Constante: Como se pudo observar en las Figuras 25 y 20, las perspectivas frontal y elevada permiten realizar una identificación clara y, por ende, un seguimiento constante de los deportistas presentes en el video de evaluación, siempre y cuando los cruces entre ellos sean mínimos; por otro lado, la perspectiva lateral, tal y como se discutió para las Figuras 26 y 27, hace que los cruces entre los jugadores sean más evidentes, lo que limita la capacidad del algoritmo de seguimiento de realizar un rastreo efectivo.

Identificación de Cruces en Líneas de Meta: Para la perspectiva frontal, y en base al análisis del escenario deportivo del video de prueba N°3 que se observa en la Figura 25, se evidencia que desde este ángulo y al tener la cámara ubicada frente a los jugadores y a su mismo nivel de altura es inviable identificar el momento en que los deportistas cruzan la segunda línea de meta, es decir, aquella que se encuentra al fondo del video, esto debido a que el análisis que se está realizando es en base a una imagen de dos dimensiones y el parámetro que se requiere identificar involucra un análisis basado en la profundidad de la imagen.

Para la perspectiva frontal - elevada, y tomando en cuenta los resultados obtenidos en la Figura 20, se hace evidente que este ángulo de grabación permite identificar ambas líneas de meta en el video de prueba, lo que a su vez facilita el análisis de cruces de los deportistas. La perspectiva lateral no fue evaluada en este parámetro ya que, al no cumplir con la identificación y el seguimiento constante de los jugadores, se descartó la realización de más pruebas desde esta ubicación.

En este punto, no es posible definir si existe o no una altura específica para la ubicación de la cámara que permita obtener mejores o peores resultados, en cuanto a la identificación de cruces en las líneas de meta, ya que las pruebas realizadas con cámara elevada, en donde se verifica el cumplimiento de este requerimiento, se hicieron únicamente a una altura de tres metros.

4.4.3.3. Análisis de Selección de Alternativa – N° de Jugadores a Evaluar. El último parámetro para evaluar en cuanto a las condiciones de grabación y evaluación del Test fue el número de jugadores a valorar por video. En la tabla N° 10 se expone el análisis de selección correspondiente. El único parámetro de evaluación utilizado evalúa que la cantidad de jugadores presentes en el video de prueba no interfiera con la capacidad de los algoritmos de detección y seguimiento implementados, de realizar la identificación y el seguimiento constante de los jugadores.

Tabla 11. Análisis de Selección de Alternativa – Número de Jugadores a Evaluar.

Número de Jugadores	**Parámetros de Evaluación**
	Identificación y Seguimiento Constante
Más de 4	No
Igual a 4 o menos	Si

Identificación y Seguimiento Constante: Como se observó en la Figura 21, para el video de prueba N° 2.1 con Detectron2, al tener más de 4 jugadores presentes en el video, los constantes cruces entre ellos hacen que el algoritmo de seguimiento empiece a confundir los identificadores asignados a cada uno, lo que a su vez ocasiona que el rastreo se pierda. En cambio, cuando se tienen 4 o menos deportistas en el video de prueba (Figura 25), los cruces entre jugadores son mínimos, por lo que el seguimiento es más efectivo.

En base a las evaluaciones realizadas para la selección de tipo y número de cámaras, perspectiva de grabación y número de jugadores presentes por video, se definió lo siguiente en cuanto a las condiciones de grabación del test para la etapa final de desarrollo, así como para futuras implementaciones del algoritmo: Las grabaciones del Test de Course – Navette se realizarán en base a una perspectiva de grabación frontal y elevada al escenario deportivo y podrán tener un máximo de 4 jugadores para evaluar por video. El tipo de cámara a utilizar se deja a decisión del usuario, sin embargo, y en base al análisis presentado anteriormente, quedo demostrado que es posible obtener una grabación completa de la ejecución del Test haciendo uso de un solo dispositivo móvil con cámara de grabación.

4.4.4. Etapa de Evaluación. A continuación, se presentan los resultados obtenidos al implementar las subetapas planteadas en el diagrama de Flujo de la Figura 14, haciendo uso de la alternativa correspondiente a la transformación de perspectiva, mencionada en la Tabla N°4.

Con respecto a la alternativa de estimación de distancias en el video, la decisión de no implementarla en esta etapa se debió a que, en primer lugar, y en base a las conclusiones expuestas en los artículos mencionados en la sección de análisis de alternativas algorítmicas para la etapa de evaluación, este método tiene una tasa de error considerable ya que para la estimación automática de la distancia requiere, por una parte, que el usuario ingrese la altura de la cámara de grabación, y que, adicional a esto, el terreno en donde se realicen las pruebas sea completamente plano. A diferencia de este método, las pruebas realizadas con la transformación de perspectiva permitieron obtener los resultados buscados a partir de la selección manual de la región de interés, la cual se realiza, en base a la previa demarcación de esta en el escenario por parte de los

deportistas; dado este resultado, se optó por la alternativa más simple de implementar, es decir, la transformación de perspectiva.

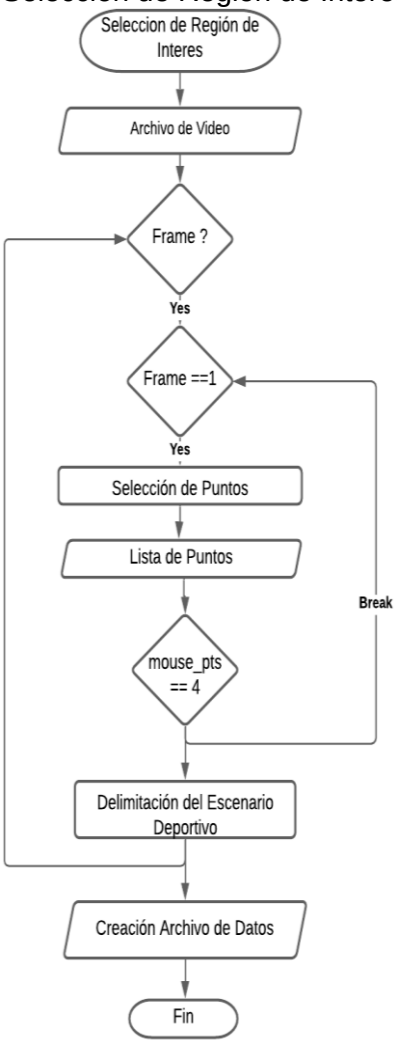
Para cada una de las subetapas se presenta el diagrama de flujo y la explicación correspondiente a su funcionamiento. Todas las funciones mencionadas en esta sección fueron previamente introducidas en la sección 3.2.5 del documento.

4.4.4.1. Selección de Región de Interés. La primera parte de la etapa del algoritmo de evaluación consiste en la obtención del escenario del test, es decir, sus límites, los cuales están comprendidos por una línea de meta inicial y final. En el diagrama de flujo de la Figura 29 se resume el funcionamiento del algoritmo y en la Figura 28 A y B se ilustra el resultado de la selección realizada.

Figura 28. Selección de Puntos de la Región de Interés

A B

Figura 29. Diagrama de Flujo – Selección de Región de Interés.



Selección de Puntos

Una vez se inicia la lectura del video de prueba a través del ciclo while, el algoritmo procede a seleccionar los cuatro puntos que demarcan la región de interés del escenario deportivo, a través de 1).

$$1)\ get\_mouse\_points(event, x, y)$$

Los parámetros de entrada de esta función son: event, que corresponde a la acción del mouse, y las coordenadas X, Y de los puntos seleccionados con este. Esta línea también se encarga de registrar y almacenar cada uno de los puntos seleccionados.

Delimitación del Escenario Deportivo

En el momento en que se han seleccionado los 4 puntos de la región, el ciclo de evaluación para el primer frame se rompe y el algoritmo dibuja sobre este, mediante la función cv2.Polylines 3), un rectángulo con los puntos seleccionados, que simula los límites del escenario.

$$3)\ cv2.polylines(frame, [mouse\_pts])$$

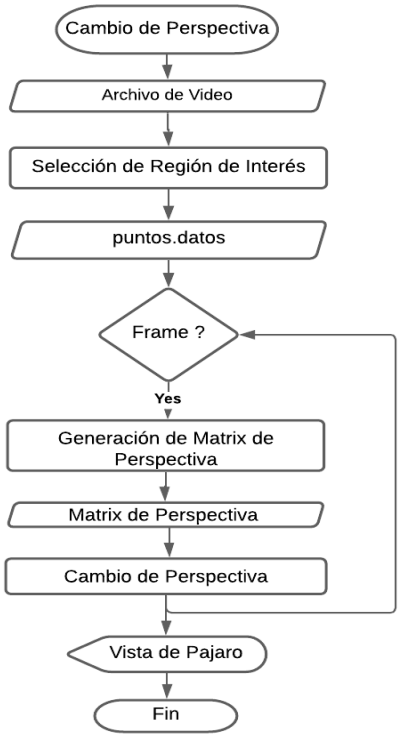
Creación Archivo de Datos

Finalmente, cuando se cierra el ciclo, la función open 4), genera un archivo de datos con la información de los puntos de la región de interés que será posteriormente transformada.

$$4)\ open('puntos.data', 'wb')$$

4.4.4.2. Cambio de Perspectiva. Para tener un mejor rastreo del recorrido de los deportistas, se realizó un cambio de perspectiva de la región de interés seleccionada, que permitió enfocar el análisis del algoritmo a las coordenadas de los pies de los jugadores, de las cuales, se extrae el punto de rastreo utilizado por la etapa de seguimiento. En el diagrama de flujo de la Figura 30 se resume el funcionamiento del algoritmo, y en la Figura 31 se ilustra el resultado del cambio de perspectiva o, como comúnmente se conoce, la vista de pájaro.

Figura 30. Diagrama de Flujo – Cambio de Perspectiva.



Los parámetros de entrada del algoritmo son la ruta del video y el archivo de datos con los puntos de la región de interés seleccionados.

Generación de Matriz de Perspectiva

En cada uno de los frames leídos en el ciclo While, el algoritmo utiliza la función 1) para generar la matrix de perspectiva de la región de interés.

1) $M = cv2.getPerspectiveTransform(src, dst)$

Cambio de Perspectiva

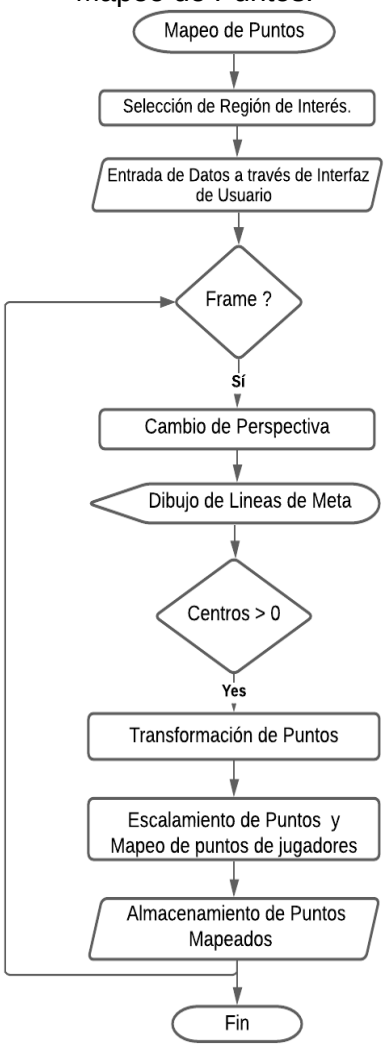
2) utiliza la matrix de transformación almacenada en la variable M y, en base a ella, realiza el cambio de perspectiva del frame actual, manteniendo el tamaño de la imagen de entrada.

2) $cv2.warpPerspective(frame, M, (IMAGE\_W, IMAGE\_H))$

Figura 31. Transformación de Perspectiva de la Región de Interés.

4.4.4.3. Mapeo de Puntos. Luego de haber implementado el cambio de perspectiva del escenario, se realizó el mapeo de los puntos de seguimiento de cada uno de los jugadores presentes en el video con el fin de poder identificar sus cruces sobre las líneas de meta. En el diagrama de flujo de la Figura 32 se resume el funcionamiento del algoritmo, y en la Figura 33 se ilustra el resultado del mapeo de puntos y del análisis de cruces.

Figura 32.Diagrama de Flujo – Mapeo de Puntos.



Una vez dentro del ciclo de lectura del video, el algoritmo crea una imagen de ceros (Imagen negra) sobre la cual se dibujan las líneas de meta del escenario deportivo, haciendo uso de 1). Esta imagen creada se observa en la Figura 33A.

Dibujo de Líneas de Meta

$$1)\ cv2.lines(frame)$$

Transformación de Puntos

Para evaluar si hay o no jugadores dentro de la región de interés se utiliza un condicional IF que se encarga de verificar el tamaño de la lista de puntos de seguimiento de los deportistas (Centros), si esta es mayor a cero significa que existen jugadores dentro del ROI seleccionado, por lo que se procede a realizar la transformación de dichos puntos a través de 2).

$$2)\ warped = cv2.warpPerspective(centros, M)$$

Escalamiento y Mapeo de Puntos Transformados.

Luego de esto, se realiza el escalamiento de los puntos transformados para su respectiva representación en la imagen de ceros generada anteriormente, tal y como se puede observar en la Figura 33 A y B.

$$3)\ Warped_{scaled} = ((warped[i][0] * \text{scale\_w}), (\text{warped}[i][1] * \text{scale\_h}))$$

$$4)\ cv2.circle(frame)$$

En la línea 3 se multiplican los puntos transformados por los factores de escala (scale_w, scale_h), los cuales se dejaron iguales a 1 para que la imagen transformada tenga el mismo tamaño que aquella utilizada en la obtención de la matrix M. Con la función 4) se dibujan los puntos sobre la imagen de ceros.

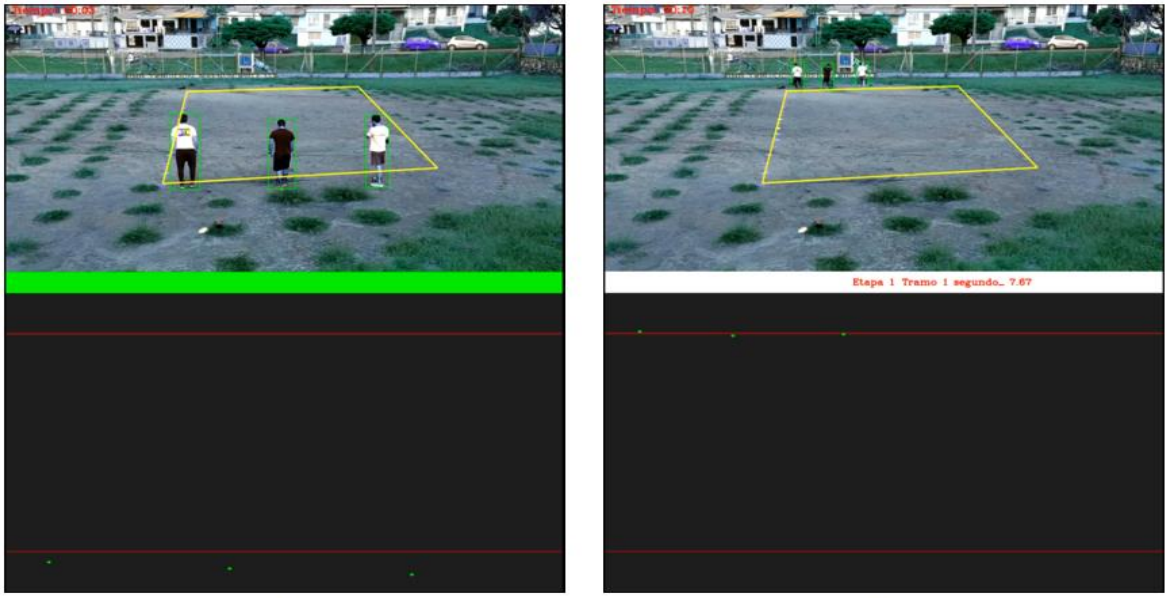
Almacenamiento de Puntos Mapeados

Los puntos mapeados se almacenan en la lista que se utiliza para el análisis de cruces con las líneas de meta, este análisis se realiza en base a la comparación de las coordenadas de los puntos y las coordenadas de las líneas de meta.

$$5)\ lista\_corredores.append(puntos)$$

Como se puede observar en la Figura 33A, a medida que se emite una señal sonora, el programa se encarga de generar una franja verde ilustrada en la parte inferior del frame del video, en el momento en que esto sucede, el programa toma las coordenadas de los jugadores para evaluar si alcanzaron a cruzar las líneas de meta a tiempo con la señal sonora; cabe resaltar que esta demarcación solo será reflejada en el momento en que se emita la señal sonora, durante el resto del video, dicha franja reflejará el número de la etapa, el tramo y los segundos de la etapa que está en curso, logrando tener así un control minuto a minuto de la ejecución del test.

Figura 33. Mapeo de Puntos en la Región de Interés.



A B

Link de Youtube https://youtu.be/UwYjoG6zTZ4

4.4.5. Interfaz de Usuario. A continuación, se enuncian los principales componentes de la interfaz de usuario del programa, en el siguiente enlace se encuentra un video con la explicación de uso de esta.

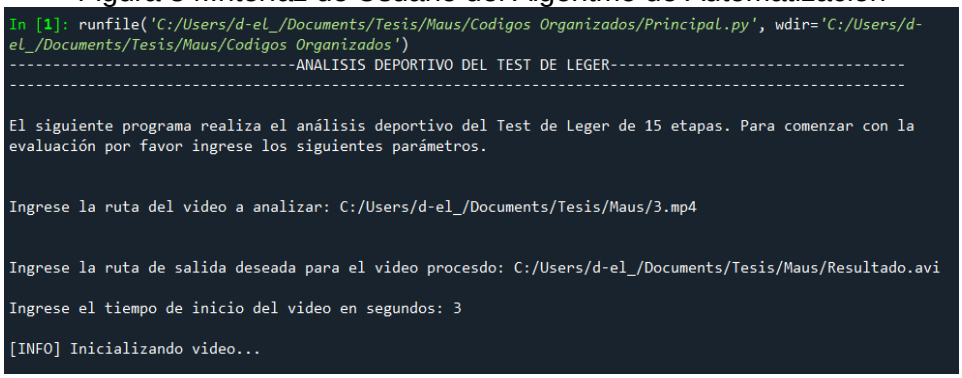
4.4.5.1. Selección de Región de Interés. La selección de la región de interés, como se mencionó anteriormente, se realiza en un código complementario al programa principal, el cual permite realizar la selección de puntos que se observa en la Figura 28, en base al siguiente criterio de orden:

- Punto1: abajo a la izquierda.

- Punto2: abajo a la derecha.
- Punto3: arriba a la izquierda.
- Punto4: arriba a la derecha.

4.4.5.2. Entrada de Datos. Para facilitar la interacción del usuario con el programa se implementaron tres entradas de datos por consola dentro del programa principal, la primera corresponde a la ruta donde se encuentra alojado el video a ser procesado, la segunda es la ruta donde el usuario va a guardar la salida del video ya procesado y la última entrada corresponde al tiempo de inicio en segundos de la prueba, tal y como se observa en la Figura 34. Esta interfaz permite tener un mejor control a la hora de ejecutar el algoritmo, ya que evita que se tengan que realizar cambios estructurales de código, tales como la modificación de variables para la lectura del video a analizar.

Figura 34.Interfaz de Usuario del Algoritmo de Automatización



4.4.5.3. Visualización de Resultados. Una vez se empiece la ejecución del algoritmo para el test de Course Navette, la interfaz implementada se encarga de ir imprimiendo por consola el identificador del jugador y la condición de aprobación y desaprobación del tramo correspondiente, tal y como se ilustra en la Figura 36, paralelo a esto, se tiene la representación gráfica del video que se está procesando y que se puede observar en la Figura 33.

De igual forma, el algoritmo genera archivos “.txt” con el nombre de cada jugador, a medida que se va realizando el análisis de las posiciones; dichos archivos van almacenando las evaluaciones correspondientes a cada tramo realizado, es decir, se imprime la etapa, el tramo, la condición de aprobación o desaprobación y la capacidad pulmonar calculada, todo esto de forma independiente para cada uno de los deportistas, tal y como se observa en la Figura 36 A, B y C.

Para el caso en el que el jugador no apruebe la etapa, se imprime el número de la etapa y el tramo en que sucede la advertencia y se muestra la condición de desaprobación, tal y como se ilustra en la Figura 36C. También existe la posibilidad de que el deportista sea descalificado luego de dos advertencias por incumplimiento en la llegada a las líneas de meta, para este caso, se imprime la notificación de que el jugador fue descalificado y se informa el número de etapa y tramo en que sucedió la penalización, así como la capacidad pulmonar alcanzada. Estos archivos se guardan automáticamente en la ruta donde se encuentra alojado el código de la aplicación, como se observa en la Figura 36D.

Figura 35. Visualización de Resultados por Consola

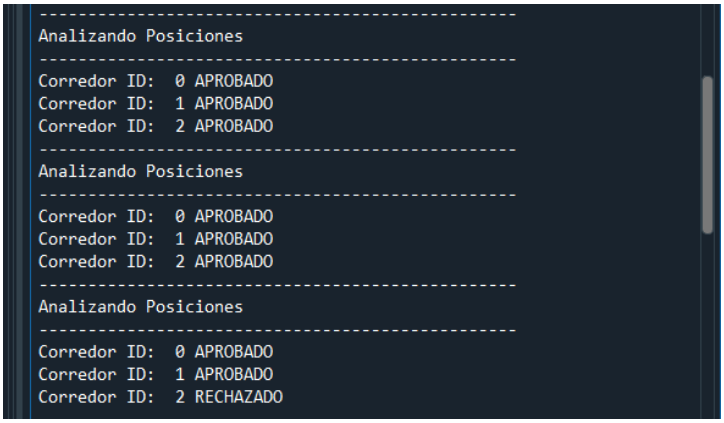
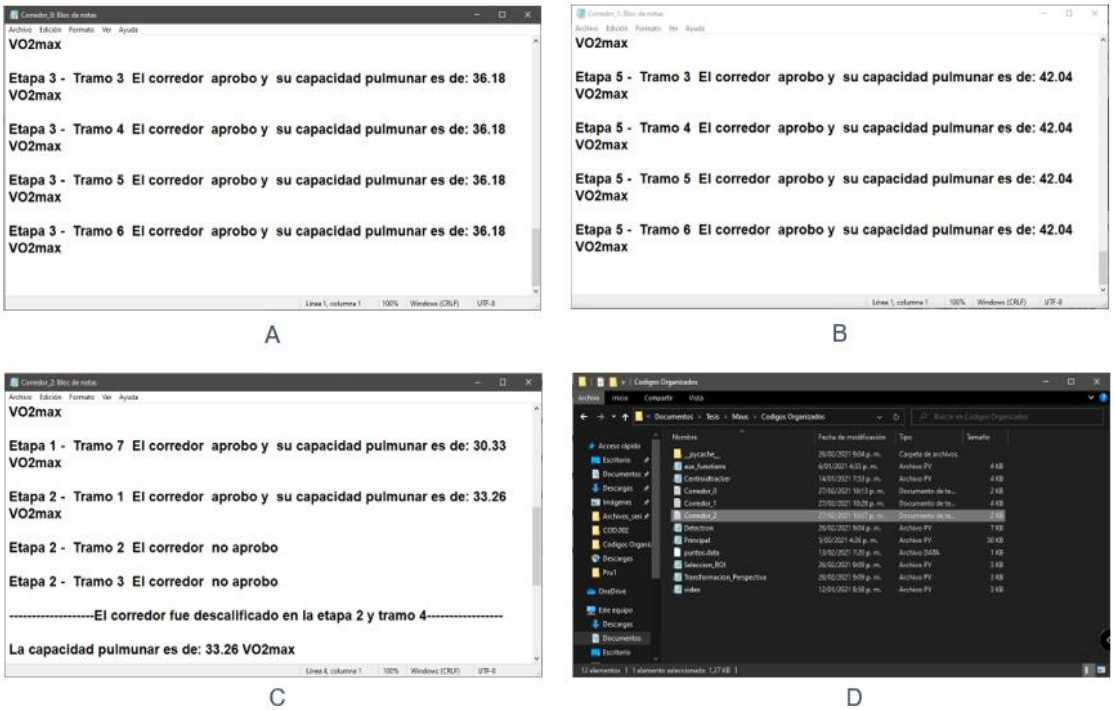


Figura 36. Archivos de Salida del Algoritmo.



4.4.6. Exploración de Hardware y Creación del Entorno de Desarrollo. Como se mencionó al inicio de este capítulo, de acuerdo con lo planteado en la sección de alcances y limitaciones del anteproyecto de este trabajo, el algoritmo de automatización se diseñó para trabajar en base a videos pregrabados de la ejecución del test, debido al alto costo computacional que representaba un procesamiento en tiempo real. Durante las primeras fases de pruebas realizadas, se evaluaron diferentes alternativas de hardware que permitieran que este procesamiento, fuera lo más rápido y efectivo posible, en la Tabla N°12 se exponen las tres alternativas analizadas, sus especificaciones técnicas y el tiempo de procesamiento obtenido con cada una de ellas, los valores expuestos para este último parámetro son el resultado del procesamiento del video de prueba N° 3, cuyas características se encuentran en la Tabla N° 5.

Tabla 12. Análisis de Selección de Alternativa de Hardware.

Alternativas de Hardware	Especificaciones Técnicas		Tiempo de Procesamiento
Equipo de Cómputo 1	CPU Clock	Intel Pentium 2.6GHz	4 - 6 Días
	GPU	No	
	RAM	5GB	
	Sistema Operativo	Linux	
Google Colab	CPU Clock	Intel Xeon 2.3GHz	3 - 5 Horas
	GPU	NVIDIA Tesla K80	
	RAM	12GB	
	Sistema Operativo	Linux	
Equipo de Cómputo 2	CPU Clock	Ryzen 3600 3.6GHz	1 - 2 Horas
	GPU	NVIDIA GTX 1650S	
	RAM	16GB	
	Sistema Operativo	Windows	

Las primeras pruebas para el análisis de selección de alternativa de hardware se realizaron en el equipo de cómputo 1, el cual era el que se encontraba a disposición para el desarrollo del proyecto, sin embargo, y de acuerdo con los resultados expuestos en la Tabla N°12, el elevado tiempo de ejecución obtenido con él, hizo que fuera necesario buscar una alternativa que permitiera llevar a cabo este proceso de forma más rápida

Indagando sobre herramientas libres disponibles en la web, se llegó a Google Colab[61], un servicio de GPU de Google que se encuentra alojado en la nube, y que se ha vuelto

[61] S. Diamantas, S. Astaras, and A. Pnevmatikakis, “Depth Estimation in Still Images and Videos Using a Motionless Monocular Camera,” IST 2016 - 2016 IEEE Int. Conf. Imaging Syst. Tech. Proc., (marzo 2018), pp. 129–134.

bastante común para la investigación en las áreas del Deep Learning, esta herramienta permitió disminuir de gran forma el tiempo de procesamiento de los videos, no obstante, y dado que se trata de un servicio en la Web, se requiere de una conexión estable y continua a internet para no tener problemas durante la ejecución del programa. En base a este resultado, se decidió invertir en un equipo de cómputo con mejores especificaciones técnicas que permitiera mejorar los resultados obtenidos con Google Colab.

Con respecto a la creación del entorno de desarrollo, en el Anexo A del documento se encuentra una guía para acceder al documento *Instalación de Software*, el cual contiene la explicación detallada de la creación e instalación del entorno de ejecución y, la solución de los posibles errores que pueden surgir durante este proceso.

5. ANÁLISIS DE RESULTADOS

5.1. Limitaciones del Algoritmo

Durante las pruebas realizadas con el algoritmo creado se hicieron evidentes ciertas limitaciones para tener en cuenta a la hora de ejecutar el programa, estas fueron:

1. Interrupción de Ejecución por Cambio de Identificadores.

   Como se había mencionado previamente en la sección 4.4.3.2, con el fin de evitar el error correspondiente a la perdida de seguimiento por oclusión o cruce de jugadores, se había planteado el requerimiento de grabar la ejecución del test únicamente desde una perspectiva frontal, sin embargo, al analizar el video de prueba N° 4, se evidencio una vez más el problema con el cruce de los deportistas, tal y como se observa en la Figura 37, en la que se aprecia como el jugador identificado con el número 1 en la imagen A, cambia al identificador número 55 en la imagen B, debido al cruce que hace con el sujeto que se encuentra al fondo del video, el cual corresponde a una persona externa a la ejecución del test.

   Figura 37. Cambio de Identificadores por Cruce de Personas.

   A B

   El cambio de identificadores provoca que el programa termine su ejecución de forma precipitada, a causa de que no encuentra el nuevo identificador dentro de una serie de listas que contienen los posibles estados de Aprobación/Desaprobación para los jugadores que se encuentran realizando el test. Dentro de las posibles alternativas de solución para este error se encuentran la detección y clasificación de personas por color[62] y la utilización de algoritmos de

[62] F. S. Khan, R. M. Anwer, J. Van De Weijer, A. D. Bagdanov, M. Vanrell, and A. M. Lopez, “Color Attributes for Object Detection,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., (2012) p. 3306–3313.

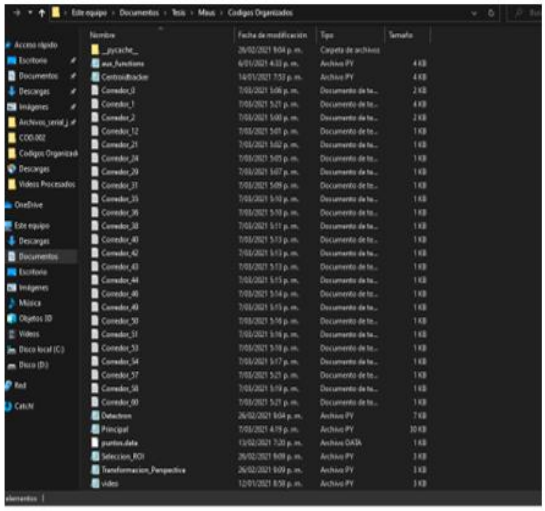
seguimiento más robustos, tales como PoseFlow[63] o KeyTrack[64], sin embargo, y tal y como lo comprobó Janssen en[65], estos métodos no pueden modelar con precisión las relaciones temporales que se requieren para realizar la identificación y el seguimiento de las instancias de pose humana, y requieren, además, de un cálculo computacional significativo. Debido a esto, y ya que, para el caso específico de este proyecto, el error puede ser evitado controlando que no existan cruces entre jugadores o con agentes externos al ambiente deportivo, se decidió no indagar más en la solución específica del problema.

2. Creación de Múltiples Identificadores por Sujetos Externos al Test.

   La segunda limitación que se encontró durante la ejecución del algoritmo y que está relacionada con el tema de los agentes externos, mencionado en el punto anterior, es que se evidencia una creación de archivos ".txt" adicionales para identificadores que no corresponden a los jugadores que se encuentran realizando el Test dentro de la región de interés seleccionada, tal y como se observa en la Figura 38 A y B; este error se debe principalmente a que el algoritmo de detección utilizado (Detectron2) realiza la identificación de todos los sujetos que se encuentren en el video que está analizando.

Figura 38. Creación de Múltiples Identificadores para Agentes Externos

A



B

[63] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose Flow: Efficient Online Pose tracking," (2018), p. 1–12.
[64] M. Snower, A. Kadav, F. Lai, and H. P. Graf, "15 Keypoints Is All You Need," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., (2020), p. 6737–6747.
[65] "Players Tracking using AI". {En línea}. {11 abril de 2021} disponible en: (https://www.linkedin.com/pulse/players-tracking-using-ai-stephan-janssen/?articleId=6617113389682438144).

Este error se trató de solucionar a través de software, sin embargo, y debido a que la creación de estos archivos .txt está ligada a la verificación de las listas de estados de Aprobación/Desaprobación creadas para los identificadores asignados en el primer frame, no fue posible limitar la aparición de los archivos excedentes ya que no había registro de esos identificadores. Una forma más sencilla de evitar este tipo de inconvenientes es que se procure grabar el video de la ejecución del test en un escenario deportivo sin personas ajenas al test mismo.

5.2. VALIDACIÓN DE RESULTADOS

Para la validación de los resultados obtenidos con el algoritmo de automatización se realizó una comparación entre las evaluaciones realizadas por un entrenador deportivo (Tabla 13) y las evaluaciones arrojadas por el programa (Tabla 14).

Tabla 13. Evaluaciones Deportivas Manuales.

Evaluación de los Jugadores				
Jugadores	**Nº Etapas**	**Nº Tramos**	**Velocidad (Km/h)**	**Capacidad Pulmonar VO_{2max}**
ID0	3	6	9,5	36,1915
ID1	8	9	12	50,834
ID2	2	1	9	33,263

Tabla 14. Evaluaciones Deportivas Automáticas.

Evaluación de los Jugadores				
Jugadores	**Nº Etapas**	**Nº Tramos**	**Velocidad (Km/h)**	**Capacidad Pulmonar VO_{2max}**
ID0	3	6	9,5	36,18
ID1	5	5	10,5	42,047
ID2	2	1	9	33,26

Dado que la ejecución del algoritmo de automatización es interrumpida en el minuto 4:45 del video, debido al cambio de identificador de uno de los jugadores, la Tabla N° 14 solo presenta las evaluaciones que realiza el programa hasta el punto de interrupción, mientras que la Tabla N° 13 se realiza en base al análisis de la ejecución total del programa. Sin embargo, y dado que los deportistas con los identificadores 0 y 2 terminan su ejecución del test antes del minuto mencionado, es posible comparar sus resultados, los cuales, tal y como se aprecia en ambas tablas, concuerdan casi al 100%, en la capacidad pulmonar calculada.

5.3. RECOMENDACIONES DE USO.

Haciendo una recopilación de todas las observaciones hechas a lo largo del documento, y en base a los resultados obtenidos, se presentan las siguientes recomendaciones de uso del algoritmo para la obtención de mejores resultados.

- El video deberá ser grabado en un escenario deportivo en donde se aprecie únicamente a los jugadores que se encuentren realizando el Test de Leger, puesto que el algoritmo de detección utilizado reconoce a todos los individuos presentes en la grabación, lo que puede acarrear posibles errores en el seguimiento de los deportistas.
- Según las pruebas realizadas, se recomienda la grabación del video desde una perspectiva frontal y elevada al escenario deportivo. El valor de la elevación es subjetivo ya que solo se realizaron pruebas a 3 metros sobre el nivel del suelo, es probable que una ubicación de la cámara a mayor o menor altura presente mejores o peores resultados. No es necesario implementar más de una cámara para grabar toda la ejecución del test.
- El video deberá tener solo máximo cuatro jugadores para evaluar, y dichos jugadores deberán mantener una distancia prudente entre ellos para evitar que existan cruces que puedan alterar los identificadores. En las pruebas realizadas, los deportistas mantuvieron una distancia aproximada de 2 metros, sin embargo, es probable que se puedan obtener resultados satisfactorios utilizando una distancia menor o mayor a ese valor.
- Se recomienda a los jugadores que realizan el test, que a la hora de terminar su prueba salgan del escenario por un costado en el que no intervengan con el recorrido de otro jugador.
- Para una ejecución más rápida y eficiente del programa, es deseable contar con las siguientes especificaciones de hardware: Procesador Ryzen 5 o Intel i5 en adelante, Tarjeta Gráfica dedicada NVIDIA, Almacenamiento interno mínimo de 30, RAM de 8 G. Cabe aclarar que las anteriores especificaciones no aseguran un procesamiento en tiempo real del programa.

6. CONCLUSIONES

- Se demostró que es posible implementar un algoritmo para la automatización del Test de Course – Navette, basado en el lenguaje de programación de Python, y que cuenta con las siguientes características: interfaz de usuario para la entrada de datos que facilita la interacción con el programa, análisis del recorrido de los deportistas, detección de cruces en las líneas de meta de acuerdo a la señal sonora que rige el desarrollo del test, cálculo de la capacidad pulmonar correspondiente a la etapa final alcanzada y visualización de resultados a través de archivos de texto, independientes para cada jugador, que quedan disponibles para que el usuario pueda realizar un análisis posterior al procesamiento. Sin embargo, se encontraron ciertas limitaciones en su funcionamiento, relacionadas con los errores ocasionados por la interacción de agentes externos al escenario deportivo, con las personas que se encuentran ejecutando el test.
- Se evaluó la posibilidad de obtener una estrategia de automatización para el Test de Course – Navette haciendo uso de OpenPose, sin embargo, y de acuerdo con los resultados obtenidos, se llegó a la conclusión de que esta librería de detección no cumplía con el requerimiento básico que se necesitaba para la implementación de la primera etapa del programa; tal y como se expuso en la sección de resultados, OpenPose, presentó una deficiencia en la detección a profundidad de los jugadores que no fue posible de mejorar, debido a esto, y con el fin de continuar con el desarrollo del programa, se definió como herramienta a utilizar en la etapa de detección, la librería de Detectron2, la cual, contrario a OpenPose, permitió mantener la detección de los deportistas frame a frame, en una amplia variedad de condiciones a las que no respondía apropiadamente OpenPose.
- Producto de la identificación de los principales parámetros y variables involucrados en el proceso de automatización del Test de Course – Navette, se evidenció la necesidad de implementar las tres etapas de desarrollo planteadas en el capítulo 4 del documento, las cuales, permitieron llevar a cabo los procesos de detección, seguimiento y evaluación de cada uno de los jugadores, de forma satisfactoria y de acuerdo con los requerimientos del programa.
- Durante las pruebas realizadas, se evaluaron diversos parámetros relacionados con el tipo y número de cámara a utilizar, la perspectiva de grabación de los videos, y el número máximo de jugadores a evaluar, esto con el fin de determinar el mejor arreglo de cámaras (posición, altura, tipo y número) que permitiera una captura eficiente de todo el test y un análisis más simple para el programa. En base al análisis efectuado, se concluye que es posible obtener resultados satisfactorios haciendo uso de un solo dispositivo móvil como elemento de grabación, sin embargo, para evitar errores de

seguimiento, se define una perspectiva de grabación frontal y elevada frente a la ubicación del escenario deportivo, y se limita a 4, el número máximo de jugadores a evaluar por video. Con respecto al valor de la elevación de la cámara, no se realizan indicaciones especificas ya que, en las pruebas realizadas, no se evaluaron los resultados que se podrían obtener a diferentes alturas.

- Durante la comparación de los resultados obtenidos con el programa implementado, frente a las evaluaciones realizadas por un entrenador deportivo, para un mismo video de prueba, se evidenció que el programa de automatización desarrollado evalúa con gran precisión la ejecución del test de cada uno de los jugadores, y por ende, presenta resultados iguales a los obtenidos por un evaluador humano.
- La principal limitación que se encontró durante el desarrollo del programa de automatización estuvo relacionado con el elevado consumo computacional que representó la ejecución de los algoritmos implementados para la etapa de detección, seguimiento y evaluación; durante el análisis de alternativas de hardware, Sección 4.4.6, realizado con el fin de evaluar la mejor opción para solventar, en cierta medida, esta condición, se encontró que existen herramientas gratuitas y disponibles al público en general, tales como Google Colab, que permiten afrontar esta clase de retos asociados a las aplicaciones de visión artificial, sin embargo, y tal y como se mencionó en dicho análisis, esta herramienta está sujeta a condiciones externas muy susceptibles a cambios espontáneos, lo que dificultó, en cierta medida, su uso dentro del programa.
- Uno de los principales desafíos dentro del desarrollo del programa de automatización, fue el seguimiento de los jugadores en un ambiente tan exigente como lo es el escenario deportivo en donde se ejecuta el Test de Course – Navette, ya que los constantes cruces y oclusiones entre jugadores ocasionaban que la herramienta implementada fallara en su función principal, sin embargo, y de acuerdo con la revisión del estado del arte realizada, este reto es común en una gran cantidad de aplicaciones similares a la desarrollada, en las cuales, ni las herramientas de seguimiento más robustas logran los resultados deseados; pese a esto, la solución implementada con el programa de CentroidTracker presenta resultados muy satisfactorios, siempre y cuando, se mantengan en la medida de lo posible, las recomendaciones de uso planteadas en la sección 5.3.

7. TRABAJO A FUTURO

- El algoritmo creado fue diseñado para realizar el análisis del Test de Course - Navette a partir de un video previamente grabado, esto puede ser mejorado en trabajos futuros a tal punto de que se pueda realizar un análisis en tiempo real en el escenario en donde se esté ejecutando el Test, para esto es necesario hacer uso de hardware más robusto y eficiente, algunas opciones son: Tarjeta Gráfica Geforce RTX 3090, Procesador Ryzen 9 5900x, RAM de 64 GB, Almacenamiento de estado sólido de 512 GB y una fuente certificada Evga Supernova G5 - 750w - Gold Full.
- El algoritmo puede ser modificado para retornar más parámetros o métricas producto de la evaluación del recorrido de los deportistas, tales como su aceleración o velocidad instantánea. De igual forma, es posible añadir etapas de análisis que permitan identificar posibles lesiones en los jugadores, así como también, estrategias de juego.
- Se plantea realizar un trabajo de análisis de optimización, en cuanto a posibles estrategias de software y hardware, que puedan ser implementadas dentro del proyecto y que permitan mejorar los resultados obtenidos con el algoritmo desarrollado.
- Con respecto al error correspondiente a la perdida de seguimiento por oclusión o cruce entre jugadores, se propone evaluar los resultados obtenidos al adicionar al algoritmo implementado, una etapa de detección y clasificación de personas por colores, que permita mantener los identificadores asignados a los jugadores cuando el algoritmo de seguimiento principal falle.
- La metodología de desarrollo y la estrategia de automatización planteadas en este proyecto pueden ser extrapoladas al análisis de otros test deportivos, de tal forma que se pueda hacer frente a los productos que se encuentran en el mercado actual, a través de modelos basados en software libre, independientes de requerimientos especiales en cuanto al escenario y que no hagan uso de marcadores sobre el cuerpo del deportista.

BIBLIOGRAFÍA

Visión por computadora - Libro online de IAAR.”. {En línea}. {18 marzo de 2021} disponible en: (https://iaarbook.github.io/vision-por-computadora/).

“A 2019 guide to Human Pose Estimation with Deep Learning.”. {En línea}. {18 marzo de 2021} disponible en: (https://nanonets.com/blog/human-pose-estimation-2d-guide/)

“What is Sports Analytics? Victor Holman Explains - Agile Sports Analytics.”. {En línea}. {10 marzo de 2021} disponible en: (https://www.agilesportsanalytics.com/what-is-sports-analytics/)

“Hudl: We Help Teams and Athletes Win.”. {En línea}. {18 marzo de 2021} disponible en: (https://www.hudl.com/).

“Kitman Labs - Powering Human Performance.”. {En línea}. {18 marzo de 2021} disponible en: (https://www.kitmanlabs.com/).

“SAS: Analytics, Artificial Intelligence and Data Management | SAS.”. {En línea}. {18 marzo de 2021} disponible en: (https://www.sas.com/en_ae/home.html).

“The analytics database | Exasol.”. {En línea}. {18 marzo de 2021} disponible en: (https://www.exasol.com/).

“SportVU - Stats Perform.”. {En línea}. {21 marzo de 2021} disponible en: (https://www.statsperform.com/team-performance/football-performance/optical-tracking/).

E. Cheshire, C. Halasz, and J. K. Perin, “Player Tracking and Analysis of Basketball Plays,” (2015).

R. M. Nieto and J. M. M. Sanchez, “An automatic system for sports analytics in multi-camera tennis videos,” En: IEEE Int. Conf. Adv. Video Signal Based Surveill., (2013).

V. Vovk, S. Skuratovskyi, P. Vyplavin, and I. Gorovyi, “Light-Weight Tracker for Sports Applications,” Signal Process. Symp. SPSympo, (2019).

K. Apostolou and C. Tjortjis, “Sports Analytics algorithms for performance prediction,” 10th Int. Conf. Information, Intell. Syst. Appl. IISA, (2019).

V. C. Pantzalis and C. Tjortjis, “Sports Analytics for Football League Table and Player Performance Prediction,” (Octu 2020).

M. Stein et al., “Revealing the Invisible: Visual Analytics and Explanatory Storytelling for Advanced Team Sport Analysis,” Symp. Big Data Vis. Immersive Anal. BDVA, (2018).

L. Sha et al., “Interactive Sports Analytics: An Intelligent Interface for Utilizing Trajectories for Interactive Sports Play Retrieval and Analytics,” ACM Trans. Comput. Interact., vol. 29, (2010).

M. Gowda et al., “Bringing IoT to sports analytics,” Proc. 14th USENIX Symp. Networked Syst. Des. Implementation, (2017).

Y. Xu and Y. Peng, “Real-Time Possessing Relationship Detection for Sports Analytics,” Chinese Control Conf. (Jul 2020).

G. C. García and J. D. Secchi, “Test Course Navette de 20 metros con etapas de un minuto. Una idea original que perdura hace 30 años,” Apunt. Med. l’Esport, vol. 49, no. 183, (2014).

M. Firinola G, “Pruebas de Campo para la Valoración del Consumo Máximo de Oxígeno, La Velocidad Aeróbica Máxima, y La Resistencia Intermitente,” Rev. electrónica Ciencias Apl. al Deport., vol. 2, no. 5, {En línea}. {2019} disponible en: http://www.romerobrest.edu.ar/ojs/index.php/ReCAD%0APruebas

“OpenCV”. {En línea}. {31 marzo de 2021} disponible en: (https://opencv.org/)

“Python.org”. {En línea}. {31 marzo de 2021} disponible en: (https://www.python.org/)

R. Jena, “Out of the Box: A combined approach for handling occlusions in Human Pose Estimation,” (2019).

Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, (2021).

T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., p. 3686–3693. {En línea}. {2015} disponible en: (http://arxiv.org/abs/1405.0312).

M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2D Human Pose Estimation: New Benchmark and State of the Art Analysis,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., no. December, (2014).

“GitHub - Detectron2”. {En línea}. {31 marzo de 2021} disponible en: (https://github.com/facebookresearch/detectron2).

“Facebook AI”. {En línea}. {31 marzo de 2021} disponible en:(https://ai.facebook.com/).

“GitHub - facebookresearch/Detectron”. {En línea}. {10 marzo de 2021} disponible en:(https://github.com/facebookresearch/Detectron/).

R. A. G, N. Neverova, and I. Kokkinos, “DensePose: Dense Human Pose Estimation In The Wild”, (2018.

A. Kirillov, P. Dollár, R. Girshick, and K. He, “Panoptic Feature Pyramid Networks,” (2019).

X. Chen, P. Dollár, R. Girshick, and K. He, “TensorMask: A Foundation for Dense Object Segmentation,” (2019).

K. He, G. Gkioxari, R. Girshick, and P. Dollár, “Mask R-CNN”.

“Model Zoo Detectron2”. {En línea}. {10 marzo de 2021} disponible en: (https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md).

“GOTURN : Deep Learning based Object Tracking | Learn OpenCV”. {10 marzo de 2021} disponible en: (https://learnopencv.com/goturn-deep-learning-based-object-tracking/).

“Object Tracking using OpenCV (C++/Python)”. {10 marzo de 2021} disponible en: (https://learnopencv.com/object-tracking-using-opencv-cpp-python/).

D. Held, S. Thrun, and S. Savarese, “Learning to Track at 100 FPS with Deep Regressions Networks,” Comput. Vis. – ECCV 2016 Lect. Notes Comput. {10 marzo de 2021}, p. 749–765 disponible en:(http://davheld.github.io/GOTURN/GOTURN.html).

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” Proc. IEEE Int. Conf. Comput. (2011).

Y. Sakai, T. Oda, M. Ikeda, and L. Barolli, “An Object Tracking System Based on SIFT and SURF Feature Extraction Methods,” 18th Int. Conf. Network-Based Inf., vol. 18, (2015).

D. G. Viswanathan, “Features from Accelerated Segment Test (FAST)”, (2011), p. 5.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6314, (2010).

“Simple object tracking with OpenCV - PyImageSearch”. {10 marzo de 2021} disponible en: (https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/)

“Spatial data structures and algorithms (scipy.spatial) — SciPy”. {02 abril de 2021} disponible en: (https://scipy.github.io/devdocs/tutorial/spatial.html).

“SciPy.org — SciPy.org”. {02 abril de 2021} disponible en: (https://www.scipy.org/).

D. F. Villamarín Zapata, “Estado del Arte, Herramientas y Aplicaciones para Transformaciones Geométricas 3D,” X Congr. Cienc. y Tecnol. (2015).

OpenCV: Geometric Image Transformations”. {31 marzo de 2021} disponible en: (https://docs.opencv.org/master/da/d54/group__imgproc__transform.html).

N. Cross, “Design Research: A Disciplined Conversation” Des. Issues, vol. 15, (abril de 1999).

F. Rodríguez Guisado, “Prescripción de Ejercicio para la Salud (I): Resistencia Cardiorrespiratoria,” Apunt. - Educ. Física y Deport., no. 39,(1995).

J. B. Maga, “Estimación de la Distancia a un Objeto con Visión Computacional,” Ingeniería, vol. 21, (2017).

S. Diamantas, S. Astaras, and A. Pnevmatikakis, “Depth Estimation in Still Images and Videos Using a Motionless Monocular Camera”. (marzo de 2018).

S. Diamantas, S. Astaras, and A. Pnevmatikakis, “Depth Estimation in Still Images and Videos Using a Motionless Monocular Camera,” IST 2016 - 2016 IEEE Int. Conf. Imaging Syst. Tech. Proc., (marzo 2018).

F. S. Khan, R. M. Anwer, J. Van De Weijer, A. D. Bagdanov, M. Vanrell, and A. M. Lopez, “Color Attributes for Object Detection,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., (2012).

Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, “Pose Flow: Efficient Online Pose tracking,” (2018).

M. Snower, A. Kadav, F. Lai, and H. P. Graf, “15 Keypoints Is All You Need,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., (2020).

“Players Tracking using AI”. {En línea}. {11 abril de 2021} disponible en: (https://www.linkedin.com/pulse/players-tracking-using-ai-stephan-janssen/?articleId=6617113389682438144).

C. Curilem Gatica, A. Almagiá Flores, and Y. F. Tuillang, “Aplicación del Test Course Navette en Escolares,” Rev. Mot. Humana, Edición, vol. 16, (2015).

ANEXOS

Anexo A. Documentación Técnica del Algoritmo de Automatización.

Para la documentación técnica del Test de Course – Navette, se hizo uso de la plataforma de alojamiento de código para el control de versiones y colaboración, GitHub, en donde se creó el repositorio oficial del proyecto, que puede ser encontrado en el siguiente enlace.

Dentro del repositorio se encuentran los siguientes archivos:

1. Códigos: En esta carpeta se alojan todos los scripts desarrollados
2. Explicación Códigos: Este archivo corresponde al documento con la explicación completa y detallada de cada uno de los scripts.
3. Instalación de Software: Este archivo corresponde al documento con la guía completa de creación e instalación del entorno de desarrollo del proyecto.

De igual forma, dentro de la descripción del proyecto, se agregan tres enlaces de YouTube, el primero corresponde a la explicación del Test y los últimos dos son los videos de los resultados finales.

Para la correcta ejecución del programa, se deben realizar los siguientes pasos:

1. Luego de crear el entorno de ejecución, asegúrese de guardar todos los scripts y el video del Test a analizar en una misma carpeta.
2. Diríjase al script de Selección_ ROI.py y ejecútelo para realizar la selección de los 4 puntos que definen la región de interés a analizar dentro del video. Las indicaciones para la selección de los puntos se encuentran dentro del script, luego de realizar la selección de los puntos, se debe crear un archivo de nombre “puntos.data” dentro de la carpeta que contiene los demás códigos.
3. Diríjase al script de Principal.py y ejecútelo para realizar el análisis de posiciones de los jugadores del video. Para la correcta ejecución de este script se necesitan de tres parámetrows, la dirección de ruta del video a analizar, la dirección de ruta donde se va a guardar el resultado final y el tiempo de inicio del Test en el video, asegúrese de ingresar de forma correcta todos los datos.