Estudio Comparativo de Algoritmos de Visión Computacional Orientados al Control de Iluminación Publica

Christian Eduardo Bastidas Carlosi
Jhon Alexander Urbina Burbano

San Juan de Pasto
Universidad CESMAG
Facultad de Ingeniería
Programa de Ingeniería Electrónica
2025

Estudio Comparativo de Algoritmos de Visión Computacional Orientados al Control de Iluminación Publica

Christian Eduardo Bastidas Carlosi
Jhon Alexander Urbina Burbano

Asesor
José Camilo Eraso Guerrero

San Juan de Pasto
Universidad CESMAG
Facultad de Ingeniería
Programa de Ingeniería Electrónica
2025

Nota de exclusión

“El pensamiento que se expresa en esta obra es exclusiva responsabilidad del autor y no compromete la ideología de la Universidad CESMAG”.

Dedicatoria

A mis padres, por su amor incondicional, por su esfuerzo constante y por ser el ejemplo que me ha guiado en cada paso de mi vida. Gracias por enseñarme el valor del trabajo, la honestidad y la perseverancia. Este logro es el reflejo de su dedicación y del apoyo que siempre me brindaron, incluso en los momentos más difíciles.

Jhon Alexander Urbina

A mis padres, quienes con su apoyo, comprensión y sacrificio me permitieron alcanzar esta meta. Gracias por estar presentes en cada etapa de mi formación y por motivarme a seguir adelante sin rendirme. Todo lo conseguido en este camino es fruto de los valores que me inculcaron y del amor que siempre me han ofrecido.

Christian Eduardo Bastidas

Agradecimientos

Expresamos nuestro más profundo agradecimiento a la Universidad por brindarnos las herramientas, el conocimiento y el acompañamiento necesarios durante todo nuestro proceso de formación profesional. Su compromiso con la excelencia académica nos motivó a dar siempre lo mejor de nosotros.

Agradecemos sinceramente a cada una de las personas que forman parte de la comunidad universitaria, por su disposición, atención y apoyo constante. Cada esfuerzo, desde las distintas áreas, contribuyó de manera significativa a nuestro crecimiento académico y personal.

Nuestro reconocimiento especial es para los docentes del programa de Electrónica, quienes con su dedicación, exigencia y compromiso nos guiaron en la adquisición de conocimientos fundamentales para nuestra carrera. Gracias por su paciencia, entrega y por inspirarnos a continuar aprendiendo más allá del aula.

De manera muy especial, queremos expresar nuestra gratitud al Ingeniero José Camilo Eraso Guerrero, por su acompañamiento, orientación y asesoría durante el desarrollo de este trabajo de grado. Su compromiso, experiencia y disposición fueron clave para alcanzar los objetivos propuestos y culminar con éxito esta investigación.

Contenido

Lista de tablas

Lista de figura

Introducción

La gestión urbana, la eficiencia energética y la sostenibilidad constituyen prioridades para las ciudades contemporáneas. Entre los desafíos más relevantes se encuentra la administración de la iluminación pública, dado su impacto directo en el consumo de energía, los costos operativos y la calidad de vida de la ciudadanía. Los esquemas tradicionales, basados en niveles de iluminación fijos o en control no inteligente, suelen provocar un uso ineficiente de recursos y una huella ambiental innecesaria.

En este contexto, la incorporación de sistemas inteligentes apoyados en visión computacional permite abordar el problema de manera eficaz. Gracias al reconocimiento de patrones y a la toma de decisiones adaptativa, es posible ajustar la iluminación en función de la presencia de personas y de las condiciones del entorno, reduciendo consumos superfluos y mejorando el desempeño del sistema de alumbrado.

Este trabajo se centra en comparar el rendimiento de tres enfoques de visión computacional orientados al control de iluminación pública: (i) detección de objetos con YOLO (familia YOLOvX), (ii) estimación de pose humana con OpenPose y (iii) un método de extracción de características (área, contornos, centroides, relación de aspecto y proporción altura–ancho del cuerpo). La comparación se realiza en escenarios de exteriores y considera tres dimensiones de desempeño: exactitud, precisión y costo computacional.

El objetivo es identificar el enfoque que ofrezca el mejor balance entre capacidad de detección de personas y eficiencia de cómputo, aportando criterios técnicos para el desarrollo futuro de un sistema de dimerización autónomo en alumbrado público. Con ello, se busca contribuir a la gestión energética responsable y a la construcción de ciudades más sostenibles.

1. El problema de investigación

1.1 Objeto de investigación:

Algoritmos de visión computacional **YOLO**, **OpenPose** y de **Extracción de Características**, orientados al control de la iluminación pública.

1.2 Línea de investigación

Automatización y control: La línea de sistemas de automatización y control de la Universidad CESMAG desarrolla procesos investigativos orientados al modelamiento, simulación, diseño, desarrollo y evaluación de algoritmos de control, sistemas de control, sistemas inteligentes, control de procesos industriales, sistemas embebidos, acondicionamiento y procesamiento de señales, robótica, domótica e inteligencia artificial. [1]

1.3 Sub línea de investigación

Inteligencia artificial: La inteligencia artificial es una respuesta al deseo de aproximar el comportamiento humano y el pensamiento racional a diversos sistemas para la solución de determinadas problemáticas a través de diferentes técnicas para la solución de problemas, entre las que se destacan la lógica difusa, las redes neurales, los sistemas neuro-difusos y los algoritmos genéticos. De esta forma los resultados que se desprenden de los procesos investigativos desarrollados en esta línea se orientan a la generación de algoritmos y metodologías que presentan un comportamiento autónomo, dinámico que se adecua a la evolución del entorno. [1]

1.4 Descripción o planteamiento del problema

La gestión ineficiente de la iluminación pública constituye un desafío significativo para las empresas encargadas de su mantenimiento, particularmente en lo relacionado con el consumo energético y los costos operativos. En el caso de la empresa Sepal S.A., encargada del alumbrado público en la ciudad de Pasto, se reporta un consumo mensual de aproximadamente 4.344.329 kWh, correspondiente a más de 35.000 luminarias instaladas en la red pública. Este elevado consumo refleja la magnitud del impacto energético asociado al funcionamiento continuo de las

lámparas, muchas de las cuales operan a su máxima potencia durante toda la noche, sin considerar la presencia o ausencia de personas en los espacios circundantes.

Figura 1. Datos de consumo de iluminarias proporcionado por Sepal S.A.

Diversas investigaciones han demostrado resultados positivos en el ámbito del control de la iluminación. Por ejemplo, Donovan Meza, en su trabajo [2], diseñó e implementó un sistema embebido que logró reducir en un 40% el consumo eléctrico diario en aplicaciones de iluminación. No obstante, su investigación se centra en entornos indoor, lo que deja de lado varios desafíos propios de escenarios exteriores, como interferencias, oclusiones, variaciones de iluminación, largas distancias entre cámaras y personas, así como el costo computacional del procesamiento.

Para abordar las complejidades del entorno exterior, el paradigma de las Ciudades Inteligentes (Smart Cities) promueve la implementación de sistemas de alumbrado público inteligente y adaptativo [3]. Estos sistemas se basan en la Telegestión y la detección activa de presencia para optimizar el consumo. La limitación de los sensores de movimiento básicos en exteriores se supera mediante la integración de visión artificial y algoritmos de Aprendizaje Profundo (Deep Learning) [4].

El Deep Learning, particularmente utilizando Redes Neuronales Convolucionales (CNNs), ha revolucionado la detección de objetos y patrones, ofreciendo una solución robusta para la identificación precisa y en tiempo real de peatones y vehículos en la vía pública [5], [6].

Estos modelos son capaces de gestionar eficazmente las variaciones lumínicas y las oclusiones inherentes a los escenarios exteriores, incluso en condiciones de baja visibilidad o nocturnas [7]. Al integrar cámaras inteligentes que ejecutan estos algoritmos (Edge Computing), las luminarias pueden ajustar su nivel de intensidad lumínica de manera dinámica y predictiva, garantizando la seguridad sin desperdicio energético. La implementación de estos sistemas ha demostrado la capacidad de lograr ahorros energéticos significativamente superiores en el alumbrado público respecto a los sistemas tradicionales.

Ante lo anterior, surge la necesidad de implementar sistemas de control de iluminación en espacios exteriores. Estos sistemas deben enfrentar condiciones más complejas, entre ellas la presencia de **obstáculos** (ramas, objetos, animales, etc.) y factores ambientales cambiantes como la lluvia o la neblina. En este contexto, la **visión computacional** se presenta como una herramienta clave para el **reconocimiento de siluetas humanas** que permita controlar de manera eficiente la iluminación pública en presencia de personas.

Por ello, resulta fundamental identificar los algoritmos de mayor rendimiento capaces de diferenciar con precisión las siluetas humanas de otros elementos presentes en el entorno (animales, ramas, troncos u objetos), garantizando así el funcionamiento adecuado del sistema propuesto. Los sistemas de detección basados en **Aprendizaje Profundo** (*Deep Learning*) han demostrado ser superiores a los métodos tradicionales en términos de precisión y robustez ante variaciones ambientales [8].

Particularmente, los algoritmos que realizan la detección de objetos en tiempo real, como **YOLO** (You Only Look Once) en sus diversas versiones (v3, v4, v5, v7, v8) y **SSD** (Single Shot MultiBox Detector), son candidatos ideales. Estos modelos se caracterizan por su **alta velocidad de inferencia**, crucial para sistemas embebidos de bajo costo en luminarias (*Edge Computing*), y mantienen una **elevada precisión** incluso en condiciones de baja luminosidad y oclusiones parciales [9], [10]. Otros modelos basados en dos etapas, como **Faster R-CNN** (Region-based Convolutional Neural Network), aunque pueden ofrecer una precisión marginalmente superior, a menudo se descartan en aplicaciones de tiempo real en entornos *Edge* debido a su **mayor costo computacional** [11].

En este sentido, el presente trabajo se orienta a la **comparación** de estos **algoritmos de visión computacional** (YOLO, SSD, Faster R-CNN) aplicados al control inteligente de la iluminación pública, evaluando métricas clave como la **precisión** (AP/mAP) y la **velocidad de procesamiento** (FPS) para determinar el balance óptimo entre rendimiento y eficiencia energética [12].

1.5 Formulación de problema

¿Cuál es el rendimiento que presentan los algoritmos de YOLO, OpenPose y de Extracción de Características basados en visión computacional en términos de exactitud, precisión y costo computacional, para el reconocimiento de personas enfocado al control de iluminación pública?

1.6 Objetivos

1.6.1 Objetivo General

Analizar el rendimiento que presentan los algoritmos de YOLO, OpenPose y de Extracción de Características basados en visión computacional en términos de exactitud, precisión y costo computacional, para el reconocimiento de personas enfocado al control de iluminación pública.

1.6.2 Objetivo Específicos

- Implementar los algoritmos Yolo, OpenPose y de Extracción de Características
- Establecer los protocolos de experimentación, con el fin de que el algoritmo controle la iluminación en exteriores
- Determinar el rendimiento de los algoritmos de visión computacional en términos de exactitud, precisión y costo computacional

1.7 Justificación

El reciente desarrollo e impacto del paradigma de Deep Learning en el campo de la Inteligencia Artificial ha permitido un notable avance en el reconocimiento de patrones en

imágenes y video, alcanzando en algunos casos un desempeño superior al humano. Un factor clave en este progreso es la capacidad de procesar grandes volúmenes de información en aplicaciones exitosas [2].

La implementación de algoritmos de visión computacional ofrece importantes ventajas en diversas áreas de la sociedad, tales como transformar sistemas no controlados en sistemas autónomos, operar en tiempo real, diferenciar personas de objetos, adaptarse a distintas condiciones de iluminación e identificar múltiples individuos de manera simultánea, manteniendo alta precisión en diferentes condiciones climáticas y realizando análisis continuos sin necesidad de intervención humana. La aplicación de la visión por computador ha demostrado ser una estrategia efectiva para la optimización del consumo energético en sistemas de iluminación [13].

Estas ventajas pueden aprovecharse en el futuro para el desarrollo de sistemas inteligentes aplicados al alumbrado público, con el propósito de reducir el consumo energético, incrementar la autonomía de los sistemas, disminuir las emisiones de gases de efecto invernadero y optimizar los costos que afectan a la población en general.

En esta investigación se plantea realizar una comparación de algoritmos de visión computacional orientados al reconocimiento de personas, con el fin de determinar cuál de ellos presenta un mejor desempeño en escenarios de espacios abiertos. La correcta selección del algoritmo de detección (como YOLO, SSD o Faster R-CNN) es crítica, ya que debe proporcionar una alta precisión sin comprometer la velocidad de procesamiento y el costo computacional inherente a los sistemas embebidos (Edge Computing) [14]. El algoritmo con mayor rendimiento en las pruebas podrá ser implementado en el desarrollo futuro de un prototipo de sistema autónomo de dimerización para lámparas de alumbrado público. Dicho sistema buscará regular de manera inteligente la intensidad lumínica en función de variables como la hora de operación, las condiciones ambientales y el flujo vehicular y peatonal, con el objetivo de optimizar el consumo de energía y prolongar la vida útil de las luminarias.

La implementación de esta tecnología representa una oportunidad para modernizar y hacer más eficientes los sistemas de alumbrado público de la ciudad de Pasto. Al ofrecer una solución autónoma, adaptable y energéticamente eficiente, se pretende contribuir directamente al desarrollo de Ciudades Inteligentes (Smart Cities), en concordancia con las tendencias globales de sostenibilidad y gestión inteligente de recursos públicos. Asimismo, este tipo de soluciones puede convertirse en una herramienta estratégica para las empresas prestadoras del servicio de alumbrado, al permitirles optimizar sus operaciones, reducir costos y ofrecer un servicio más avanzado y respetuoso con el medio ambiente.

1.8 Delimitación

Este proyecto se centra en el diseño, implementación y evaluación de distintos algoritmos de visión computacional, con énfasis en medir su precisión, exactitud y costo computacional. La atención se dirige al análisis del comportamiento de estos algoritmos mediante la observación de videos de tráfico peatonal. Dicho análisis constituye un aporte clave para la implementación exitosa de técnicas de visión computacional orientadas a la optimización del consumo energético en el alumbrado público.

Es importante señalar que el alcance de este proyecto se limita a la fase de análisis y evaluación de algoritmos, sin contemplar la implementación física del sistema en entornos reales. Esta delimitación responde a diversos factores, entre los que destacan los requerimientos legales y administrativos necesarios para instalar equipos en espacios públicos, como parques o vías urbanas, lo cual exige permisos especiales por parte de las empresas concesionarias del servicio de alumbrado.

A pesar de que no se implementará un sistema físico en un entorno real, se realizará una simulación utilizando videos que representen el funcionamiento del sistema bajo condiciones reales. Para ello, se emplearán grabaciones que simulan el comportamiento de una cámara instalada a alturas de 6 y 8 metros, correspondientes a las ubicaciones típicas de cámaras en parques públicos.

Asimismo, el tiempo disponible para la ejecución del proyecto constituye una restricción significativa, ya que el desarrollo, validación e instalación de un sistema físico de dimerización autónoma implica procesos logísticos, técnicos y administrativos que exceden los plazos de esta investigación.

Por estas razones, se opta por enfocar el trabajo en la simulación y el análisis comparativo de diferentes algoritmos de control, priorizando aquellos que demuestren altos niveles de eficiencia, estabilidad y adaptabilidad ante diversas condiciones de operación. Este enfoque permitirá generar una base sólida de conocimientos y resultados, que podrá servir posteriormente como punto de partida para el diseño e implementación de un prototipo funcional en escenarios reales, siempre que se cuente con los recursos, autorizaciones y condiciones necesarias para su desarrollo.

2. Tópicos del marco teórico

2.1 Antecedentes

2.1.1 Deep Learning para la detección de peatones y vehículos sobre FPGA

En [2], se propone un estudio para implementación del paradigma Deep Learning con una CNN (redes neuronales convolucionales) para clasificar imágenes de señales de tránsito vehicular con el propósito de medir el tiempo de entrenamiento y su desempeño en la clasificación y también se investiga la tecnología relacionada con FPGAs (Field Programmable Gate Array), para determinar la forma en que se puede acelerar el cómputo implicado en ese tipo de redes con estos dispositivos, validándolos como una alternativa de implementación para sistemas embebidos.

En este trabajo se utiliza una imagen en escala de grises de dimensiones m x n, la cual se introduce en una arquitectura de red neuronal convolucional (CNN) implementada en un FPGA. Inicialmente, la imagen atraviesa una capa convolucional compuesta por seis filtros de tamaño 5x5 (k = 6), generando como resultado un volumen de salida con dimensiones 12 x 12 x 6. Este volumen resultante se procesa nuevamente mediante una segunda capa convolucional, esta vez con 12 filtros, obteniendo una salida final de 4 x 4 x 12, tal como se ilustra en la Figura 2.

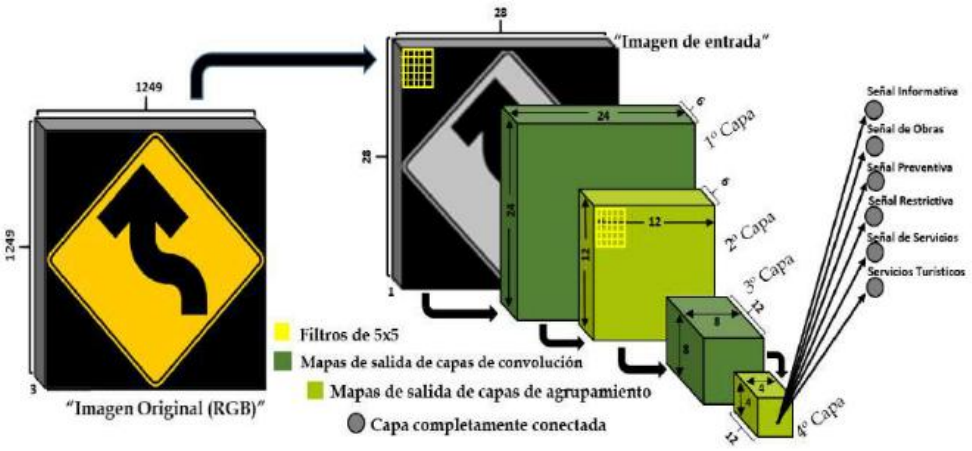


Figura 2. Arquitectura de la CNN implementada

Nota. Autor Vizcaya Cárdenas, Rigoberto,2018

El trabajo realizado fue con el propósito de evaluar si es posible implementar Redes Neuronales Convolucionales, que tengan pocas capas de convolución y aun así obtener resultados adecuados en la tarea que realizan. El error en la clasificación más bajo que se obtuvo fue de 8.87%, cuando el tamaño del lote fue de 32 y el experimento de 1400 épocas (en 3.16 horas de entrenamiento), lo que significa que de cada 100 imágenes de tránsito vehicular que se le presenten a la CNN, aproximadamente 91 de ellas las clasifique correctamente.

2.1.2 Estudio técnico económico de alumbrado público contemplando energía fotovoltaica e inteligencia artificial

El estudio técnico-económico del sistema de alumbrado público presentado en [15] analiza de forma detallada la viabilidad de implementar tecnologías avanzadas para optimizar el consumo energético y mejorar la eficiencia operativa en zonas rurales y urbanas. En particular, se enfoca en un proyecto de iluminación pública vial que conecta al municipio de Bojacá con la vereda El Corzo, en el departamento de Cundinamarca, Colombia. Esta vía, por su importancia estratégica y su bajo nivel de iluminación, ha sido seleccionada como caso de estudio para evaluar la aplicabilidad de fuentes de energía convencionales y no convencionales, incluyendo la energía fotovoltaica.

El artículo destaca cómo la integración de sistemas de inteligencia artificial (IA) permite una regulación inteligente de la tensión y una gestión más eficiente del consumo energético, aspectos clave para garantizar la sostenibilidad del sistema. La digitalización de datos es importante en esta estrategia, permitiendo la recolección, procesamiento y análisis de información en tiempo real, lo cual facilita la toma de decisiones automatizadas y la identificación de patrones de consumo.

La propuesta combina tecnologías de automatización con modelos de predicción basados en IA para controlar la intensidad del alumbrado de acuerdo con variables contextuales como la hora del día, el clima, la presencia de vehículos o peatones y las condiciones específicas del entorno. Esto permite reducir el consumo energético sin comprometer la seguridad vial.

Desde el punto de vista económico, el estudio compara los costos de implementación, operación y mantenimiento entre sistemas tradicionales alimentados por la red eléctrica y sistemas híbridos que integran paneles solares fotovoltaicos. Se evidencia que, a pesar de un

mayor costo inicial de los sistemas con energía solar y tecnología inteligente, el retorno de inversión (ROI) a mediano y largo plazo es favorable debido al ahorro energético, la disminución de intervenciones de mantenimiento correctivo y la mayor vida útil de los componentes.

Igualmente, el artículo propone un enfoque modular y escalable, lo que permite adaptar la solución a diferentes contextos geográficos y presupuestarios. El uso de inteligencia artificial también contribuye a anticipar fallos, optimizar el uso de recursos y garantizar la operación continua del sistema, incluso en condiciones adversas.

2.1.3 Sistema embebido de detección de movimiento mediante visión artificial

En el artículo referenciado en [16], se presenta el diseño e implementación de un sistema embebido de bajo costo orientado a la detección de movimiento utilizando técnicas de visión artificial. El núcleo del sistema se basa en una Raspberry Pi 3, una plataforma compacta y versátil que permite ejecutar algoritmos de procesamiento de imágenes en tiempo real, siendo adecuada para aplicaciones de vigilancia, monitoreo ambiental y sistemas de seguridad inteligente.

El algoritmo principal empleado se fundamenta en la técnica de sustracción de fondo, una metodología clásica pero eficiente en la detección de objetos en movimiento dentro de una escena estática. Esta técnica consiste en comparar cada nuevo fotograma capturado con un modelo de fondo previamente construido, permitiendo así identificar áreas de cambio que corresponden a elementos en movimiento. La implementación fue desarrollada en lenguaje Python, aprovechando bibliotecas especializadas como OpenCV para el procesamiento de imágenes y NumPy para el manejo de datos.

Durante el desarrollo y evaluación del sistema, se analizó el rendimiento de la Raspberry Pi 3 al ejecutar este tipo de algoritmos. Se determinó que el sistema era capaz de procesar imágenes a una tasa promedio de 12,5 fotogramas por segundo (fps), lo que corresponde a un intervalo de muestreo de aproximadamente 80 milisegundos por fotograma. Esta tasa se considera adecuada para aplicaciones en tiempo real, especialmente en entornos donde la detección de movimiento no requiere latencias extremadamente bajas, como en vigilancia residencial, monitoreo rural o control de acceso.

Aparte de la detección de movimiento, el sistema demostró una capacidad estable para la extracción de características visuales, permitiendo una posible extensión hacia tareas más

complejas como el reconocimiento de objetos, conteo de personas o clasificación de eventos. La eficiencia alcanzada valida el uso de plataformas embebidas de bajo consumo energético como la Raspberry Pi 3 en aplicaciones de visión computacional, especialmente en proyectos donde se requiere autonomía, bajo costo y portabilidad.

2.1.4 Algoritmo YOLO para aprendizaje profundo e inteligencia artificial

El estudio presentado en [17] ofrece un análisis detallado sobre la implementación del algoritmo YOLOv4 (You Only Look Once) en tareas de detección de múltiples objetos, aplicando técnicas avanzadas de *deep learning* e inteligencia artificial. La investigación se centra específicamente en aplicaciones de videovigilancia urbana, con el objetivo de optimizar el control del tráfico vehicular mediante sistemas automatizados de visión computacional, lo cual resulta directamente aplicable a soluciones modernas de gestión inteligente en entornos urbanos.

YOLOv4 se destaca como uno de los algoritmos más eficientes en tareas de detección en tiempo real, gracias a su arquitectura mejorada que equilibra con eficacia la velocidad de inferencia y la precisión. Este algoritmo realiza la detección de objetos en una sola pasada (*single shot*) por la imagen, dividiéndola en una cuadrícula y aplicando predicciones simultáneas sobre las posibles ubicaciones y clases de los objetos. Esta estructura permite un desempeño superior frente a otras arquitecturas más pesadas y secuenciales.

En el estudio mencionado, el modelo YOLOv4 fue entrenado con un conjunto de datos personalizado compuesto por imágenes de tráfico vial en la India. Dicho dataset incluía nueve clases relevantes, entre ellas vehículos de diferentes tipos y peatones, simulando un entorno de tráfico real. Los resultados obtenidos evidenciaron un desempeño sobresaliente: el algoritmo alcanzó una precisión de detección del 99.28% sobre el conjunto de prueba, y una tasa de acierto superior al 94.44% en clases específicas, lo cual refleja una alta fiabilidad incluso en contextos con múltiples objetos y condiciones ambientales variables.

Al considerar el enfoque de nuestro proyecto, orientado a la detección de peatones y vehículos para el control adaptativo del alumbrado público, el uso de YOLOv4 se perfila como una opción altamente efectiva. Su capacidad para operar en tiempo real, junto con su precisión y adaptabilidad, lo convierte en un recurso ideal para implementar sistemas de iluminación inteligente que respondan dinámicamente a la presencia de personas o tráfico vehicular, reduciendo el consumo energético y mejorando la seguridad en las vías.

2.1.5 Caracterización para la ubicación en la captura de video

El estudio presentado en [18], publicado en la *Revista Colombiana de Tecnologías de Avanzada (RCTA)*, ofrece un análisis detallado sobre la caracterización para la ubicación en la captura de video, con un enfoque en su aplicación a técnicas de visión artificial para la detección de personas. Este artículo establece una base técnica clave para comprender los principios de la captura y procesamiento de imágenes en sistemas de visión artificial, destacando su importancia en contextos como la seguridad pública, el monitoreo de espacios públicos y la automatización de procesos basados en la detección de objetos o individuos.

El enfoque central de este trabajo reside en cómo la ubicación y el posicionamiento de las cámaras de video son determinantes para optimizar los resultados en la implementación de algoritmos de visión computacional. A través de un análisis técnico exhaustivo, se abordan aspectos esenciales de la instalación, la calibración y la configuración de cámaras, lo cual resulta de gran ayuda para maximizar el rendimiento de los sistemas de detección de personas. También, se destacan las buenas prácticas para garantizar que las cámaras capturen las imágenes de manera efectiva, sin distorsión ni zonas ciegas, y se obtengan los datos más precisos posibles.

El artículo también profundiza en el uso de Python como herramienta de programación para el desarrollo de sistemas de visión artificial. Python es una opción popular debido a su simplicidad y al amplio soporte de bibliotecas especializadas como OpenCV, TensorFlow y Keras, que permiten la implementación eficiente de algoritmos de detección de personas y objetos. Se describen las fases de programación y entrenamiento de modelos, abarcando desde la recolección de datos hasta el ajuste de parámetros de los modelos, incluyendo la etapa de entrenamiento supervisado para mejorar la precisión del algoritmo.

Uno de los aspectos más relevantes que aborda el estudio es la interacción entre la ubicación de las cámaras y el rendimiento del sistema de visión artificial. La orientación de las cámaras debe estar alineada con las necesidades del entorno de aplicación, asegurando que el ángulo de visión cubra adecuadamente las áreas clave sin perder calidad en las imágenes o en el procesamiento posterior. Esto se convierte en un desafío especialmente en aplicaciones dinámicas, como la vigilancia en tiempo real de zonas con alta afluencia de personas, donde el algoritmo debe ser capaz de identificar individuos de forma precisa y rápida, a pesar de la variabilidad de las condiciones lumínicas, el movimiento de objetos y la obstrucción de vistas parciales.

De igual manera, el artículo subraya la importancia de contar con una infraestructura de soporte que permita la integración de las cámaras de manera fluida con los sistemas de procesamiento de imágenes y de inteligencia artificial. Esto incluye aspectos como la conexión en red, la disponibilidad de almacenamiento para los datos generados, y la capacidad de transmitir imágenes de alta resolución a través de plataformas seguras para su posterior análisis. Esta infraestructura debe ser diseñada de tal manera que se optimice tanto el rendimiento como el costo, sin comprometer la calidad de la detección.

En términos de aplicaciones prácticas, este estudio resulta esencial para el desarrollo de sistemas que integren inteligencia artificial para la detección de personas, ya que proporciona los lineamientos necesarios para configurar correctamente los equipos y garantizar que los algoritmos entrenados en el entorno adecuado logren un alto nivel de precisión. La correcta caracterización de la ubicación de las cámaras también permite la optimización de los sistemas de visión artificial en aplicaciones de seguridad urbana, monitoreo de accesos, automatización del tráfico y gestión de espacios públicos, entre otros.

2.1.6 Sistema inteligente y compacto de iluminación

El estudio presentado en [19] aborda el diseño, implementación y evaluación de un sistema de iluminación inteligente y compacto, orientado principalmente hacia la dimerización de la luz y el control adaptativo de la misma. Este enfoque busca no solo mejorar la eficiencia energética, sino también ofrecer una experiencia de iluminación más ergonómica, personalizada y sensible al entorno. El artículo destaca cómo las tecnologías emergentes, al integrarse con sistemas embebidos, pueden transformar el rendimiento y la funcionalidad de soluciones lumínicas modernas.

Una de las características centrales del sistema propuesto es su capacidad para ajustar automáticamente la intensidad luminosa en función de diversas variables del entorno, como la detección de movimiento, la luz ambiente y el horario del día. Este comportamiento adaptativo es posible gracias a la incorporación de sensores de presencia, sensores de luminosidad y algoritmos de control embebido, los cuales permiten tomar decisiones en tiempo real sobre cuándo encender o apagar la lámpara, así como qué nivel de iluminación es el más adecuado en cada situación.

El diseño del sistema incluye el uso de tecnología LED, conocida por su bajo consumo energético y alta durabilidad, junto con un microcontrolador programado para gestionar las entradas de los sensores y aplicar las técnicas de regulación de luz. Esta arquitectura permite operar en distintos modos de funcionamiento: encendido automático al detectar movimiento, apagado por inactividad, y modulación del brillo según las condiciones de iluminación natural presentes en el ambiente. Igualmente, se incluye una función avanzada de ajuste del tono de la luz (temperatura de color), lo cual permite generar ambientes más cálidos o fríos dependiendo del momento del día, favoreciendo el confort visual y el ahorro energético.

La versatilidad del sistema también se manifiesta en su tamaño compacto, lo cual facilita su integración en diferentes entornos, desde espacios domésticos hasta aplicaciones urbanas, sin requerir grandes adaptaciones estructurales. Esta característica hace que el diseño sea ideal para implementaciones en zonas donde se necesita iluminación localizada, eficiente y controlada, como paraderos, pasillos peatonales, parques o vías rurales.

Los resultados experimentales presentados en el estudio validan la funcionalidad del sistema, evidenciando que se logró mantener niveles adecuados de luminosidad con un consumo optimizado de energía eléctrica. De igual manera, se logró una regulación del tono de la luz, lo cual influye positivamente en la percepción del entorno y el bienestar del usuario. La combinación entre hardware y software permite una gestión inteligente que responde dinámicamente a las necesidades del espacio, alineándose con los principios de sostenibilidad y eficiencia energética.

Este tipo de soluciones adquiere una relevancia especial en el contexto de proyectos de iluminación pública inteligente, como el que se plantea en nuestro estudio, ya que la capacidad de adaptación del sistema a variables externas permite una gestión más racional de los recursos. La automatización del encendido/apagado y la regulación de la intensidad luminosa no solo disminuyen el gasto energético, sino que también contribuyen a extender la vida útil de los componentes del sistema, reducir costos de mantenimiento y minimizar el impacto ambiental.

2.2 Supuestos teóricos

2.2.1 Visión Computacional

La visión computacional es una rama impórtate de la inteligencia artificial (IA) y del procesamiento de señales que tiene como objetivo dotar a las máquinas de la capacidad de percibir e interpretar información visual del entorno, emulando de forma automatizada las funciones del sistema visual humano. Esta disciplina permite a los sistemas informáticos capturar, analizar y extraer conocimiento útil a partir de imágenes estáticas o secuencias de video, a través del uso de algoritmos avanzados y modelos matemáticos.

En esencia, la visión computacional busca que los dispositivos electrónicos sean capaces de entender escenas visuales mediante la adquisición de datos desde cámaras digitales, sensores ópticos o sistemas de captura tridimensional, para luego aplicar técnicas de análisis que permitan identificar, segmentar y clasificar objetos, reconocer patrones, detectar movimiento, estimar distancias, y comprender el contexto visual de forma autónoma.

El proceso técnico de la visión computacional involucra varias etapas, entre ellas: adquisición de imagen, preprocesamiento (mejoramiento de contraste, eliminación de ruido), segmentación, extracción de características (bordes, formas, texturas, colores), clasificación y, en muchos casos, seguimiento y predicción del comportamiento de los objetos detectados. Estas tareas se apoyan en técnicas clásicas del procesamiento digital de imágenes, pero en la actualidad han sido potenciadas por el uso de redes neuronales convolucionales (CNN) y modelos de aprendizaje profundo (deep learning), que permiten una interpretación visual mucho más precisa y adaptativa.

El uso de la visión computacional se ha extendido ampliamente en aplicaciones reales donde se requiere un procesamiento visual eficiente y en tiempo real. Por ejemplo, en el contexto de sistemas inteligentes de alumbrado público, esta tecnología permite detectar la presencia de peatones, ciclistas o vehículos, activando o ajustando la iluminación solo cuando sea necesario, lo que contribuye a una gestión energética más eficiente y sostenible.

También, la incorporación de visión artificial en sistemas embebidos de bajo costo (como Raspberry Pi, Jetson Nano o ESP32-CAM) ha democratizado su implementación en una amplia gama de proyectos, desde investigaciones académicas hasta soluciones comerciales y urbanas.

2.2.2 Redes neuronales convolucionales

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) constituyen una arquitectura esencial dentro del aprendizaje profundo (deep learning), diseñada específicamente para procesar datos con estructura espacial como imágenes y videos. Este tipo de red neuronal ha demostrado ser altamente eficiente en la extracción de patrones espaciales y características significativas, lo que la convierte en una herramienta fundamental para la visión por computadora [20].

Una CNN está compuesta por un conjunto de capas organizadas jerárquicamente, cada una con funciones específicas [20]:

• Capas convolucionales: se encargan de aplicar filtros (kernels) sobre la entrada para generar mapas de características, permitiendo detectar patrones locales como bordes, texturas y formas. A medida que la red profundiza, se capturan representaciones de mayor complejidad.

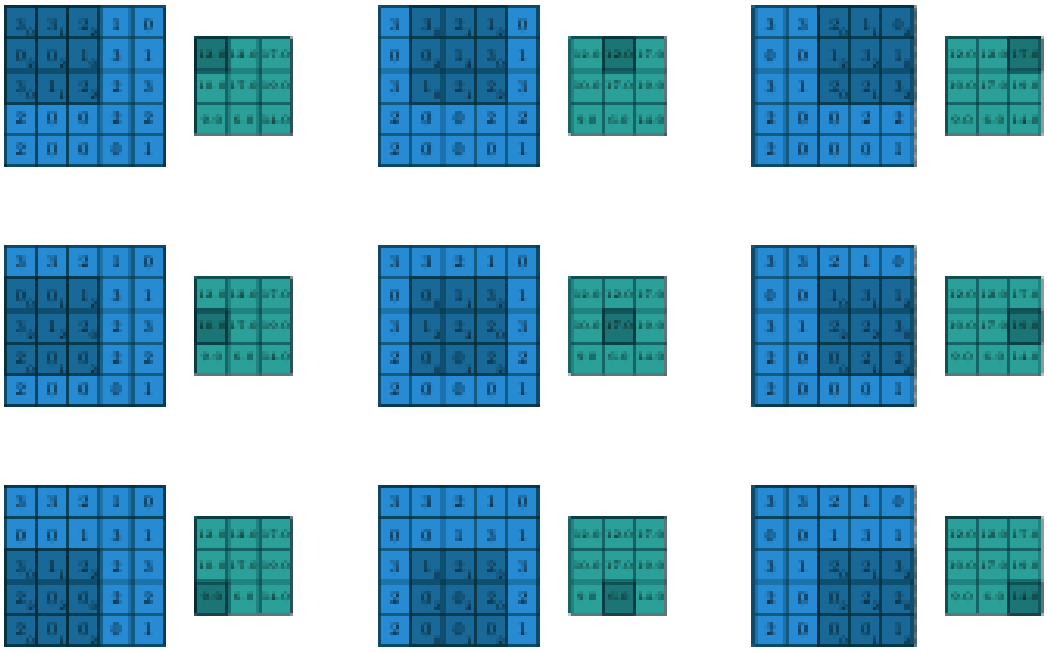


Figura 3. Ejemplo de una convolución discreta en dos dimensiones

Nota. Dumoulin y Visin [20].

• Capas de activación: introducen no linealidad en el modelo. La más utilizada es la función ReLU (Rectified Linear Unit), que mejora la capacidad de aprendizaje de la red.

• Capas de agrupamiento (pooling): reducen la dimensionalidad de los mapas de características conservando la información más relevante. Esto mejora la eficiencia computacional y disminuye el riesgo de sobreajuste.

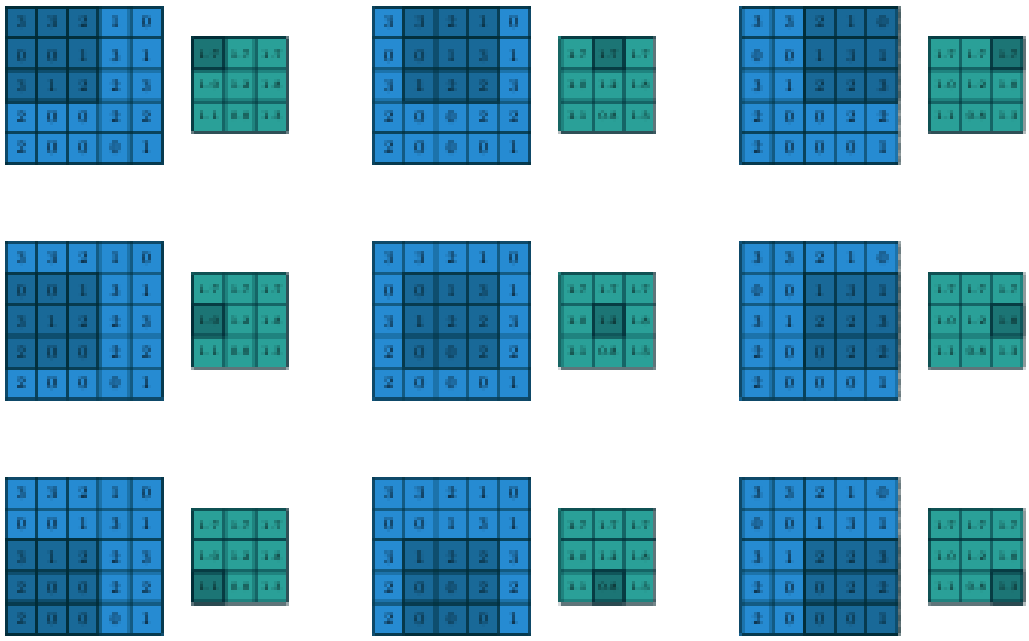


Figura 4. Ejemplo de average pooling aplicado a una matriz 5×5

Nota. Dumoulin y Visin [20].

• Capas completamente conectadas (fully connected): integran las características extraídas en capas anteriores para generar la salida final, que puede corresponder a una clasificación, una regresión o una tarea de detección de objetos.

La principal ventaja de las CNN frente a los métodos tradicionales de visión artificial es que aprenden automáticamente las características relevantes directamente a partir de los datos, eliminando la necesidad de diseñar manualmente descriptores como bordes o puntos de interés [20].

En el contexto de esta investigación, las redes neuronales convolucionales juegan un papel crucial, ya que permiten identificar con precisión elementos presentes en el entorno urbano como peatones, ciclistas, vehículos y otros objetos clave, cuya detección influye directamente en la toma de decisiones para el control adaptativo de sistemas de iluminación pública. Mediante la implementación de CNNs, es posible desarrollar sistemas capaces de reaccionar en tiempo real

ante la presencia de personas o tráfico, regulando la intensidad luminosa de forma automática para optimizar el consumo energético sin comprometer la seguridad.

Igualmente, las CNN han sido integradas exitosamente en algoritmos de última generación como YOLO, Faster R-CNN, Mask R-CNN, entre otros, permitiendo realizar tareas complejas de detección y segmentación con alta precisión y velocidad. Estas capacidades son especialmente útiles en sistemas embebidos de visión artificial utilizados en entornos urbanos, donde se requiere una respuesta rápida y eficiente.

El entrenamiento de estas redes requiere conjuntos de datos etiquetados y un procesamiento computacional considerable, aunque actualmente existen bibliotecas como TensorFlow, PyTorch, Keras y OpenCV, que facilitan la implementación de modelos CNN tanto en entornos de investigación como en sistemas reales.

2.2.3 Algoritmos de Deep Learning

Los sistemas de Deep Learning se utilizan para identificar objetos en imágenes, transcribir discursos en texto, relacionar noticias, publicaciones o productos con los intereses de los usuarios y seleccionar resultados de búsqueda relevantes [3], en este documento los algoritmos que se utilizaran son:

YOLO: El algoritmo YOLO (You Only Look Once) es una técnica de detección de objetos en tiempo real que ha tenido un impacto significativo en el área de la visión por computadora. A diferencia de métodos tradicionales basados en regiones propuestas, YOLO realiza la detección como un único proceso de regresión, mapeando directamente desde los píxeles de la imagen hasta las coordenadas de las cajas delimitadoras y sus respectivas clases [9].

En su funcionamiento, YOLO divide la imagen de entrada en una cuadrícula de tamaño S×S, donde cada celda predice B cuadros delimitadores, la confianza para dichos cuadros y C probabilidades de clase. La salida por celda se representa como:

$$Salida = S * S * (B * 5 + C) \quad (1)$$

Donde:

- **S×S:** número total de celdas en la imagen.
- **B:** número de bounding boxes que cada celda predice.

- **5:** corresponde a $(x, y, w, h,$ confianza)
- **C:** número de clases.

Este enfoque permite realizar detección con latencias menores a los 25 milisegundos, haciéndolo ideal para aplicaciones en tiempo real como vehículos autónomos, videovigilancia y sistemas embebidos [21].

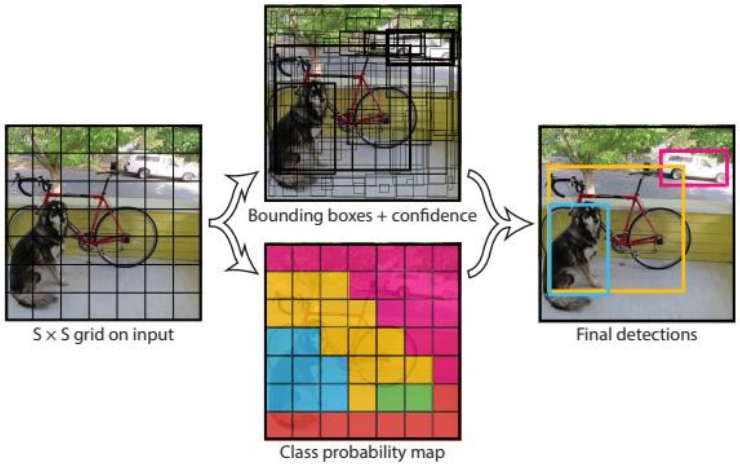


Figura 5.Modelo documento YOLO

Nota. Imagen tomada del documento [21]

Arquitectura de YOLOv8

Desde su primera versión, YOLO ha evolucionado con mejoras en precisión y eficiencia. Las versiones posteriores —YOLOv2, YOLOv3, YOLOv4 y YOLOv5— introdujeron componentes como capas más profundas, normalización por lotes y mejoras en las funciones de pérdida.

La versión YOLOv8, desarrollada por Ultralytics, representa el estado del arte al incorporar una arquitectura moderna y eficiente, eliminando componentes obsoletos como los anchores boxes, y añadiendo capacidades adicionales como segmentación y estimación de poses.

YOLOv8 presenta una arquitectura modular compuesta por tres bloques:

1. **Backbone (C2f):** Utiliza una variante eficiente del CSPDarknet para la extracción de características jerárquicas de la imagen.
2. **Neck (FPN + PAN):** Funde información a múltiples escalas mediante pirámides de características, mejorando la detección de objetos de diferentes tamaños.
3. **Head (Anchor-Free):** Genera las predicciones finales sin necesidad de anclas predeterminadas, lo que simplifica el entrenamiento y mejora la generalización.

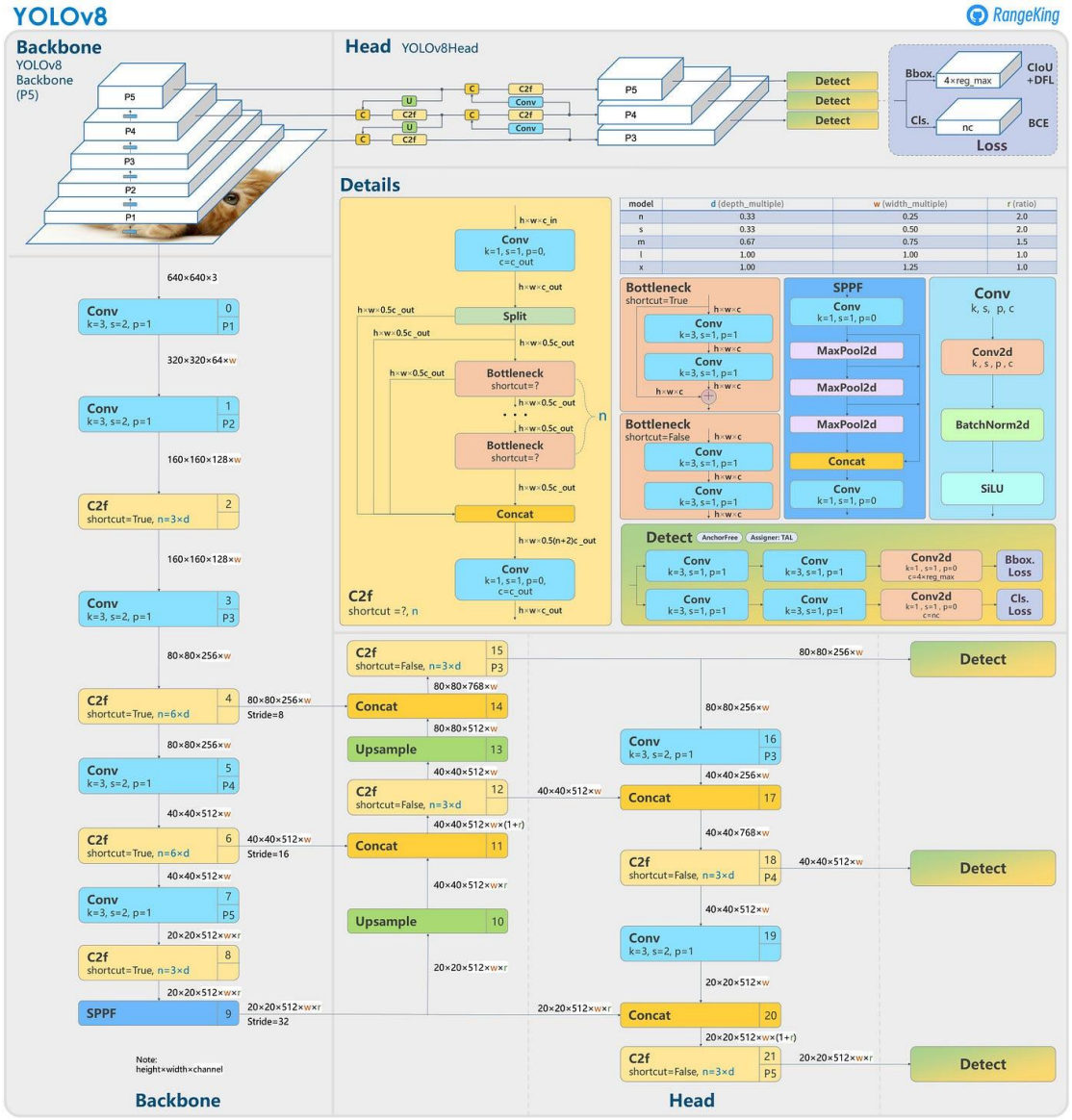


Figura 6.Estructura del modelo YOLOv8

Nota. Imagen tomada del documento [22]

Datos COCO

El algoritmo YOLOv8 ha sido entrenado principalmente con el conjunto de datos COCO (Common Objects in Context), uno de los más utilizados en visión por computadora. Este dataset contiene más de 330,000 imágenes anotadas con 80 clases de objetos comunes en contextos cotidianos, e incluye no solo cajas delimitadoras, sino también segmentación y anotaciones para keypoints

3

Figura 7.Muestra de anotación en el conjunto de datos MS COCO

Nota. Imagen tomada del documento [23]

Comunidad y Código Abierto

YOLOv8 ha sido liberado como **software de código abierto**, lo cual ha impulsado una comunidad activa de desarrollo. Esto permite su uso académico e industrial sin restricciones, fomentando la mejora continua y la creación de variantes adaptadas a casos de uso específicos. Ultralytics proporciona una documentación oficial completa, así como soporte para tareas de detección, segmentación y clasificación mediante una sola interfaz [14], O como se puede observar en la Figura 8.

Figura 8. Detección con YOLO [14]

OPENPOSE: es una biblioteca de código abierto desarrollada por el Carnegie Mellon Perceptual Computing Lab, diseñada para la detección y estimación de poses humanas en imágenes y videos en tiempo real.

El algoritmo utiliza una red neuronal convolucional (CNN) para estimar la posición de múltiples puntos clave (keypoints) del cuerpo humano, incluyendo cuerpo, rostro, manos y pies, alcanzando hasta 135 puntos de referencia en su versión más completa.

A diferencia de los enfoques top-down, que primero detectan personas y luego sus partes, OpenPose emplea un enfoque bottom-up.

Este enfoque permite detectar primero las partes del cuerpo individualmente (por ejemplo, codos, rodillas o tobillos) y posteriormente agruparlas en estructuras coherentes que representan la silueta humana completa en la escena.

El proceso de detección se basa en dos componentes fundamentales:

Mapas de confianza (Confidence Maps): describen la probabilidad de que un punto de la imagen corresponda a una parte específica del cuerpo.

Campos de afinidad (Part Affinity Fields, PAFs): representan, mediante vectores, la relación espacial entre dos puntos clave (por ejemplo, entre el hombro y el codo o entre la rodilla y el tobillo).

Matemáticamente, OpenPose aprende estos patrones a través del entrenamiento de una CNN que genera, para cada píxel de la imagen, las salidas $S_j(x)\ y\ L_C(x)$, correspondientes a los mapas de confianza y los campos de afinidad respectivamente.

Durante el entrenamiento, se minimiza una función de pérdida definida como:

$$E = \sum_j ||S_j - S^*_j||^2 + \sum_c ||L_C - L^*_C||^2$$

son los valores reales del conjunto de entrenamiento.

El modelo aprende así a predecir la ubicación y orientación de las articulaciones humanas.

Una vez generados los mapas y los campos de afinidad, el algoritmo aplica un proceso de agrupamiento (matching) basado en la asociación por proximidad de vectores (Nearest Neighbor Matching) para reconstruir la postura completa de cada persona.

Esto permite identificar múltiples individuos de manera simultánea, incluso cuando hay superposición o parcial ocultamiento de las siluetas.

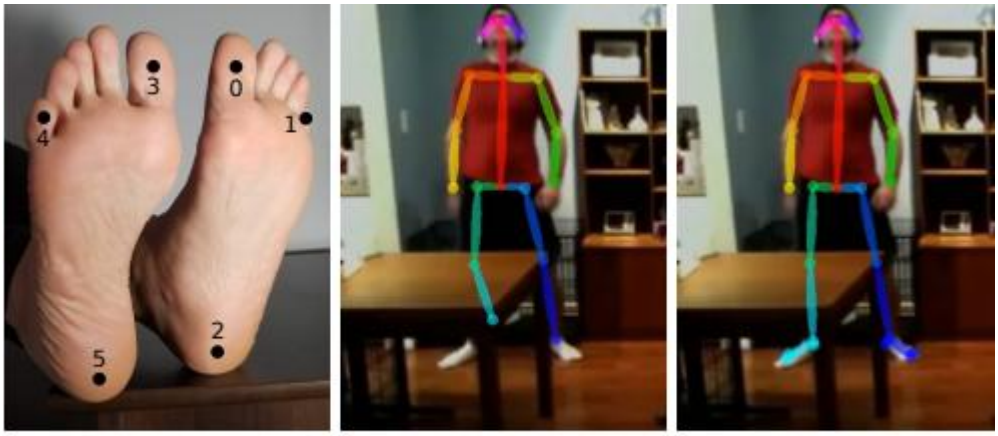


Figura 9. Análisis de puntos Clave (ketpoint) del pie

Nota. Tomada de [24]

Extracción de características: La extracción de características en el procesamiento de imágenes abarca tanto características globales como locales para describir y entender la información presente en las imágenes. Las características globales se refieren a propiedades que abarcan toda la imagen, como color predominante, forma general y distribución de texturas. Por otro lado, las características locales se centran en regiones específicas de la imagen, como patrones detallados, texturas y la disposición de elementos pequeños.[25]

Como se menciona en el artículo clasificador de imágenes de frutas basado en inteligencia artificial, esta técnica lo que hace es utilizar los espacios de los pixeles y realiza una relación entre ellos y dependiendo del histograma de la imagen se realiza una extracción del objeto deseado al considerar características locales, se puede analizar la textura de la piel de la fruta, la presencia de ciertos patrones o incluso características como la forma y el tamaño de detalles específicos, como el tallo o las marcas distintivas. Estas características locales proporcionan información detallada que puede ser crucial para la clasificación precisa de diferentes variedades de frutas.[25]

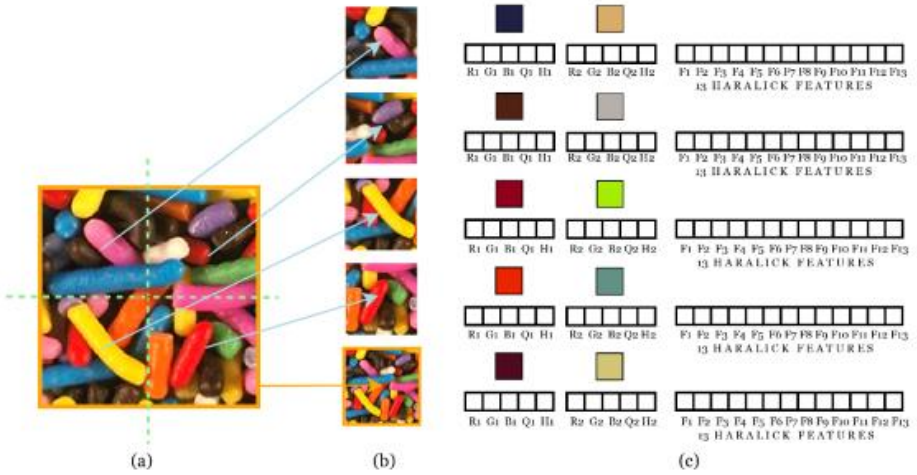


Figura 10. Vector de características extraído de una imagen de la base de datos vistex

Nota. Imagen tomada de [25]

2.2.4 Alumbrado público.

El alumbrado público es un servicio esencial que consiste en la iluminación de vías, parques, plazas, espacios recreativos y demás zonas de libre circulación que no se encuentren bajo la responsabilidad directa de ninguna persona natural o jurídica del ámbito privado o

público [24]. Este servicio desempeña un papel importante en la seguridad vial, la movilidad urbana y la percepción de seguridad de los ciudadanos, contribuyendo al embellecimiento y funcionalidad de los entornos urbanos durante las horas nocturnas.

La responsabilidad de la operación, mantenimiento y expansión del sistema de alumbrado público suele recaer en las autoridades municipales. En el caso de la ciudad de Pasto, la empresa encargada de prestar este servicio es Sepal, la cual debe cumplir con lo estipulado por el Reglamento Técnico de Iluminación y Alumbrado Público (RETILAP), normativa expedida por el Gobierno Nacional [26].

El RETILAP establece una serie de requisitos técnicos, ambientales, de eficiencia y de seguridad que deben cumplir todas las instalaciones de alumbrado público en Colombia. Su objetivo es garantizar niveles adecuados de iluminación para una correcta visibilidad, asegurar la calidad del servicio, proteger a los usuarios y al personal técnico, y minimizar los impactos ambientales derivados del uso de tecnologías de iluminación. Entre los aspectos más relevantes regulados por el RETILAP se encuentran:

Los niveles de iluminancia mínima, uniformidad lumínica, y el índice de reproducción cromática requeridos para distintas áreas públicas.

La obligación de implementar soluciones que promuevan la eficiencia energética, como el uso de lámparas LED y sistemas de regulación automática.

La reducción de la contaminación lumínica mediante el adecuado diseño de luminarias y la correcta orientación del flujo luminoso.

Las condiciones de seguridad eléctrica, mecánica y térmica de los equipos instalados, así como los protocolos de mantenimiento y revisión periódica.

Este marco regulatorio también busca incentivar la incorporación de tecnologías modernas que permitan un mayor control y eficiencia, como sistemas inteligentes que automatizan el encendido y apagado del alumbrado, sensores de presencia, tecnologías de comunicación para el monitoreo remoto, e incluso algoritmos de inteligencia artificial capaces de adaptar la iluminación a las condiciones del entorno.

También, se promueve la implementación de fuentes de energía no convencionales, como la energía solar fotovoltaica, lo que reduce la dependencia de la red eléctrica y contribuye a la sostenibilidad del sistema. Así, el alumbrado público no solo cumple una función operativa, sino

que también se convierte en un componente estratégico para el desarrollo de ciudades inteligentes y sostenibles.

En el contexto de este proyecto, el análisis del alumbrado público y su relación con las tecnologías emergentes es clave para proponer soluciones más eficientes, inteligentes y adaptadas a las necesidades reales del entorno, alineadas con las exigencias del RETILAP y la política energética nacional [26].

2.2.5 Google colab

Google Colaboratory, conocido comúnmente como Google Colab, es una plataforma de computación en la nube basada en Jupyter Notebooks que permite a los usuarios escribir y ejecutar código Python directamente desde un navegador web. Desarrollado por Google Research, Colab proporciona un entorno de ejecución preconfigurado para tareas de aprendizaje profundo, ofreciendo acceso gratuito a unidades de procesamiento gráfico (GPU) y unidades de procesamiento tensorial (TPU), lo que facilita la implementación y prueba de modelos de aprendizaje automático sin necesidad de configuraciones locales complejas.[27]

En el estudio realizado por Pessoa et al. (2018), se llevó a cabo un análisis detallado del rendimiento de Google Colab como herramienta para acelerar aplicaciones de aprendizaje profundo, especialmente en el ámbito de la visión por computadora. El artículo evalúa los recursos de hardware disponibles, el rendimiento en tareas específicas y las limitaciones inherentes al uso de esta plataforma. Los autores compararon el rendimiento de Colab con estaciones de trabajo tradicionales y servidores robustos equipados con múltiples núcleos físicos, utilizando casos de prueba que incluyen búsqueda combinatoria basada en árboles paralelos y aplicaciones de visión por computadora como detección y clasificación de objetos, así como localización y segmentación de objetos. [27]

Los resultados del estudio indican que, si bien Google Colab ofrece un rendimiento comparable al de estaciones de trabajo dedicadas para ciertas tareas, presenta limitaciones significativas en cuanto a la escalabilidad y la disponibilidad de recursos, especialmente en lo que respecta al número de núcleos de CPU disponibles. A pesar de estas restricciones, Colab se destaca por su accesibilidad y facilidad de uso, lo que lo convierte en una herramienta valiosa para investigadores y educadores que buscan una solución flexible y económica para ejecutar y compartir código Python en el ámbito del aprendizaje profundo.

2.2.6 Factor de transferencia de rendimiento

En el marco de esta investigación, se plantea un procedimiento para estimar el coste computacional de algoritmos que no han sido ejecutados directamente en un entorno local, pero cuyos datos de referencia se tienen en una plataforma diferente. Este procedimiento se fundamenta en la utilización de un parámetro denominado Factor de Transferencia de Rendimiento (FTR, por sus siglas en inglés), el cual constituye una herramienta ampliamente utilizada en la comparación y extrapolación de métricas de rendimiento entre diferentes entornos de ejecución.

El FTR se define como la relación entre el valor de una métrica medida en el entorno objetivo y el valor de la misma métrica medida en el entorno de referencia. Matemáticamente, esta relación se expresa de la siguiente manera:

$$FTR = \frac{M_{Pycharm}}{M_{colab}} \quad (2)$$

donde Mpycharm es el valor medido en el entorno local utilizando PyCharm, y Mcolab es el valor obtenido en el entorno de referencia Google Colab. Las métricas M pueden representar diversas variables de coste computacional, tales como el tiempo de ejecución en segundos, el consumo de memoria RAM en megabytes, el uso de GPU en vatios, la frecuencia de CPU en megahercios o el rendimiento gráfico medido en FPS (cuadros por segundo).

El procedimiento para aplicar este factor se basa en un supuesto de proporcionalidad, el cual indica que, si se han evaluado dos algoritmos (por ejemplo, YOLO y extracción de características) en ambos entornos y se ha calculado su relación de rendimiento, ese cociente puede ser utilizado para estimar el comportamiento de un tercer algoritmo (por ejemplo, OpenPose) en el entorno objetivo, aun cuando este último no haya sido ejecutado allí. Esto se debe a que, en condiciones controladas, la relación de rendimiento entre plataformas tiende a ser consistente para tareas de naturaleza similar, especialmente cuando se utilizan el mismo conjunto de datos, las mismas bibliotecas y versiones de software, y un entorno de ejecución comparable.

La fórmula para realizar esta estimación parte de la medición en el entorno de referencia y se ajusta aplicando el factor de transferencia:

$$M_{pycharm} = M_{colab} \times FTR \quad (3)$$

En términos prácticos, este método se implementa siguiendo un proceso en tres etapas. En primer lugar, se ejecutan algoritmos que sí se pueden correr en ambos entornos (en este caso,

YOLO y extracción de características), registrando todas las métricas relevantes en cada ejecución. En segundo lugar, se calcula el FTR para cada métrica, dividiendo el valor obtenido en PyCharm entre el valor correspondiente en Colab. Finalmente, estos factores se aplican a las mediciones de OpenPose realizadas en Colab para obtener una estimación de cómo se comportaría este algoritmo si fuera ejecutado en PyCharm en el computador local.

Este enfoque se encuentra respaldado en la literatura de arquitectura de computadores y evaluación de rendimiento, donde se ha demostrado que las relaciones de tiempo y consumo entre plataformas pueden mantenerse estables bajo cargas de trabajo equivalentes. Por ejemplo, en [11] describen que el rendimiento relativo entre dos sistemas puede ser expresado como la razón entre sus tiempos de ejecución:

$$\frac{Performance A}{Performance B} = \frac{T_B}{T_A} \tag{4}$$

donde TA y TB representan el tiempo de ejecución en los sistemas A y B, respectivamente. Este principio es directamente aplicable al cálculo de un FTR, ya que el tiempo de ejecución y otras métricas de rendimiento pueden transformarse de un sistema a otro mediante una simple relación de proporcionalidad.

El uso de este tipo de metodología se ve reforzado también en textos como [15] donde se detalla cómo obtener métricas comparables entre sistemas, y en estudios presentados en conferencias IEEE donde se realizan comparaciones de hardware mediante relaciones de rendimiento relativo. En dichas investigaciones, se destaca que la clave para que estas estimaciones sean fiables radica en garantizar que las pruebas se realicen en condiciones controladas y repetibles, utilizando el mismo conjunto de datos y evitando variaciones externas que puedan introducir sesgos en las mediciones.

Aplicando este marco teórico a la presente investigación, el cálculo del FTR no solo permite estimar el coste computacional de OpenPose en PyCharm sin necesidad de ejecutar el algoritmo localmente, sino que además proporciona una base cuantitativa sólida para comparar diferentes técnicas de visión por computadora bajo un mismo estándar de referencia. Este enfoque optimiza recursos, evita sobrecargar el equipo local con procesos pesados y, al mismo tiempo, permite obtener predicciones fundamentadas en datos empíricos y respaldadas por una teoría ampliamente aceptada en la comunidad científica.

2.3 Definición de conceptos

2.3.1 Definición nominal de conceptos

Las variables a medir en este proyecto son:

Exactitud (E): Capacidad de un sistema o instrumento para proporcionar un valor lo más cercano posible al valor verdadero o de referencia. En otras palabras, mide el error absoluto en la medición.

Precisión (P): Capacidad de un sistema o instrumento para generar resultados consistentes y repetibles al realizar múltiples mediciones bajo las mismas condiciones. La precisión indica qué tan cercanos están los resultados entre sí, independientemente de si son exactos o no.

Costo computacional: Conjunto de recursos necesarios (tiempo de procesamiento, memoria y consumo energético) que un algoritmo o sistema requiere para realizar una tarea determinada. En términos simples, describe el impacto del algoritmo en los recursos de cómputo.

Estas métricas permiten evaluar el rendimiento de un algoritmo y pueden derivarse a partir de una matriz de confusión, la cual organiza las frecuencias de predicción comparando clases predichas con clases reales, como se muestra en la Tabla 1.

Tabla 1. Matriz de confusión [14]

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

En el caso de este proyecto:

Verdaderos Positivos (VP): El sistema detecta una persona y realmente sí hay una persona.

Falsos Negativos (FN): El sistema no detecta una persona, pero sí hay una persona.

Falsos Positivos (FP): El sistema detecta una persona, pero no hay nadie.

Verdaderos Negativos (VN): El sistema dice que no hay persona y realmente no hay nadie.

2.3.2 Definición operativa de conceptos

Para calcular la exactitud se utiliza la siguiente ecuación.

$$E = \frac{VP + VN}{N} \tag{5}$$

Para medir la precisión se utiliza la ecuación.

$$P = \frac{VP}{VP+FP} \tag{6}$$

Donde:

E: Exactitud.

P: Precisión.

VP: Verdaderos positivos.

FP: Falsos positivos.

VN: Verdaderos negativos.

N: Tamaño muestral utilizado.

En cuanto al costo computacional, se tomará como referencia la metodología planteada en [12], la cual consiste en ejecutar los algoritmos bajo las mismas condiciones: en un mismo equipo de cómputo, utilizando las mismas herramientas de programación y conjuntos de imágenes idénticos. Esto garantiza que las mediciones se realicen en condiciones controladas y repetibles.

El costo computacional se medirá a partir de tres factores principales:

- **Tiempo de ejecución:** Segundos
- **Consumo grafico**: FPS
- **Rendimiento computacional**

GPU: Vatios
Memoria: Megabyte
CPU: Mega Hertz
RAM: Megabyte

2.4 Hipótesis

2.4.1 Hipótesis de investigación

El algoritmo de visión computacional Yolo presenta un 95% de exactitud y precisión, el algoritmo OpenPose presenta un 90% de exactitud y precisión y el algoritmo de extracción de características presenta un 80% de exactitud y precisión. Por su parte se espera que los algoritmos YOLO y OpenPose tengan un 10% más en cada una de las variables de costo computacional en comparación con el algoritmo de extracción de características para el reconocimiento de personas enfocado al control de iluminación pública.

2.4.2 Hipótesis nula

El algoritmo de visión computacional Yolo no presenta un 95% de exactitud y precisión, el algoritmo OpenPose no presenta un 90% de exactitud y precisión y el algoritmo de extracción de características no presenta un 80% de exactitud y precisión. Por su parte se espera que los algoritmos YOLO y OpenPose no tengan un 10% más en cada una de las variables de costo computacional en comparación con el algoritmo de extracción de características para el reconocimiento de personas enfocado al control de iluminación pública.

2.4.3 Hipótesis alternativa

El algoritmo de visión computacional Yolo presenta un 90% de exactitud y precisión, el algoritmo OpenPose presenta un 85% de exactitud y precisión y el algoritmo de extracción de características presenta un 75% de exactitud y precisión. Por su parte se espera que los algoritmos YOLO y OpenPose tengan un 5% más en cada una de las variables de costo

computacional en comparación con el algoritmo de extracción de características para el reconocimiento de personas enfocado al control de iluminación pública.

3. Metodología

3.1 Enfoque

El enfoque de la investigación es **cuantitativo**, dado que las variables de interés (exactitud, precisión y costo computacional) se expresan en valores numéricos y permiten realizar comparaciones objetivas entre los algoritmos analizados.

3.2 Paradigma

El neopositivismo se asocia con la investigación cuantitativa desde una perspectiva metodológica [14]. En este trabajo, la finalidad es comparar exactitud, precisión y costo computacional, con el objetivo de identificar qué algoritmo de visión computacional ofrece un mejor desempeño en función de las variables mencionadas y de los antecedentes revisados.

3.3 Método

Al tratarse de una investigación con enfoque cuantitativo, se adoptó el método científico (también conocido como método empírico-analítico). Este proporciona una estructura rigurosa para evaluar la eficiencia de los algoritmos de visión computacional en el control de iluminación pública. Su aplicación permite obtener evidencia empírica y objetiva sobre el desempeño de los algoritmos, contribuyendo a abordar de manera sistemática el problema planteado.

3.4 Tipo de investigación

La investigación es de tipo descriptiva, puesto que se centra en determinar qué algoritmos de visión computacional resultan más eficientes para el control adaptativo de la iluminación pública. El propósito es identificar la técnica más adecuada para su implementación en entornos exteriores, considerando factores como la presencia de obstáculos, variaciones ambientales y la necesidad de una adaptación dinámica en tiempo real. Para ello, se establecieron parámetros específicos que permitan evaluar el rendimiento de cada algoritmo en condiciones controladas, generando métricas de exactitud, precisión y costo computacional.

3.5 Diseño de investigación

El diseño corresponde a un **experimento puro**, ya que se compararon distintos algoritmos de visión computacional orientados a la iluminación pública.

RG X1 O1
RG X2 O2
RG X3 O3
RG X4 O4

Donde:

RG: Sistema de dimerización estándar de las lámparas de alumbrado público de tecnología LED

X1: Algoritmo de extracción de características orientado a la iluminación publica

X2: Algoritmo de visión computacional YOLO orientado a la iluminación publica

X3: Algoritmo de visión computacional OpenPose orientado a la iluminación publica

X4: Sistema de dimerización estándar de las lámparas de alumbrado público de tecnología LED

O1: Exactitud, precisión y costo computacional del sistema con algoritmo de extracción de características

O2: Exactitud, precisión y costo computacional del sistema con algoritmo YOLO

O3: Exactitud, precisión y costo computacional del sistema con algoritmo OpenPose

O4: Exactitud, precisión y costo computacional del sistema sin algoritmos de visión computacional

3.6 Universo

Algoritmos de visión computacional orientados a la iluminación pública.

3.7 Muestra

La muestra son los algoritmos de visión computacional YOLO, OpenPose y Extracción de Características.

3.8 Técnicas de recolección de información

La técnica utilizada en esta investigación es la **observación directa**, aplicada a la ejecución de algoritmos de visión computacional en entornos controlados. Mediante esta técnica se obtuvieron las métricas de exactitud, precisión y costo computacional de cada algoritmo, lo que permitió realizar una comparación objetiva de su desempeño en el contexto del control de la iluminación pública.

3.8.1 Validez de la técnica

La técnica de recolección de información es válida, debido a que varios artículos científicos, como [24] y [27], han utilizado métodos similares para recolectar datos sobre variables clave como exactitud, precisión y costo computacional en el contexto de algoritmos de visión computacional. Estos estudios han seguido estándares metodológicos rigurosos y han sido publicados en revistas especializadas, como se evidencia en el caso de [24]. Esta validación previa respalda la solidez de la metodología utilizada en esta investigación para obtener resultados precisos y comparables en la evaluación de los algoritmos considerados.

3.8.2 Confiabilidad técnica

La confiabilidad se asegura mediante la aplicación de pruebas tanto en escenarios simulados como en condiciones reales. Para ello, se trabajó con conjuntos de videos obtenidos en espacios públicos que incluyen diversas situaciones: presencia de personas, obstáculos y variaciones de iluminación. Estas pruebas garantizan que las métricas recolectadas reflejen el desempeño de los algoritmos bajo diferentes condiciones de operación.

3.9 Instrumentos de Recolección de la Información

Los instrumentos utilizados para obtener los datos de esta investigación fueron:

- **Cámara de seguridad Logitech C920x HD Pro**: utilizada para la captura de videos de prueba en un parque durante condiciones nocturnas, con resolución 1920 × 1080 píxeles.

- **Entorno de programación Python**: empleado para la ejecución de los algoritmos de visión computacional y el registro de métricas.
- **Google Colab:** entorno en la nube con acceso a GPU, usado para pruebas iniciales de alto rendimiento.
- **PyCharm (entorno local):** utilizado para pruebas en un equipo de cómputo convencional, con el fin de evaluar el rendimiento en condiciones de operación reales.
- **Herramientas de análisis de desempeño computacional:** incluyeron mediciones de tiempo de procesamiento por fotograma (ms/frame), FPS alcanzados, uso de CPU, GPU, memoria RAM y consumo energético.

3.10 Procesamiento de la información

El procesamiento de la información se realizó a partir de los datos recolectados durante la ejecución de los algoritmos de visión computacional, con el propósito de analizar su desempeño en términos de exactitud, precisión y costo computacional.

En primer lugar, se organizaron los registros obtenidos de cada prueba en hojas de cálculo, donde se consolidaron las métricas capturadas como el tiempo de procesamiento por fotograma (ms/frame), los fotogramas por segundo (FPS), y el uso promedio de CPU, GPU y memoria RAM. Esta información permitió establecer comparaciones entre los distintos algoritmos y entornos de ejecución utilizados.

Los datos fueron clasificados de acuerdo con el algoritmo empleado (YOLO, OpenPose y Extracción de Características) y con el entorno de ejecución (Google Colab y PyCharm). De esta manera, se facilitó la identificación de diferencias de rendimiento entre las pruebas realizadas en la nube y las desarrolladas en un equipo local.

El análisis incluyó la elaboración de tablas y gráficos comparativos utilizando herramientas como Python y Microsoft Excel, con el fin de representar visualmente los resultados obtenidos y facilitar la interpretación de los datos.

Adicionalmente, en el caso del algoritmo OpenPose, se aplicó un Factor de Transferencia de Rendimiento (FTR) que permitió estimar su comportamiento en diferentes entornos de ejecución. Este factor se obtuvo a partir de las variaciones observadas entre las métricas obtenidas en Google Colab y las estimadas en PyCharm, brindando una aproximación al rendimiento real del modelo en condiciones operativas locales.

4. Resultados

4.1 Diseño e implementación de algoritmos.

En la presente investigación, el diseño e implementación de algoritmos se centró en tres módulos principales: detección de objetos con YOLO, estimación de poses humanas mediante OpenPose y extracción de características orientada al análisis computacional posterior.

Cada uno de estos componentes fue desarrollado e integrado considerando las condiciones específicas del escenario de estudio: la vigilancia y análisis de un entorno de parque público. El sistema se estructuró bajo un enfoque modular que permite la ejecución secuencial de cada proceso, garantizando la trazabilidad de los resultados desde la adquisición del video hasta el análisis de los datos obtenidos.

Para representar de forma clara la interacción y secuencia de los procesos, se elaboró el diagrama de flujo mostrado en la Figura 4, el cual describe el recorrido lógico del sistema: desde la adquisición del video y su preprocesamiento, pasando por la detección de personas con YOLO, la estimación de puntos clave corporales mediante OpenPose y la posterior extracción de características, hasta llegar al almacenamiento y análisis de los resultados. Este diagrama evidencia la naturaleza modular del sistema, así como la interdependencia entre cada algoritmo para cumplir el objetivo general de monitoreo y análisis inteligente.

Los videos obtenidos no fueron procesados en tiempo real, sino que se analizaron posteriormente en un entorno controlado de laboratorio. Este enfoque permitió comparar de manera uniforme el rendimiento de los tres algoritmos bajo las mismas condiciones de entrada, sin depender de las limitaciones del hardware en campo. A través de este procesamiento diferido se obtuvieron métricas como tiempo de ejecución, uso de CPU, consumo de memoria RAM y tasa de cuadros procesados por segundo (FPS), las cuales se detallan en los apartados siguientes.

Aunque el sistema no fue acoplado directamente a una lámpara con dimerización real, el objetivo principal de esta etapa fue evaluar el rendimiento de los algoritmos de visión computacional como base para un futuro sistema de control de iluminación inteligente.

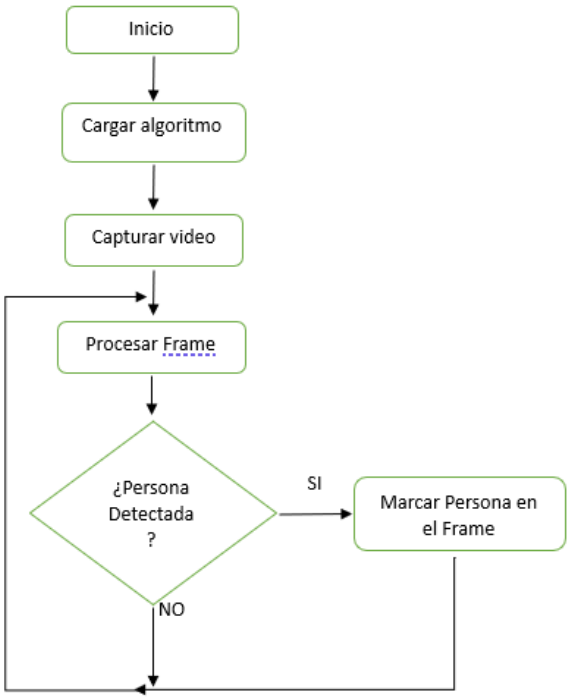


Figura 11. Diagrama de bloques básico que todos los códigos van a realizar

Nota. Autor

4.1.1 Implementación algoritmo YOLO.

Para implementar el algoritmo YOLO, se llevaron a cabo varias etapas que permitieron su integración en un sistema de detección en tiempo real de objetos en secuencias de video e imágenes estáticas. La metodología empleada se basa en redes neuronales convolucionales (CNN), lo que permite un procesamiento eficiente y preciso de los datos visuales.

Se empleó la librería OpenCV para el procesamiento de imágenes y videos, mientras que el desarrollo y la personalización del código se realizaron en el entorno de programación PyCharm. Esta combinación permitió modificar y adaptar el algoritmo YOLO, optimizando su rendimiento en términos de precisión, exactitud y costo computacional, con el objetivo de mejorar su desempeño en escenarios de vigilancia reales.

Aunque la documentación oficial de YOLO presenta métricas de rendimiento de sus algoritmos, en este estudio se realizaron pruebas preliminares en un escenario más acorde a las condiciones de trabajo previstas. Para ello, se utilizó un video que contó con la presencia de 58 personas a lo largo de todo su transcurso. El objetivo de estas pruebas fue comprobar el

funcionamiento inicial de las diferentes versiones de YOLOv8 antes de su aplicación en los experimentos principales de la investigación.

Para llevar a cabo el desarrollo del algoritmo se evaluaron diferentes versiones de los algoritmos pree entrenados de la versión YOLOv8, los cuales fueron:

YOLOv8n (Nano): Esta versión es la más liviana de YOLOv8, ya que posee menos parámetros y requiere menos costo computacional para su implementación, esta versión de YOLOv8 tiene una velocidad alta de procesamiento, pero una baja eficiencia al momento de detectar personas complicando así la implementación para el objetivo de la investigación, en la Figura 12 se puede observar su respuesta ante una escena presentada.

Figura 12. Detección YOLOv8n

Nota. Autor

YOLOv8m (Medium): Es la versión de YOLO8 intermedia que combina rapidez y precisión, esta versión posee más parámetros que la versión anterior mejorando la capacidad de detección a distancias más alejadas, pero por ende requiere más costo computacional para su funcionamiento, en la Figura 13 se puede observar su comportamiento.

Figura 13. Detección YOLOv8m

Nota. Autor

YOLOv8x (Extra Large): Es la Versión de YOLOv8 más grande y precisa, tiene una mayor cantidad de parámetros y capas que mejora mucho más la detección de objetos sin importar a que distancia se coloque siempre y cuando se ajusten bien los parámetros del algoritmo, requiere de un costo computacional alto. Los resultados obtenidos con esta versión se muestran en la Figura 14.

Figura 14. Detección YOLOv8x

Nota. Autor

En la Tabla 2 se presentan los resultados obtenidos durante la evaluación de la capacidad de detección, incluyendo el número de aciertos, errores y el tiempo de ejecución correspondiente

a cada versión del algoritmo YOLO. Cabe resaltar que estos valores corresponden a pruebas preliminares, cuyo propósito fue verificar el correcto funcionamiento de los algoritmos antes de su aplicación formal en los experimentos de la investigación.

Para la obtención de estos datos, se procesó un video de 10 minutos de duración, en el cual aparecieron 40 personas a lo largo de toda la grabación. Dicho video fue analizado por las tres versiones del algoritmo (YOLOv8n, YOLOv8m y YOLOv8x), registrándose los resultados mostrados en la tabla.

Tabla 2. Resultados comparativos de versiones del algoritmo YOLO

	YOLOv8n	**YOLOv8m**	**YOLOv8x**
Acierto	23	36	40
Errores	17	4	0
Tiempo de análisis (s)	600	780	1150

Una vez realizada la evaluación de los algoritmos, se seleccionó YOLOv8m por presentar el mejor comportamiento en términos de exactitud, precisión y costo computacional para los requerimientos de la investigación. Las otras versiones fueron descartadas debido a sus limitaciones: YOLOv8n mostró dificultades en la detección de personas a distancias largas y con baja visibilidad, mientras que YOLOv8x, aunque más eficiente, presentó un tiempo de procesamiento considerablemente mayor. En este contexto, YOLOv8m resultó ser la opción más adecuada, ya que sus limitaciones pudieron ser compensadas mediante la modificación de su estructura y la incorporación de condicionales en la programación, de forma que el modelo se enfocara exclusivamente en la detección de personas.

Con el fin de optimizar su desempeño, el algoritmo fue adaptado a las condiciones del proyecto. Se realizaron varias modificaciones al código, entre las cuales destacan:

- Ajuste del tamaño de entrada de las imágenes, asegurando que los cuadros procesados mantuvieran una resolución uniforme compatible con los requerimientos de YOLO.

- Restricción de clases, de modo que el modelo solo ejecutara detecciones sobre la categoría “persona”, descartando otros objetos de la base de datos preentrenada.

- Normalización de imágenes, lo que permitió estandarizar los valores de los píxeles en un rango óptimo para el modelo, mejorando la estabilidad en la detección.

- Conversión a escala de grises, reduciendo la carga computacional al procesar una sola capa de información visual en lugar de las tres del espacio de color RGB. La transformación se implementó mediante la función cv2.cvtColor() de la librería OpenCV:

```
frame1 = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
frame2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
```

  Esta conversión simplifica el procesamiento al reducir la cantidad de datos sin comprometer la capacidad del modelo para detectar personas en condiciones de baja iluminación.
  Como los resultados de detección se realizan sobre imágenes en color, se convirtió el frame de escala de grises a BGR antes de la visualización. Esto permite mostrar las detecciones en la ventana de OpenCV.

```
gray_frame = cv2.cvtColor(frame2, cv2.COLOR_GRAY2BGR)
```

- Binarización de imágenes y segmentación en recuadros, siguiendo la metodología propuesta por la teoría de YOLO, en donde cada cuadro de la imagen es evaluado de forma independiente para identificar la presencia de objetos.

- Umbral mínimo de confianza del 55%, establecido con el fin de reducir falsas detecciones y mostrar únicamente objetos relevantes. Este criterio se aplicó filtrando las predicciones del modelo:

```
results = model(gray_frame, classes=[0, 2], conf=0.55)
```

- Sobreentrenamiento orientado a la categoría “persona”, reforzando la capacidad del modelo para identificar siluetas humanas en diferentes condiciones de iluminación y distancia.

Gracias a estas optimizaciones, se logró una reducción significativa del tiempo de procesamiento, alcanzando un promedio de 530 milisegundos por cuadro, lo que permite mantener un desempeño fluido y en tiempo real sin sacrificar la precisión. El uso de escala de grises resultó especialmente útil en entornos de baja iluminación, mientras que el umbral de confianza incrementó la fiabilidad al descartar detecciones espurias.

La base de datos preentrenada en YOLO, enriquecida con las modificaciones realizadas, permitió validar que las mejoras implementadas contribuyen de forma significativa al cumplimiento de los objetivos planteados en la investigación.

A continuación, se presentan imágenes representativas del funcionamiento del algoritmo YOLO durante las pruebas realizadas:

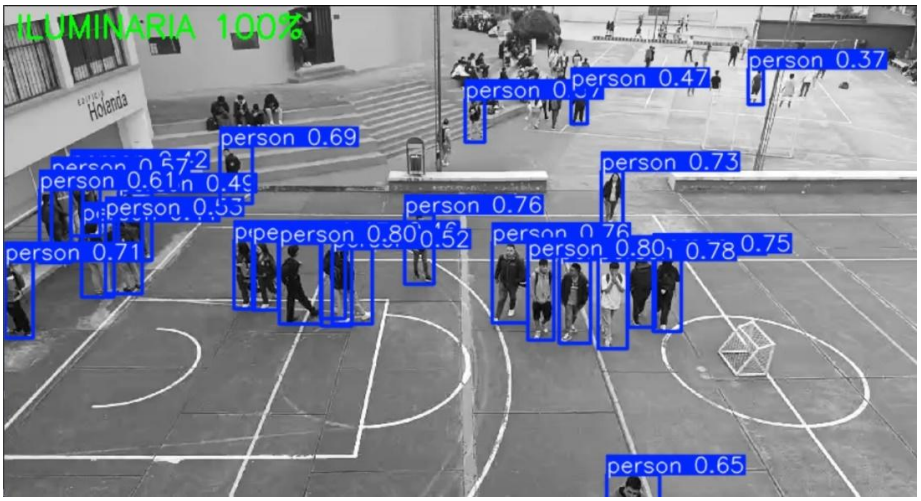


Figura 15. Pruebas realizadas al algoritmo YOLO

Nota. Autor

Figura 16.pruebas realizadas al algoritmo YOLO

Nota. Autor

Figura 17. Pruebas realizadas al algoritmo YOLO

Nota. Autor

Cada una de estas imágenes ilustra cómo el modelo identifica correctamente la presencia de personas, marcando sus ubicaciones mediante cuadros delimitadores y mostrando el porcentaje de confianza asociado a cada detección.

4.1.2 Implementación algoritmo OpenPose.

Para la implementación de OpenPose, se optó por realizar la instalación y configuración en un entorno de ejecución basado en Google Colab, aprovechando sus capacidades de

procesamiento con GPU y la facilidad para gestionar grandes volúmenes de datos de video. Este algoritmo, desarrollado por el *CMU Perceptual Computing Lab*, permite la estimación de poses humanas mediante la detección de puntos clave (keypoints) en diferentes partes del cuerpo, lo que lo convierte en una herramienta ideal para el análisis del comportamiento de personas en escenarios de vigilancia.

El procedimiento comenzó con la montura de Google Drive en el entorno Colab, lo que permitió almacenar y recuperar los resultados de manera persistente. Posteriormente, se descargó y compiló el repositorio oficial de OpenPose desde GitHub, asegurando la inclusión de submódulos y dependencias necesarias para su correcto funcionamiento. Para solventar problemas de compatibilidad con CUDA y CMake, se instalaron versiones específicas de librerías y se realizaron modificaciones en el archivo *CMakeLists.txt* para habilitar las bindings en Python (-DBUILD_python=ON).

Debido a la indisponibilidad temporal de los servidores oficiales, los modelos preentrenados se descargaron desde un repositorio privado, incluyendo los correspondientes a **BODY_25**, que permite la detección de 25 puntos clave del esqueleto humano. Este modelo fue elegido por su mayor detalle en la representación del cuerpo, lo que facilita la identificación de posturas y patrones de movimiento, incluso en condiciones de baja iluminación.

La compilación de OpenPose se realizó con soporte GPU y con la librería cuDNN deshabilitada para maximizar la compatibilidad con el entorno Colab. Esto permitió obtener un equilibrio entre precisión y velocidad de procesamiento, manteniendo una ejecución fluida en la detección de poses sobre secuencias de video.

Durante el desarrollo, se implementaron funciones auxiliares en Python para la visualización de resultados directamente en la notebook de Colab, así como para la exportación de datos a Google Drive. En las pruebas realizadas, se procesaron videos nocturnos con resolución Full HD (1920×1080 píxeles), identificando con éxito la presencia de personas y generando sus correspondientes mapas de esqueleto sobre cada fotograma.

El sistema fue evaluado bajo diferentes condiciones, incluyendo:

- Escenarios de baja visibilidad con iluminación artificial limitada.
- Situaciones con una o varias personas en movimiento.
- Casos con obstrucciones parciales del cuerpo.

Los resultados confirmaron que OpenPose, configurado con el modelo BODY_25, logró identificar correctamente la mayoría de los puntos clave en entornos nocturnos, aunque la calidad de la detección disminuye cuando la visibilidad es muy reducida o cuando la persona se encuentra en movimiento rápido. La ejecución en Colab permitió un tiempo de procesamiento promedio adecuado para análisis offline, lo que valida su uso como herramienta de apoyo en el sistema general de vigilancia planteado en la investigación.

En las siguientes figuras se puede observar como el algoritmo se comporta en diferentes escenarios donde hay iluminación y también donde hay poca visibilidad.

Figura 18. Detección de OpenPose

Nota. Autor

4.1.3 Implementación algoritmo Extracción de Características.

La tercera técnica evaluada en este estudio fue la extracción de características basada en la sustracción de fondo y la comparación geométrica de objetos en movimiento. Este enfoque se implementó con el fin de detectar personas y clasificarlas en base a características como el área y alto. A continuación, se describen los pasos seguidos y los resultados obtenidos.

Etapa de Sustracción de Fondo y Preprocesamiento: Se emplea el modelo cv2.createBackgroundSubtractorMOG2() de OpenCV, el cual permite discriminar las regiones en movimiento respecto al fondo estático.

```
# Inicialización del sustractor de fondo
fgbg = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50,
detectShadows=True)
fgmask = fgbg.apply(frame)
```

Para reducir ruido y mejorar la segmentación, se aplican operaciones morfológicas (apertura, dilatación) y filtros de mediana, seguidos de un proceso de binarización:

```
# Filtrado y limpieza de la máscara
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (4, 4))
fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
fgmask = cv2.dilate(fgmask, kernel, iterations=2)
```

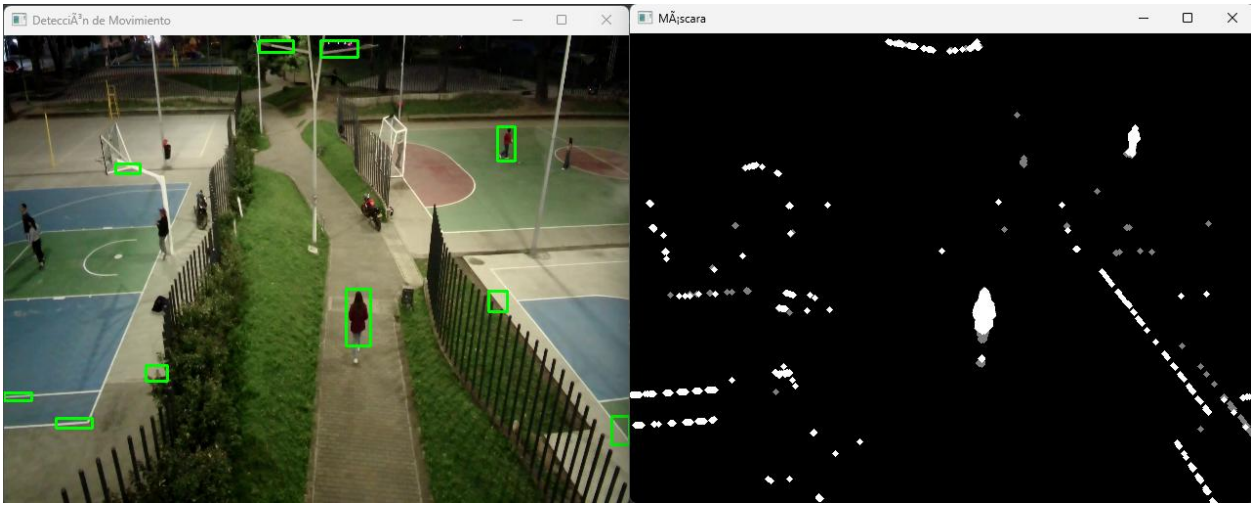


Figura 19. Sustracción de fondo

Nota. Autor

Análisis de Contornos y Clasificación Inicial: Los contornos se extraen y se evalúan únicamente si cumplen con un área mínima. Además, se aplica un criterio de proporcionalidad (altura ≥ ancho) para aproximarse a la morfología humana.

```
# Encuentra contornos
contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

```
for contour in contours:
    area = cv2.contourArea(contour)
    if area > 100:  # Umbral mínimo de área
        x, y, w, h = cv2.boundingRect(contour)
        if h >= w:
            # Posible candidato a persona
            ...
```

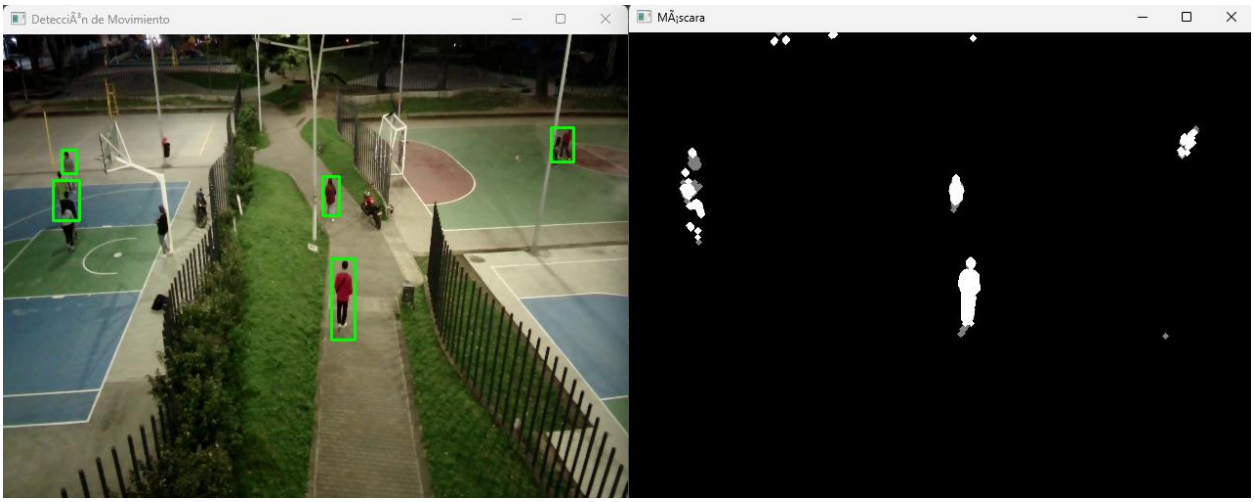


Figura 20. Sustracción de fondo con suavizado

Nota. Autor

Limitaciones en la Extracción de Características: Una vez detectados los objetos en movimiento, se calcularon varias características geométricas utilizando la función cv2.boundingRect(), que permite obtener las dimensiones de los objetos detectados (alto y ancho). A partir de estas dimensiones, se calculó el área de cada objeto. Con ello se realizaron diversas pruebas experimentales que permitieron identificar algunas limitaciones del algoritmo extracción de características en videos capturados desde posiciones elevadas:

Calidad y Resolución: La distancia entre la cámara y los objetos afecta la calidad de la imagen, lo que puede dificultar la extracción precisa de características. A pesar de utilizar videos en 1080p, la pérdida de detalle en objetos pequeños o distantes comprometió la precisión de algunas mediciones y características como el color no ser una opción utilizable.

Variabilidad Dimensional: Inicialmente, se intentó clasificar objetos basándose en movimiento, estructura humana, alto, ancho y área. Sin embargo, se observó que objetos de

diferentes clases (por ejemplo, un perro cercano y una persona distante) podían presentar dimensiones similares y difícil visualizar, lo que aumentaba la posibilidad de falsos positivos.

Dependencia de la Configuración de la Cámara: La posición, resolución y estabilidad de la cámara influyen directamente en los rangos de detección. Cualquier variación en estos parámetros puede requerir una recalibración del sistema para mantener la precisión.

Proceso de Calibración y Extracción de Datos: Para abordar las limitaciones mencionadas, se implementó una solución basada en la calibración del sistema a partir del desplazamiento de un objeto de dimensiones conocidas a lo largo de todo el campo de visión de la cámara. Concretamente, se utilizó un objeto rectangular de dimensiones fijas (80 cm de alto por 40 cm de ancho) que se desplazó por diferentes posiciones, permitiendo medir el alto y área del objeto en función de su posición relativa al centroide del cuadro. Estos datos se almacenaron en un archivo en formato JSON, lo que facilitó la recopilación de información y su posterior análisis. Este enfoque permitió obtener un conjunto de datos que refleja cómo varían las medidas en función de la posición en la imagen, estableciendo así una base sólida para la modelación del tamaño real de los objetos.

Figura 21. Calibración Ex. Características en Parque N1 a 6m

Nota. Autor

Figura 22 . Calibración Ex. Características en Parque N2 a 6m

Nota. Autor

Figura 23. Calibración Ex. Características en Parque N3 a 6m

Nota. Autor

Figura 24. Calibración Ex. Características en Parque N1 a 8m

Nota. Autor

Figura 25. Calibración Ex. Características en Parque N2 a 8m

Nota. Autor

Figura 26. Calibración Ex. Características en Parque N3 a 8m

Nota. Autor

Implementación de la Regresión Lineal Multiple : Una vez recopilados los datos de calibración, se procedió a ajustar un modelo de regresión lineal multiple utilizando la biblioteca Scikit-learn de Python. La finalidad de este modelo es establecer una relación cuantitativa entre las variables medidas (como la posición relativa y las dimensiones en la imagen) y el tamaño real del objeto.

El proceso de ajuste del modelo se llevó a cabo de la siguiente manera:

Preparación de los Datos: Se extrajeron las variables independientes (por ejemplo, la posición en los ejes X e Y) y la variable dependiente (la altura real) a partir del archivo JSON generado durante la calibración.

```
# Cargar datos desde archivo CSV
data_path = 'C:/Users/cristian/Desktop/ObjetoPers.csv'
data = pd.read_csv(data_path)
X = data[['X', 'Y']]
y = data['Altura']
```

Ajuste del Modelo: Para establecer una relación matemática entre las dimensiones observadas en la imagen y la altura real de los objetos, se utiliza un modelo de regresión lineal

múltiple. Este modelo permite predecir la altura estimada de un objeto a partir de las coordenadas de su posición en la imagen (X,Y), obtenidas del centroide del contorno detectado.

El ajuste del modelo se realiza utilizando la biblioteca Scikit-learn de Python, mediante el siguiente procedimiento:

```
# Ajuste de regresión lineal Multiple
model = LinearRegression()
model.fit(X, y)
coef_x, coef_y = model.coef_
intercept = model.intercept_
print(f"Fórmula de regresión: Altura = {coef_x:.4f} * X + {coef_y:.4f} *
Y + {intercept:.4f}")
```

A partir del conjunto de datos recopilado en las pruebas experimentales, se obtienen seis fórmulas de regresión, correspondientes a los diferentes escenarios de prueba (tres parques con dos niveles de altura de cámara cada uno). Cada fórmula refleja cómo varía la altura estimada del objeto en función de su posición dentro del campo visual.

Las ecuaciones finales son las siguientes:

```
altura_estim = -0.0022 * x + 0.1304 * y + 18.6475 # PARQUE 1A6
altura_estim = -0.0045 * x + 0.1903 * y + 39.1776 # PARQUE 2A6
altura_estim = -0.0042 * X + 0.1054 * Y + 59.9340 # PARQUE 3A6
altura_estim = -0.0015 * X + 0.1196 * Y + 35.9127 # PARQUE 1A8
altura_estim = -0.0067 * X + 0.0893 * Y + 39.4174 # PARQUE 2A8
altura_estim =  0.0047 * X + 0.0921 * Y + 37.0130 # PARQUE 3A8
```

Cada una de estas expresiones representa una ecuación de ajuste empírico, en la cual los coeficientes de X y Y determinan la sensibilidad del modelo ante la variación horizontal y vertical de los objetos dentro del plano de visión.

En términos generales, se observa que el coeficiente asociado a Y (posición vertical) tiene un mayor peso en la predicción de la altura, lo cual es consistente con la perspectiva geométrica de la cámara: a medida que el objeto se encuentra más alejado (mayor valor de Y), su altura proyectada en la imagen tiende a reducirse.

Por otro lado, las diferencias entre los interceptos (b) y las pendientes en cada escenario reflejan las condiciones particulares de instalación de las cámaras, como el ángulo de inclinación y la altura de montaje. Estas ecuaciones permiten que el sistema se autoajuste a distintos

entornos, garantizando una estimación más realista de las dimensiones de las personas detectadas en función del escenario analizado.

Integración del Modelo de Regresión en el Sistema de Detección

Con cada una de las fórmulas de regresión obtenidas, se implementa un módulo dentro del sistema de detección que permite estimar el tamaño real de los objetos detectados en función de su posición dentro del plano de la imagen.

Cuando el sistema identifica movimiento, se extraen las coordenadas X y Y del objeto detectado, las cuales se introducen en la ecuación de regresión correspondiente al escenario en análisis. De esta forma, se calcula la altura estimada del objeto en relación con su ubicación en el campo visual de la cámara.

```
margen = 0.2 * altura_estim
es_persona = (h >= altura_estim - margen) and (h <= altura_estim +
margen)

if es_persona:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(frame, "Persona", (x, y - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
```

Para optimizar la clasificación, se incorpora un margen de tolerancia del 20 %, que permite compensar pequeñas variaciones provocadas por la perspectiva o por movimientos irregulares del sujeto. Si la altura real calculada se encuentra dentro de este rango, el objeto se clasifica como *persona*.

Este enfoque híbrido, que combina regresión lineal múltiple y detección geométrica, incrementa la robustez y precisión del sistema, reduciendo las detecciones erróneas derivadas de la similitud dimensional entre distintos objetos (por ejemplo, animales o elementos del entorno).

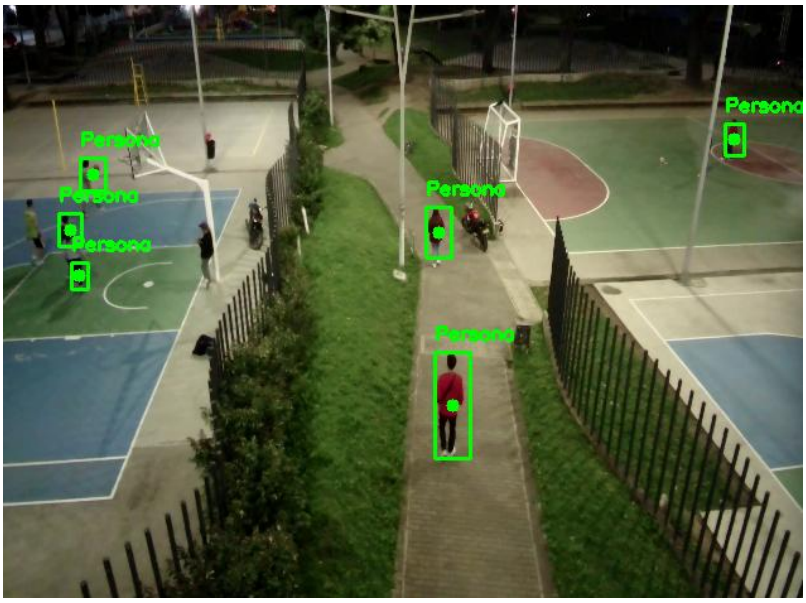


Figura 27. Detección con ecuación de regresión lineal múltiple

Nota. Autor

4.2 Protocolos de experimentación realizados a los algoritmos de visión computacional.

Con el fin de garantizar que la evaluación del desempeño de los algoritmos de visión por computadora implementados (YOLO, OpenPose y técnicas de Extracción de Características) se realice de manera objetiva, se definieron protocolos de experimentación que regulan las condiciones bajo las cuales se desarrollaron las pruebas. Estos protocolos se orientaron a controlar variables externas como la iluminación, la distancia de captura, la resolución del video y la cantidad de personas presentes en la escena, ya que dichas variables influyen directamente en la precisión y en el costo computacional de los modelos.

Para tal fin, los protocolos se estructuraron considerando los siguientes aspectos:

4.2.1 Entorno de ejecución:

Las pruebas de los tres algoritmos se realizaron inicialmente en Google Colab, aprovechando la disponibilidad de GPU para acelerar el procesamiento.

Posteriormente, se efectuaron pruebas comparativas en un entorno local (PyCharm), con el fin de estimar el rendimiento en un computador de estudio y validar las diferencias de consumo computacional.

En ambos entornos se controló la versión de librerías y dependencias, evitando inconsistencias en los resultados.

4.2.2 Condiciones de captura:

Los videos de prueba fueron grabados en un parque en condiciones nocturnas, con resolución 1920 x 1080 píxeles capturada por una cámara de seguridad Logitech C920x HD Pro.

Se establecieron escenarios con distintos niveles de iluminación artificial

Se evaluaron distancias de 6 a 8 metros ya que a este nivel de altura se ubican las cámaras de seguridad para que tengan un alcance optimo y también la altura de los postes que están ubicados en los parques tienen esta altura.

4.2.3 Configuración de los algoritmos:

YOLO: Se probó con el modelo preentrenado YOLOv3, configurado para la detección de personas y objetos relevantes.

OpenPose: Se utilizó el modelo **BODY_25**, que permite identificar 25 puntos clave en el cuerpo humano, asegurando mayor detalle en el análisis postural.

Extracción de características: Se implementaron descriptores como ORB (Oriented FAST and Rotated BRIEF) y SIFT (Scale-Invariant Feature Transform) para el reconocimiento de patrones estructurales en las imágenes.

4.2.4 Métricas de evaluación:

Exactitud y precisión de detección: Se verificó la capacidad de cada algoritmo para identificar correctamente la presencia de personas, posturas o características relevantes en la imagen.

Costo computacional: Se midieron variables como uso de CPU, memoria RAM, tiempo de procesamiento por fotograma (ms/frame) y FPS alcanzados en cada entorno de ejecución.

Robustez en condiciones adversas: Se comparó el desempeño en escenarios con baja iluminación, movimiento rápido de personas y presencia de múltiples sujetos en la misma escena.

4.2.5 Procedimiento experimental:

Cada algoritmo se ejecutó sobre los mismos videos de entrada, asegurando que los resultados fueran comparables bajo condiciones idénticas.

Los experimentos se documentaron con capturas de pantalla de los códigos en ejecución y con salidas gráficas (mapas de esqueleto de OpenPose, bounding boxes de YOLO y correspondencias de puntos en extracción de características).

Los datos recolectados fueron organizados en tablas comparativas, diferenciando el rendimiento entre Colab y PyCharm para estimar el comportamiento real en sistemas de vigilancia locales.

En resumen, el protocolo de experimentación establecido permitió uniformar las condiciones de prueba de los algoritmos, garantizando validez en los resultados obtenidos.

4.3 Rendimiento de los algoritmos de visión computacional reconociendo siluetas humanas para el control de la iluminación pública.

4.3.1 Estudio de precisión y exactitud de los algoritmos

Con el objetivo de evaluar el desempeño de los algoritmos YOLO, OpenPose y Extracción de Características, se diseñó un procedimiento experimental orientado a obtener las métricas de precisión y exactitud bajo diferentes condiciones de prueba. Para ello, se seleccionaron 15 imágenes con presencia de personas y 15 imágenes sin presencia de personas en tres escenarios distintos, considerando además dos alturas de captura: 6 metros y 8 metros. De esta forma, se buscó analizar el comportamiento de los algoritmos frente a variaciones en el entorno y en la perspectiva de detección, incluyendo situaciones con múltiples individuos, con una sola persona o sin presencia de personas.

Cada imagen fue procesada por los tres algoritmos de manera independiente, registrando los siguientes resultados:

- **Verdaderos Positivos (VP):** El sistema detecta una persona y realmente sí hay una persona.
- **Falsos Negativos (FN):** El sistema no detecta una persona, pero sí hay una persona.

- **Falsos Positivos (FP):** El sistema detecta una persona, pero no hay nadie.
- **Verdaderos Negativos (VN):** El sistema dice que no hay persona y realmente no hay nadie.

Con estos resultados se calcularon las métricas fundamentales de evaluación, definidas matemáticamente como:

$$E = \frac{VP + VN}{N} \tag{5}$$

$$P = \frac{VP}{VP+FP} \tag{7}$$

Figura 28. Detección de E. y P. (Extracción de características)

Nota. Autor

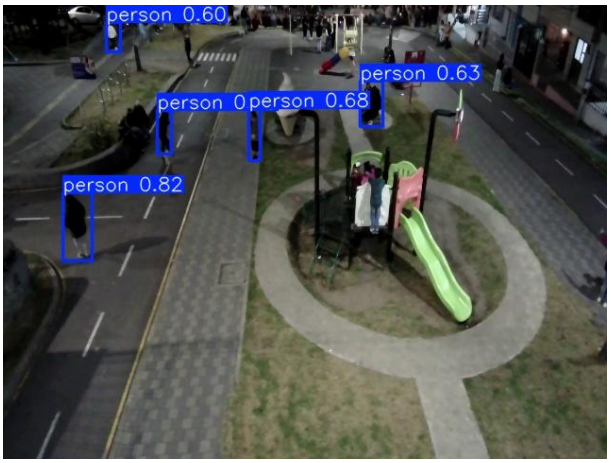


Figura 29. Detección de E. y P. (YOLO)

Nota. Autor

Figura 30. Detección de E. y P. (Openpose)

Nota. Autor

A partir de los valores obtenidos, se construyeron las correspondientes matrices de confusión, que permitieron representar de forma estructurada el comportamiento de los algoritmos frente a cada escenario:

$$\begin{bmatrix} Vp & Fp \\ Fn & Vn \end{bmatrix} \tag{7}$$

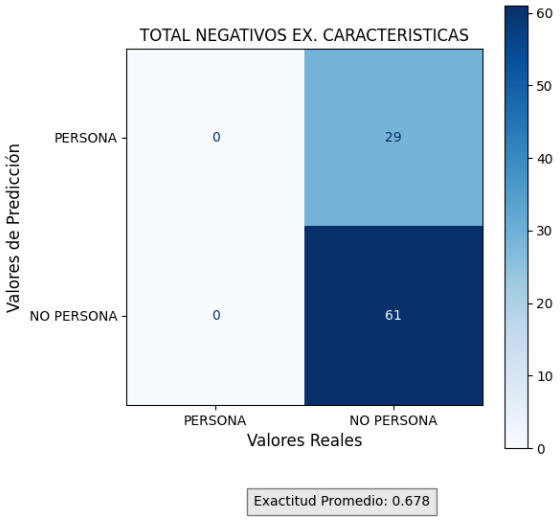


Figura 31. Matriz de confusión extracción de características (imágenes negativas)

Nota. Autor

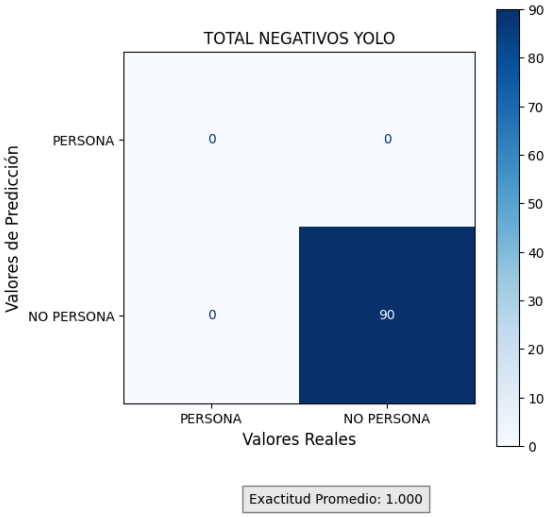


Figura 32. Matriz de confusion YOLO (imágenes negativas)

Nota. Autor

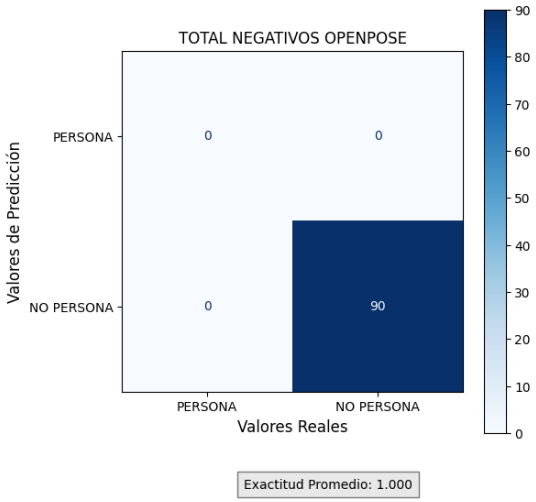


Figura 33. Matriz de confusión Openpose (imágenes negativas)

Nota. Autor

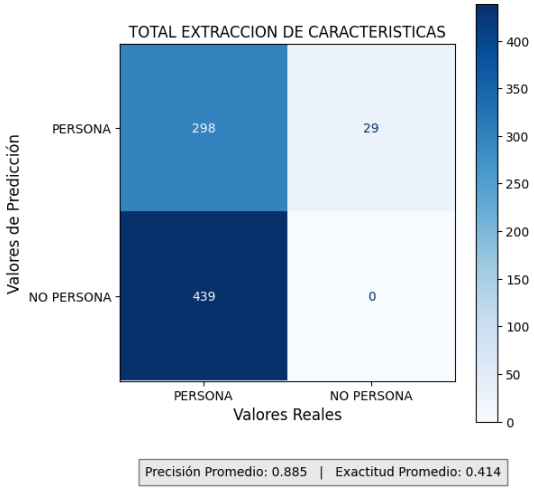


Figura 34. Matriz de confusión extracción de características (imágenes positivas)

Nota. Autor

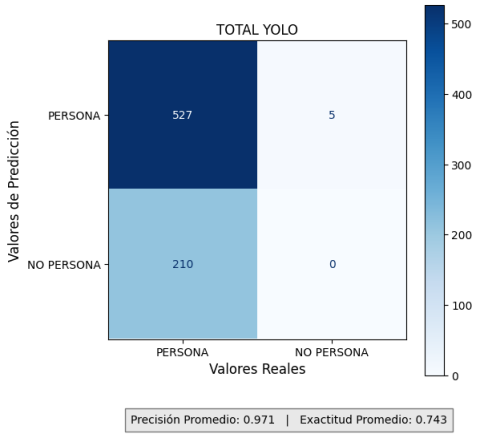


Figura 35. Matriz de confusión YOLO (imágenes positivas)

Nota. Autor

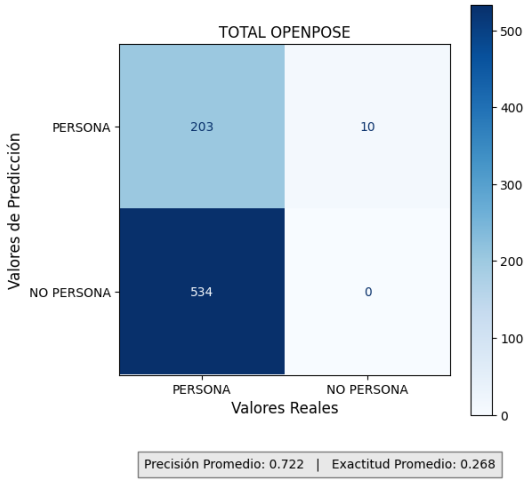


Figura 36. Matriz de confusión Openpose (imágenes positivas)

Nota. Autor

Durante el análisis se identificaron algunas particularidades que afectaron directamente el cálculo de las métricas. En las imágenes positivas (con personas), el valor de VN fue siempre igual a cero, ya que en estos casos no es posible obtener verdaderos negativos. En contraste, en las imágenes negativas (sin personas), el valor de VP resultó necesariamente cero, lo que repercutió en el cálculo de la precisión. Por este motivo, la precisión únicamente fue considerada en imágenes positivas, mientras que la exactitud se calculó tanto para imágenes positivas como negativas. Sin embargo, al combinar ambos conjuntos de resultados, la exactitud tendió a presentar valores elevados en los tres algoritmos, lo que dificultaba la diferenciación clara de su rendimiento. Para resolver esta limitación, el análisis de exactitud se realizó de manera separada en imágenes con y sin personas, lo que permitió obtener una interpretación más detallada de cada caso.

El procedimiento descrito permitió recopilar un total de 90 resultados de imágenes positivas y 90 resultados de imágenes negativas por cada algoritmo, generando una base de datos suficiente para la comparación objetiva entre las tres técnicas. En particular, dentro del conjunto de imágenes con presencia de personas, se registró un total de 737 detecciones realizadas por los algoritmos durante todas las pruebas, lo que aseguró un volumen estadístico adecuado para la evaluación de desempeño y para la elaboración de las métricas finales.

Tabla 3. Resultados de precisión y exactitud de los algoritmos

Algoritmo	Precisión (positivas)	Exactitud (positivas)	Exactitud (negativas)
YOLO	97 %	74 %	100 %
OpenPose	72 %	27 %	100 %
Extracción de Características	89 %	41 %	68 %

4.3.2 Estimación de costo computacional de los algoritmos.

Para determinar la eficiencia de los algoritmos de visión computacional en términos de rendimiento computacional, se analizaron videos capturados en distintos escenarios. En total se utilizaron seis videos, correspondientes a dos alturas de cámara diferentes (6 y 8 metros) por cada uno de los tres parques seleccionados. A partir de estos videos, se extrajeron métricas clave como el tiempo total de análisis, el promedio de cuadros por segundo (FPS), la frecuencia del procesador (CPU), el porcentaje de uso de la CPU y la cantidad total de memoria RAM utilizada

Estas mediciones se realizaron tanto en un entorno local (PyCharm) como en Google Colab, con el objetivo de evaluar la carga computacional de los algoritmos de detección de siluetas humanas. En el caso del algoritmo OpenPose, no fue posible realizar su ejecución en la máquina local debido a que los autores no habían liberado librerías compatibles para una implementación directa en entornos de escritorio. Esta limitación persistió hasta el año 2024, cuando se publicaron versiones más estables. En contraste, los algoritmos basados en YOLO y extracción de características sí pudieron ejecutarse sin inconvenientes en ambos entornos.

Para estimar el comportamiento de OpenPose en el computador de estudio, se empleó un enfoque comparativo: se analizaron los datos reales de los algoritmos que sí se ejecutaron (YOLO y extracción de características) en ambos entornos, y se usaron estos datos para construir un Factor de Transferencia de Rendimiento (FTR) que describe la relación entre el consumo de recursos en Colab y en PyCharm.

4.3.2.1 Estimación de costo computacional algoritmo YOLO

El algoritmo YOLO pudo ejecutarse satisfactoriamente tanto en Google Colab como en el entorno local PyCharm. Los resultados evidencian una marcada diferencia en el rendimiento computacional entre ambos escenarios.

En Google Colab, se obtuvo un desempeño promedio de 19–20 FPS, con tiempos de ejecución relativamente bajos aproximadamente 66 a 241 segundos, dependiendo del video procesado. El consumo de RAM osciló entre 2.0 y 2.1 GB, mientras que el uso de CPU se mantuvo en un rango de 25% a 71%, lo cual refleja una ejecución eficiente gracias a la disponibilidad de GPU y mejores recursos de cómputo.

En contraste, en PyCharm el rendimiento descendió, los FPS promedio fueron de apenas 2.3–2.4, con tiempos de ejecución mucho más altos, que variaron entre 554 y 2115 segundos. El consumo de RAM alcanzó valores cercanos a 8 GB, mientras que el uso de CPU se mantuvo elevado 78 – 81% aproximadamente, lo cual indica un fuerte impacto sobre los recursos del computador de estudio.

En la Tabla 4 se presentan de forma resumida los resultados del costo computacional de YOLO en ambos entornos.

Tabla 4. Resultados de costo computacional de YOLO

Algoritmo	Entorno	Video	Altura	Tiempo (s)	FPS	RAM (MB)	CPU (MHz)	Uso CPU (%)
YOLO	PyCharm	1	6m	1239,32	2,37	8259,52	1609	80,5
YOLO	PyCharm	1	8m	554,37	2,41	8349,39	1609	78,4
YOLO	PyCharm	2	6m	970,6	2,37	8144,25	1609	79,7
YOLO	PyCharm	2	8m	989,78	2,39	8111,24	1609	79,2
YOLO	PyCharm	3	6m	1450	2,33	8098,57	1609	79,3
YOLO	PyCharm	3	8m	2114,94	2,33	7974,12	1609	80,1
YOLO	Colab	1	6m	152,22	19,3	2041,86	2000,14	62,5
YOLO	Colab	1	8m	66,55	20,1	2093,29	2000,14	25,8
YOLO	Colab	2	6m	114,52	20,11	2151,77	2000,14	42,8
YOLO	Colab	2	8m	121,63	19,46	2137,71	2000,14	61,9
YOLO	Colab	3	6m	165,5	20,41	2178,05	2000,14	59,4
YOLO	Colab	3	8m	241,14	20,48	2145,84	2000,14	71,3

4.3.2.2 Estimación de costo computacional algoritmo Extracción de características.

El algoritmo de extracción de características también fue probado en ambos entornos de ejecución. En este caso, los resultados muestran un rendimiento mucho más estable que YOLO,

manteniendo valores de FPS elevados tanto en Colab como en PyCharm, aunque con notorias diferencias en tiempos de ejecución y uso de memoria.

En Google Colab, el algoritmo alcanzó un promedio de aproximadamente 40 FPS, con tiempos de procesamiento cortos entre 63 y 228 segundos. El consumo de memoria RAM fue muy bajo, entre 1.0 y 1.4 GB, y el uso de CPU se mantuvo en un rango variable entre 33 % y 80 %, dependiendo del video analizado.

En PyCharm, si bien el algoritmo mantuvo un rendimiento aceptable en términos de FPS, entre 17 y 20 FPS, los tiempos de ejecución fueron más altos, con valores entre 134 y 486 segundos. El consumo de RAM fue considerablemente mayor, oscilando entre 7.8 y 8.8 GB, aunque el porcentaje de uso de CPU se mantuvo relativamente bajo en comparación con Colab, entre 9 % y 21 %.

En la Tabla 5 se resumen los resultados obtenidos para este algoritmo.

Tabla 5. Resultados de costo computacional Extracción de Características

Algoritmo	Entorno	Video	Altura	Tiempo (s)	FPS	RAM (MB)	CPU (MHz)	Uso CPU (%)
Extracción	PyCharm	1	6m	330,11	17,8	8051,53	1609	14
Extracción	PyCharm	1	8m	134,11	19,95	8666,75	1609	21,4
Extracción	PyCharm	2	6m	226,21	20,36	8842,83	1609	21,4
Extracción	PyCharm	2	8m	235,7	20,08	8651,96	1609	9,8
Extracción	PyCharm	3	6m	385,51	17,52	8763,79	1609	12,7
Extracción	PyCharm	3	8m	486,28	20,31	7824,3	1609	9,4
Extracción	Colab	1	6m	153,33	41,4	1055,46	2000,17	57,9
Extracción	Colab	1	8m	63,69	42,02	1032,4	2000,17	67,1
Extracción	Colab	2	6m	114,29	40,29	1078,42	2000,17	33,4
Extracción	Colab	2	8m	132,79	35,65	1331,88	2000,17	80,5
Extracción	Colab	3	6m	161,73	41,76	1391,35	2000,17	44,4
Extracción	Colab	3	8m	228	43,29	1378,28	2000,17	39,3

4.3.2.3 Estimación de costo computacional algoritmo Open Pose

Para estimar el comportamiento de OpenPose en el computador de estudio, se empleó un enfoque comparativo: se analizaron los datos reales de los algoritmos que sí se ejecutaron (YOLO y extracción de características) en ambos entornos, y se usaron estos datos para construir un FTR que describe la relación entre el consumo de recursos en Colab y en PyCharm.

Dado que OpenPose solo se ejecutó en Colab, este FTR se utilizó como base para proyectar su rendimiento en PyCharm, generando así una estimación razonada y fundamentada.

Ecuaciones de estimación

Dado que OpenPose no pudo ejecutarse en PyCharm, se usaron algoritmos que sí pudieron ejecutarse en ambos entornos (YOLO y extracción de características) para construir Factores de Transferencia de Rendimiento (FTR). Estos factores permiten estimar cómo cambiarían las métricas si OpenPose se ejecutara en la máquina local.

Procedimiento para obtener FTR:

Con los datos obtenidos después de analizar los algoritmos en colab y pycharm se procede a realizar el cálculo de cada métrica M, el FTR se calcula como:

$$FTR = \frac{M_{Pycharm}}{M_{colab}} \quad (2)$$

Donde:

MPyCharm: valor promedio de la métrica al ejecutar un algoritmo en la máquina local.

MColab : valor promedio de la misma métrica al ejecutar el mismo algoritmo en Google Colab.

Los valores promedio se calcularon tomando todos los videos ejecutados con YOLO y extracción de características. A partir de ellos se obtuvieron los siguientes factores:

Tabla 6. Resultados del FTR del algoritmo OpenPose

Métrica	FTR calculado
Tiempo	5.89
FPS	0.12
RAM	7.30
CPU (MHz)	0.80
Uso CPU (%)	0.35

Estos factores indican, por ejemplo, que el tiempo de procesamiento es 5.89 veces mayor en PyCharm comparado con Colab, mientras que el FPS es solo un 12% del obtenido en Colab.

Estimación del rendimiento de OpenPose en PyCharm.

Aplicando las siguientes ecuaciones con los FTR obtenidos:

Tiempo estimado:

$$Tiempo_{pycharm} = Tiempo_{colab} \times FTR_{tiempo} \quad (8)$$

FPS estimado:

$$FPS_{pycharm} = FPS_{colab} \times FTR_{FPS} \quad (9)$$

RAM estimada:

$$RAM_{pycharm} = RAM_{colab} \times FTR_{RAM} \quad (10)$$

CPU estimada:

$$CPU_{pycharm} = CPU_{colab} \times FTR_{CPU} \quad (11)$$

Uso CPU estimado:

$$USOCPU_{pycharm} = USOCPU_{colab} \times FTR_{USOCPU} \quad (12)$$

En la Tabla 7 se muestra los resultados obtenidos al realizar la estimación de OpenPose si se ejecutara en PyCharm:

Tabla 7. Resultados de costo computacional Open Pose

Algoritmo	Entorno	Video	Altura	Tiempo (s)	FPS	RAM (MB)	CPU (MHz)	Uso CPU (%)
OpenPose	PyCharm	1	6m	4319.43	3	94739.03	1600.14	1.75
OpenPose	PyCharm	1	8m	1969.67	3	94739.03	1600.14	0.88
OpenPose	PyCharm	2	6m	3451.72	3	94739.03	1600.14	10.19
OpenPose	PyCharm	2	8m	3512.38	3	94739.03	1600.14	13.33
OpenPose	PyCharm	3	6m	5026.11	3	94739.03	1760.00	13.41
OpenPose	PyCharm	3	8m	4854.07	3	94739.03	1600.14	13.07
OpenPose	Colab	1	6m	733,35	25	12977,95	2000,18	5
OpenPose	Colab	1	8m	334,41	25	12977,95	2000,18	2,5
OpenPose	Colab	2	6m	586,03	25	12977,95	2000,18	29,1
OpenPose	Colab	2	8m	596,33	25	12977,95	2000,18	38,1
OpenPose	Colab	3	6m	853,33	25	12977,95	2200	38,3
OpenPose	Colab	3	8m	824,12	25	12977,95	2000,18	37,34

Estos valores deben ser interpretados como estimaciones aproximadas, no como mediciones exactas. Los valores anómalamente altos en uso de CPU estimado reflejan que la metodología, aunque útil para tener un panorama, puede sobreestimar ciertas métricas si hay mucha disparidad entre algoritmos.

Explicación de funciones utilizadas en la reelección de datos.

psutil.cpu_percent(): Mide el porcentaje de uso actual del CPU por parte del sistema.

psutil.virtual_memory().used: Devuelve la cantidad de memoria RAM utilizada en bytes. Se convierte a megabytes (MB).

psutil.cpu_freq().current: Mide la frecuencia actual de operación del CPU (en MHz).

GPUtil.getGPUs()[0]: Accede a la primera GPU disponible del sistema para extraer métricas como uso y memoria.

gpu.load * 100: Carga actual de la GPU, multiplicada por 100 para obtener porcentaje.

time.time(): Usado para calcular la duración total de la ejecución del algoritmo.

Finalmente, en la Tabla 8 se presenta una comparativa de los resultados promedio obtenidos en términos de costo computacional para los tres algoritmos evaluados YOLO, Extracción de Características y OpenPose, considerando su ejecución tanto en Google Colab como en el entorno local de PyCharm.

Tabla 8. Resultados promedio de costo computacional de los algoritmos

Algoritmo	Entorno	Tiempo Promedio (s)	FPS Promedio	RAM Promedio (MB)	CPU Promedio (MHz)	Uso CPU Promedio (%)
Extracción	Colab	142	40,4	1211	2000	53,8
Extracción	PyCharm	299	19,3	8516	1609	14,8
OpenPose	Colab	25	25	12977	2067	25,4
OpenPose	Pycharm	3	3	94739	1653	8.8
YOLO	Colab	143	19,8	2124	2000	54
YOLO	Pycharm	1219	2,3	8156	1609	79.5

5. Conclusiones

A partir del análisis integral de precisión, exactitud y costo computacional, se puede concluir que los tres algoritmos evaluados presentan comportamientos diferenciados que permiten identificar fortalezas y limitaciones para su aplicación en contextos de alumbrado público inteligente. En primer lugar, el análisis de desempeño en la detección de personas y en escenarios vacíos evidenció que YOLO constituye la opción más robusta, puesto que alcanzó un equilibrio entre exactitud y precisión, garantizando detecciones confiables y oportunas incluso en condiciones de iluminación variable. Esta característica lo convierte en un buen algoritmo para sistemas de activación de iluminación, dado que reduce al mínimo la probabilidad de errores por falsos positivos o por omisión de individuos en la escena.

Por otro lado, el algoritmo de Extracción de Características se mostró como una alternativa intermedia: aunque sus valores de exactitud fueron relativamente bajos en escenarios sin personas (68 %), logró mantener una precisión adecuada y, en todos los casos, fue capaz de detectar al menos una persona. Este resultado sugiere que, si bien no garantiza una cobertura total en escenarios de alta concurrencia, puede ser útil en aplicaciones menos críticas, donde un margen de error moderado no represente un riesgo para la operación del sistema. Además, el impacto del viento sobre la cámara demostró que factores externos pueden influir negativamente en la detección, lo cual plantea la necesidad de implementar soportes más estables o algoritmos de compensación de movimiento para mejorar la robustez de este método.

Por otro lado, OpenPose evidenció ser el algoritmo con mayores limitaciones para el problema planteado. Aunque en Google Colab alcanzó valores relativamente altos de FPS (25), su rendimiento práctico se vio comprometido por diversos factores. En primer lugar, la incompatibilidad para ejecutarse en el entorno local (PyCharm) obligó a recurrir a estimaciones mediante Factores de Transferencia de Rendimiento (FTR). En segundo lugar, su desempeño resultó muy sensible a las condiciones de captura: en escenarios con baja iluminación o con cámaras ubicadas a gran altura, el algoritmo llegó a no detectar personas. A esto se suma una limitación crítica: OpenPose requiere imágenes de muy alta calidad y se ve seriamente afectado cuando las personas se encuentran a largas distancias de la cámara, lo que restringe su aplicabilidad en entornos no controlados.

Las estimaciones de costo computacional realizadas a partir del FTR reflejaron que, en caso de ejecutarse en la máquina de estudio, el tiempo de procesamiento sería entre 4 y 6 veces mayor que en Colab, acompañado de un uso de memoria RAM extremadamente elevado. Esto hace que el algoritmo resulte inviable para entornos de implementación real sin disponer de hardware especializado de alto rendimiento.

En cuanto a la evaluación computacional general, se identificó que los algoritmos presentan un incremento en tiempo de procesamiento y consumo de memoria RAM al pasar de Colab a PyCharm, lo cual fue cuantificado a través de los FTR obtenidos. Este hallazgo resalta la importancia de seleccionar adecuadamente el entorno de ejecución y de optimizar el código, pues la viabilidad práctica de estos algoritmos depende no solo de su precisión en la detección, sino también de su capacidad de operar en tiempo real con los recursos disponibles. Asimismo, el uso de funciones como psutil.cpu_percent(), psutil.virtual_memory().used, psutil.cpu_freq().current, y GPUtil.getGPUs()[0], permitió recolectar de manera sistemática indicadores de carga computacional, otorgando un panorama más completo sobre el impacto de cada algoritmo en el hardware utilizado.

En resumen, los resultados permiten establecer que, para el objetivo de activar un sistema de alumbrado público mediante detección de personas, el algoritmo YOLO es el que ofrece un mejor balance entre rendimiento, precisión y costo computacional, siendo la alternativa más confiable y eficiente para aplicaciones en campo. La Extracción de Características puede considerarse como un respaldo aceptable en contextos menos exigentes, mientras que OpenPose, debido a sus altas demandas de hardware y baja estabilidad en condiciones adversas, se perfila como la opción menos adecuada para este proyecto. Finalmente, la metodología empleada —que incluyó pruebas experimentales en tres parques con diferentes condiciones, ejecución en dos entornos computacionales y estimaciones con FTR— permitió no solo comparar algoritmos en términos de desempeño técnico, sino también generar un marco de análisis reproducible para futuros trabajos orientados a la evaluación de sistemas de visión computacional en escenarios urbanos reales.

6. Trabajos futuros

Como proyección de esta investigación, se plantea una serie de líneas de trabajo que pueden fortalecer y ampliar los resultados alcanzados. En primer lugar, resulta adecuado evaluar versiones más recientes y optimizadas de los algoritmos, incluyendo variantes de YOLO adaptadas a dispositivos de borde (*edge devices*) que permiten operar en hardware de bajo consumo energético, como Raspberry Pi o Jetson Nano. Esto no solo reduciría la dependencia de equipos de alto rendimiento, sino que también acercaría la propuesta a entornos de implementación real en campo.

De igual forma, se recomienda la integración de sensores complementarios (infrarrojos, térmicos o de profundidad) que permitan mejorar la robustez del sistema en condiciones de baja visibilidad, lluvia o neblina, donde las cámaras RGB convencionales presentan limitaciones. La fusión de datos multimodales podría incrementar la precisión de detección y reducir las falsas alarmas.

Finalmente, sería de gran interés vincular este sistema de detección con plataformas de gestión energética basadas en IoT, de modo que no solo se active o regule la iluminación, sino que se optimice el consumo global en función de patrones de ocupación real y predicciones de uso. Este tipo de integración, enmarcada en el concepto de *Smart Cities*, abriría la posibilidad de contar con redes de alumbrado autónomas, adaptativas y sostenibles, contribuyendo directamente a la reducción de costos y a la mitigación del impacto ambiental en las ciudades contemporáneas.

Referencias

[1] Programa de Ingeniería Electrónica, Proyecto Educativo del Programa, Universidad CESMAG, Pasto, 2015.

[2] R. Vizcaya Cárdenas, “Deep Learning para la Detección de Peatones y Vehículos,” pp. 1–122, 2018. [Online]. Available: http://ri.uaemex.mx/handle/20.500.11799/70995

[3] F. Marino, G. Giammanco, y G. G. Trovato, "Adaptive Street Lighting Predictive Control," Energy Procedia, vol. 111, pp. 790–799, Mar. 2017.

[4] R. F. Fernandes y P. N. F. B. Lopes, "Adaptive Street Light Controlling for Smart Cities," International Journal of Advanced Engineering Research and Science (IJAER), vol. 5, no. 10, pp. 65–70, 2018.

[5] V. K. Janga, M. Janga, y P. S. Reddy, "Smart street light monitoring and controlling system using IOT," en 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2017, pp. 1007–1012.

[6] P. Galarza Bravo y J. L. Flores Calero, "Detección de peatones en la noche usando Faster R-CNN e imágenes infrarrojas," Ingeniería de Sistemas, Revista Científica, Tecnológica y Empresarial, vol. 3, no. 1, pp. 55–61, 2018.

[7] G. E. Cueva, A. C. Salazar, J. E. P. Ramos, y V. E. P. Ramos, "Detección de peatones en el día y en la noche usando YOLO-v5," Revista Ciencias Técnicas Agropecuarias, vol. 31, no. 1, 2022.

[8] S. Ren, K. He, R. Girshick, y J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[9] J. Redmon y A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[10] W. Liu et al., "SSD: Single Shot MultiBox Detector," en European Conference on Computer Vision, Amsterdam, Netherlands, 2016, pp. 21–37.

[11] S. Shi, Q. Wang, and X. Chu, “Performance modeling and evaluation of distributed deep learning frameworks on GPUs,” in *2018 IEEE 16th Intl Conf. on Dependable, Autonomic and Secure Computing; 16th Intl Conf. on Pervasive Intelligence and Computing; 4th Intl Conf. on Big Data Intelligence and Computing; and Cyber Science and Technology*

Congress (DASC/PiCom/DataCom/CyberSciTech), Aug. 2018, pp. 949–957.

[12] G. Jocher et al., "YOLOv5 by Ultralytics," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[13] A. I. El-Sayed et al., "Energy efficient smart street lighting system using deep learning and IoT," Energy Reports, vol. 9, pp. 32–43, Sep. 2023

[14] G. G. J. y M. V. S. y S. S. A., "Real-time object detection on edge devices for smart city applications," en *2021 7th International Conference on Smart Structures and Systems (ICSSS)*, Chennai, India, 2021, pp. 1–6.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[16] U. P. Naik, V. Rajesh, R. R. Kumar, and Mohana, "Implementation of YOLOv4 Algorithm for Multiple Object Detection in Image and Video Dataset using Deep Learning and Artificial Intelligence for Urban Traffic Video Surveillance Application," en Proc. 2021 4th Int. Conf. Electr. Comput. Commun. Technol. (ICECCT), Mar. 2021, pp. 1–6, doi: 10.1109/ICECCT52121.2021.9616625.

[17] A. Carpio, J. Carlos, P. Astete, and R. Felipe, De equipos de protección eléctrica en personas, orientado a sistemas de vigilancia basados en cámaras IP, 2024.

[18] Departamento de Ingeniería Electrónica, Grado en Ciencia de Datos. Trabajo Fin de Grado: Estimación de Pose y sus Aplicaciones, [s.l.], Revista Colombiana de Tecnologías de Avanzada (RCTA), 2024.

[19] M. S. Benjumea Mesa, "Propuesta para la implementación del sistema 'LED' para la iluminación pública en Antioquia," Escuela de Ingeniería, [s.l.], 2019.

[20] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," MILA, Université de Montréal & AIRLab, Politecnico di Milano, Jan. 12, 2018. [Online]. Available: https://arxiv.org/abs/1603.07285

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," en Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[22] M. Yaseen, "What is YOLOv8: An In-Depth Exploration of the Internal Features of the

Next-Generation Object Detector," arXiv preprint arXiv:2408.15857, Aug. 29, 2024.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," en Lecture Notes in Computer Science (LNCS), vol. 8693, ECCV 2014, Zurich, Switzerland, Sept. 6–12, 2014, pp. 740–755, Springer, Cham, doi: 10.1007/978-3-319-10602-1_48.

[24] R. Borja-Robalino, A. Monleón-Getino, and J. Rodellar, "Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning," Rev. Ibérica Sist. Tecnol. Inf., no. June, pp. 184–196, 2022.

[25] V. P. N. Díaz Narváez, "El concepto de ciencia como sistema, el positivismo, neopositivismo y las investigaciones cuantitativas y cualitativas," Salud Uninorte, vol. 30, no. 1, pp. 1–9, 2014.

[26] Ministerio de Minas y Energía, Reglamento Técnico de Iluminación y Alumbrado Público (RETILAP), República de Colombia, 2010.

[27] T. Carneiro, R. V. M. Da Nobrega, T. Nepomuceno, G. B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," IEEE Access, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

7. Anexos

7.1 Códigos

7.1.1 Código YOLO:

```
import cv2
from ultralytics import YOLO
import time
import psutil
import os

# Inicializar modelo y video
model = YOLO('yolov8m.pt')
cap = cv2.VideoCapture('VIDEOS ANALISIS/OTROS/U1A6.mp4')

# Parámetros
change_threshold = 25
skip_frames = 1
frame_count = 0
person_frames = 0

# Medición inicial de recursos
start_global = time.time()
process = psutil.Process(os.getpid())
initial_memory = process.memory_info().rss / (1024 ** 2)  # MB

# Leer primer frame
ret, frame1 = cap.read()
frame1 = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)

while True:
    for _ in range(skip_frames):
        ret = cap.grab()
        frame_count += 1

    ret, frame2_color = cap.read()
    if not ret:
        break

    frame2 = cv2.cvtColor(frame2_color, cv2.COLOR_BGR2GRAY)

    # Detectar movimiento
    diff = cv2.absdiff(frame1, frame2)
    _, thresh = cv2.threshold(diff, change_threshold, 255,
cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=2)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) > 0:
        start_frame = time.time()
        results = model(frame2_color, classes=[0], conf=0.50)
```

```
        annotated_frame = results[0].plot()

        # Verificar detección de persona
        person_detected = any(d.cls == 0 for d in results[0].boxes)
        if person_detected:
            person_frames += 1

        # Mostrar frame
        resized_frame = cv2.resize(annotated_frame, (1280, 720))
        cv2.imshow("Detecciones", resized_frame)

        elapsed_frame = time.time() - start_frame
        print(f"Frame {frame_count} - Tiempo de análisis:
{elapsed_frame:.2f} segundos")

    frame1 = frame2

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Liberar recursos
cap.release()
cv2.destroyAllWindows()

# Métricas finales
end_global = time.time()
total_time = end_global - start_global
fps = frame_count / total_time if total_time > 0 else 0
memory_usage = process.memory_info().rss / (1024 ** 2)
cpu_freq = psutil.cpu_freq().current
cpu_usage = psutil.cpu_percent()
ram_total = psutil.virtual_memory().used / (1024 ** 2)
cpu_temp = get_cpu_temp()

# Mostrar resumen
print(f"""
📊 RESUMEN GENERAL DEL ANÁLISIS
⏱️ Tiempo total de análisis: {total_time:.2f} segundos
🚀 FPS promedio: {fps:.2f}
🧠 Memoria usada (RAM): {memory_usage - initial_memory:.2f} MB
🖥️ Frecuencia CPU: {cpu_freq:.2f} MHz
🔄 Uso de CPU: {cpu_usage:.2f} %
💽 RAM total usada en el sistema: {ram_total:.2f} MB
📸 Frames analizados: {frame_count}
🧍 Frames con detección de personas: {person_frames}
""")
```

7.1.2 Codigo Openpose :

```
#Mounting to our Google Drive
from google.colab import drive
```

```
drive.mount('/content/drive')

#The following where we define the new directory in the mounted Google
Drive account.
!mkdir -p '/content/drive/My Drive/dataOutput'

#Importing our OS and defining our HOME_PATH for later use.
import os
OPENPOSE_PATH="./openpose/"
HOME_PATH='./'

!echo $HOME_PATH

#@title
!pip install ffmpeg-python
from os.path import exists, join, basename, splitext

def show_local_mp4_video(file_name, width=640, height=480):
  import io
  import base64
  from IPython.display import HTML
  video_encoded = base64.b64encode(io.open(file_name, 'rb').read())
  return HTML(data='''<video width="{0}" height="{1}" alt="test"
controls>
                        <source src="data:video/mp4;base64,{2}"
type="video/mp4" />
                     </video>'''.format(width, height,
video_encoded.decode('ascii')))


# see: https://github.com/CMU-Perceptual-Computing-
Lab/openpose/issues/949
# install new CMake becaue of CUDA10
!wget -q https://cmake.org/files/v3.17/cmake-3.17.2-Linux-x86_64.tar.gz
!tar xfz cmake-3.17.2-Linux-x86_64.tar.gz --strip-components=1 -C
/usr/local

# install system dependencies
!apt-get -qq install -y libatlas-base-dev libprotobuf-dev libleveldb-dev
libsnappy-dev libhdf5-serial-dev protobuf-compiler libgflags-dev
libgoogle-glog-dev liblmdb-dev opencl-headers ocl-icd-opencl-dev
libviennacl-dev

git_repo_url = 'https://github.com/CMU-Perceptual-Computing-
Lab/openpose.git'
project_name = splitext(basename(git_repo_url))[0]
!rm -rf openpose
# clone openpose
!git clone -q --depth 1 $git_repo_url
# --recursive necessary in the line below, as otherwise you can
(sometimes) get "lpthreads" errors in cmake ("undefined reference to
`pthread_create'" etc). See, for example,
https://github.com/facebookarchive/caffe2/issues/1234
!sed -i 's/execute_process(COMMAND git checkout --recursive master
WORKING_DIRECTORY
${CMAKE_SOURCE_DIR}\/3rdparty\/caffe)/execute_process(COMMAND git
checkout f019d0dfe86f49d1140961f8c7dec22130c83154 WORKING_DIRECTORY
```

```
${CMAKE_SOURCE_DIR}\/3rdparty\/caffe)/g' openpose/CMakeLists.txt
!cd openpose && git submodule update --init --recursive --remote


#Downloading the models from a private drive location since the original
servers are down.
!gdown 1_z6tj09sfHw833tJfzmpRuLSxdzWMQ8Y -O models.zip
#Unpacking the models.zip
!unzip -o models.zip -d openpose

# use 'sed' to comment out the line in the OpenPose repo that downloads
the model from the failed link
! sed -i 's/executeShInItsFolder "getModels.sh"/# executeShInItsFolder
"getModels.sh"/g'
./openpose/scripts/ubuntu/install_openpose_JetsonTX2_JetPack3.1.sh
! sed -i 's/executeShInItsFolder "getModels.sh"/# executeShInItsFolder
"getModels.sh"/g'
./openpose/scripts/ubuntu/install_openpose_JetsonTX2_JetPack3.3.sh
! sed -i 's/download_model("BODY_25"/# download_model("BODY_25"/g'
./openpose/CMakeLists.txt
! sed -i 's/78287B57CF85FA89C03F1393D368E5B7/#
78287B57CF85FA89C03F1393D368E5B7/g' ./openpose/CMakeLists.txt
! sed -i 's/download_model("body (COCO)"/# download_model("body
(COCO)"/g' ./openpose/CMakeLists.txt
! sed -i 's/5156d31f670511fce9b4e28b403f2939/#
5156d31f670511fce9b4e28b403f2939/g' ./openpose/CMakeLists.txt
! sed -i 's/download_model("body (MPI)"/# download_model("body (MPI)"/g'
./openpose/CMakeLists.txt
! sed -i 's/2ca0990c7562bd7ae03f3f54afa96e00/#
2ca0990c7562bd7ae03f3f54afa96e00/g' ./openpose/CMakeLists.txt
! sed -i 's/download_model("face"/# download_model("face"/g'
./openpose/CMakeLists.txt
! sed -i 's/e747180d728fa4e4418c465828384333/#
e747180d728fa4e4418c465828384333/g' ./openpose/CMakeLists.txt
! sed -i 's/download_model("hand"/# download_model("hand"/g'
./openpose/CMakeLists.txt
! sed -i 's/a82cfc3fea7c62f159e11bd3674c1531/#
a82cfc3fea7c62f159e11bd3674c1531/g' ./openpose/CMakeLists.txt

! ls -lha /usr/lib/x86_64-linux-gnu

# build openpose
# CUDA
# !cd openpose && rm -rf build || true && mkdir build && cd build &&
cmake .. -DUSE_CUDNN=OFF && make -j`nproc`
#!cd openpose && rm -rf build || true && mkdir build && cd build &&
cmake .. && make -j`nproc`
# CPU
# !cd openpose && rm -rf build || true && mkdir build && cd build &&
cmake -DGPU_MODE=CPU_ONLY -DUSE_MKL=OFF .. && cmake --build . --config
Release && make -j`nproc`

!cd openpose && rm -rf build || true && mkdir build

cmake_file='/content/openpose/CMakeLists.txt'
!cd openpose && sed -i 's/-DBUILD_python=OFF/-DBUILD_python=ON/g'
$cmake_file
```

```
!cd openpose && sed -i 's/-DBUILD_python_layer=OFF/-
DBUILD_python_layer=ON/g' $cmake_file

!cd openpose && sed -i 's/option(BUILD_PYTHON "Build OpenPose python."
OFF)/option(BUILD_PYTHON "OpenPose python." ON)\noption(BUILD_BIN_FOLDER
"Copy 3rd-party DLL files." ON)/g' $cmake_file

# CUDA
!cd openpose && cd build && cmake .. -DUSE_CUDNN=OFF -
DGENERATE_PYTHON_BINDINGS:BOOL="1" -DPYTHON_LIBRARY='/usr/lib/x86_64-
linux-gnu/libpython3.10.so' && make -j`nproc`

 # CPU
# !cd openpose && cd build && cmake .. -DGPU_MODE=CPU_ONLY -DUSE_MKL=OFF
-DGENERATE_PYTHON_BINDINGS:BOOL="1" -DPYTHON_LIBRARY='/usr/lib/x86_64-
linux-gnu/libpython3.10.so'
# !cd openpose && cd build && make -j`nproc`


from google.colab import drive
import os
import time
import psutil
import re

# Montar Google Drive
drive.mount('/content/drive')

# Ruta del video
video_path = '/content/ParqueN3A8.mp4'
OPENPOSE_PATH = '/content/openpose'  # Actualiza esta ruta si cambias

# Verificar si el archivo existe
if not os.path.exists(video_path):
    raise FileNotFoundError(f"No se encontró el archivo en la ruta:
{video_path}")
if not os.path.exists(OPENPOSE_PATH):
    raise FileNotFoundError(f"No se encontró la carpeta de OpenPose en:
{OPENPOSE_PATH}")

# Guardar estado inicial
start_time = time.time()
cpu_freq_start = psutil.cpu_freq().current
mem_start = psutil.virtual_memory().used / (1024 * 1024)

# Ejecutar OpenPose
!cd $OPENPOSE_PATH && rm -f ../openpose.avi
!cd $OPENPOSE_PATH && chmod -R 755 './build/'
!cd $OPENPOSE_PATH && ./build/examples/openpose/openpose.bin --video
$video_path --write_json ../outputJson --display 0 --write_video
../openpose.avi --number_people_max 100

# Medir tiempo final
end_time = time.time()
elapsed_time = end_time - start_time

# Convertir el resultado a MP4
```

```
!ffmpeg -y -loglevel info -i openpose.avi output.mp4

# Estado final de recursos
cpu_freq_end = psutil.cpu_freq().current
cpu_percent = psutil.cpu_percent(interval=1)
mem_end = psutil.virtual_memory().used / (1024 * 1024)
mem_total = psutil.virtual_memory().total / (1024 * 1024)

# Calcular métricas
mem_used = max(0.0, mem_end - mem_start)
cpu_freq = (cpu_freq_start + cpu_freq_end) / 2

# Mostrar resumen
print(f"""
📊 RESUMEN GENERAL DEL ANÁLISIS
⏱️ Tiempo total de análisis: {elapsed_time:.2f} segundos
🧠 Memoria usada (RAM): {mem_used:.2f} MB
🖥️ Frecuencia CPU: {cpu_freq:.2f} MHz
🔄 Uso de CPU: {cpu_percent:.2f} %
💽 RAM total del sistema: {mem_total:.2f} MB
""")
```

7.1.3 Código extraccion de caracteristicas :

```
import cv2
import os
import tkinter as tk
from tkinter import messagebox
import pandas as pd

# Crear carpetas para capturas
carpeta_sin = 'IMAGENES EYP/Extraccion de
caracteristicas/Negativos/DeteccionSIN/PARQUE 2A8'
carpeta_con = 'IMAGENES EYP/Extraccion de
caracteristicas/Negativos/Deteccion/PARQUE 2A8'
os.makedirs(carpeta_sin, exist_ok=True)
os.makedirs(carpeta_con, exist_ok=True)

# Video
video_path = "VIDEOS ANALISIS/Parque solo/PARQUE 2A8.mp4"
cap = cv2.VideoCapture(video_path)

# Inicializar sustractor de fondo MOG2
fgbg = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50,
detectShadows=True)

# Crear ventana de Tkinter oculta
root = tk.Tk()
root.withdraw()

# Lista para guardar resultados
resultados = []
```

```
# Interfaz para pedir VP, VN, FP, FN con campos en 0 por defecto
def pedir_datos(nombre_imagen):
    ventana = tk.Toplevel()
    ventana.title(f"Ingreso de datos - {nombre_imagen}")
    ventana.geometry("300x250")
    ventana.grab_set()

    etiquetas = ['VP', 'VN', 'FP', 'FN']
    entradas = {}

    for i, etiqueta in enumerate(etiquetas):
        tk.Label(ventana, text=f"{etiqueta}:").grid(row=i, column=0,
padx=10, pady=5, sticky='e')
        entrada = tk.Entry(ventana)
        entrada.insert(0, "0")  # Valor por defecto
        entrada.grid(row=i, column=1, padx=10, pady=5)
        entradas[etiqueta] = entrada

    resultado = {}

    def confirmar():
        try:
            for et in etiquetas:
                valor = int(entradas[et].get())
                if valor < 0:
                    raise ValueError
                resultado[et] = valor
            ventana.destroy()
        except ValueError:
            messagebox.showerror("Error", "Por favor, ingrese solo
números enteros válidos.")

    tk.Button(ventana, text="Aceptar", command=confirmar).grid(row=5,
column=0, columnspan=2, pady=15)
    ventana.wait_window()
    return resultado


# Variables de control
contador_imagenes = 0
limite = 15

while cap.isOpened() and contador_imagenes < limite:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.resize(frame, (640, 480))
    copia_sin_deteccion = frame.copy()

    fgmask = fgbg.apply(frame)

    # Morfología para limpiar la máscara
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (4, 4))
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
    fgmask = cv2.dilate(fgmask, kernel, iterations=2)
```

```
    contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 100:
            x, y, w, h = cv2.boundingRect(contour)
            if h >= w:
                cx = x + w // 2
                cy = y + h // 2

                #altura_estim = -0.0022 * cx + 0.1304 * cy + 18.6475  #
PARQUE 1A6
                #altura_estim = -0.0045 * cx + 0.1903 * cy + 39.1776  #
PARQUE 2A6
                #altura_estim = -0.0042 * cx + 0.1054 * cy + 59.9340  #
PARQUE 3A6
                #altura_estim = -0.0015 * cx + 0.1196 * cy + 35.9127 #
PARQUE 1A8
                altura_estim = -0.0067 * cx + 0.0893 * cy + 39.4174 #
PARQUE 2A8
                #altura_estim =  0.0047 * cx + 0.0921 * cy + 37.0130 #
PARQUE 3A8

                margen = 0.2 * altura_estim
                es_persona = (h >= altura_estim - margen) and (h <=
altura_estim + margen)

                if es_persona:
                    cv2.rectangle(frame, (x, y), (x + w, y + h), (0,
255, 0), 2)
                    cv2.putText(frame, "Persona", (x, y - 10),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,
0), 2)

    cv2.imshow("Video", frame)

    key = cv2.waitKey(30) & 0xFF

    if key == 27:  # Tecla ESC para salir
        break
    elif key == ord('o'):  # Tecla "o" para capturar
        nombre_img = f"captura_{contador_imagenes + 1:02d}.jpg"
        ruta_sin = os.path.join(carpeta_sin, f"sin_{nombre_img}")
        ruta_con = os.path.join(carpeta_con, f"con_{nombre_img}")

        cv2.imwrite(ruta_sin, copia_sin_deteccion)
        cv2.imwrite(ruta_con, frame)

        cv2.imshow("Detección", frame)
        cv2.waitKey(1)

        # Pedir datos
        datos = pedir_datos(nombre_img)
        vp, vn, fp, fn = datos['VP'], datos['VN'], datos['FP'],
datos['FN']
```

```
        # Calcular métricas
        precision = vp / (vp + fp) if (vp + fp) > 0 else 0
        exactitud = (vp + vn) / (vp + vn + fp + fn) if (vp + vn + fp +
fn) > 0 else 0

        resultados.append({
            'Imagen': nombre_img,
            'VP': vp,
            'VN': vn,
            'FP': fp,
            'FN': fn,
            'Precisión': round(precision, 3),
            'Exactitud': round(exactitud, 3),
        })

        contador_imagenes += 1

cap.release()
cv2.destroyAllWindows()

# Guardar resultados en Excel
df = pd.DataFrame(resultados)
df.to_excel('IMAGENES EYP/Extraccion de caracteristicas/Negativos/PARQUE
2A8.xlsx', index=False)
```

7.1.3.1 Código detección de rectángulo (regresión lineal múltiple):

```
import cv2
import numpy as np
import csv
import os
import pandas as pd
from sklearn.linear_model import LinearRegression

# === CONFIGURACIÓN ===
video_path = 'C:/Users/cristian/Desktop/tesis/VIDEOS
ANALISIS/CUADRO/Parque3A8.MP4'
output_path = 'C:/Users/cristian/Desktop/tesis/VIDEOS
ANALISIS/CUADRO/DATOS/Parque3A8.csv'

# Rango de color amarillo en HSV (ajustado a tu necesidad)
lower_yellow = np.array([0, 200, 130])
upper_yellow = np.array([164, 255, 255])

# Crear carpeta si no existe
os.makedirs(os.path.dirname(output_path), exist_ok=True)

# Lista para guardar los datos
data = []

# Cargar video
cap = cv2.VideoCapture(video_path)
frame_count = 0

while cap.isOpened():
```

```
    ret, frame = cap.read()
    if not ret:
        break
    frame_count += 1

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, lower_yellow, upper_yellow)
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        c = max(contours, key=cv2.contourArea)
        x, y, w, h = cv2.boundingRect(c)

        if h > 10:  # Filtro para eliminar ruido
            # Centro del rectángulo (no centroide del contorno)
            cx = x + w // 2
            cy = y + h // 2

            data.append([frame_count, cx, cy, h])  # Guardar datos

            # Dibujar el rectángulo y el punto central
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 255),
2)
            cv2.circle(frame, (cx, cy), 5, (0, 0, 255), -1)

            # Mostrar texto
            label = f"X:{cx} Y:{cy} H:{h}"
            cv2.putText(frame, label, (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 255), 2)

    # Mostrar el frame redimensionado
    resized_frame = cv2.resize(frame, (1280, 720))
    cv2.imshow('Detección de Objeto Amarillo', resized_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# === GUARDAR DATOS ===
with open(output_path, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Frame', 'X', 'Y', 'Altura'])
    writer.writerows(data)

cap.release()
cv2.destroyAllWindows()
print(f"Datos guardados en: {output_path}")

# === REGRESIÓN MULTILINEAL ===
df = pd.read_csv(output_path)
X_vars = df[['X', 'Y']]
y_var = df['Altura']

model = LinearRegression()
model.fit(X_vars, y_var)

print(f"\nFórmula obtenida:")
```

```
print(f"Altura = {model.intercept_:.2f} + ({model.coef_[0]:.2f} * X) +
({model.coef_[1]:.2f} * Y)")
```

7.2 Tablas de precisión y exactitud

7.2.1 Imágenes positivas

Extracción de características:

Tabla 9. Extracción de características (Imágenes positivas)

Imagen	VP	VN	FP	FN	Precisión	Exactitud
captura_01.jpg	1	0	0	0	1	1
captura_02.jpg	2	0	0	1	1	0,66666667
captura_03.jpg	2	0	2	1	0,5	0,4
captura_04.jpg	0	0	1	1	0	0
captura_05.jpg	1	0	3	1	0,25	0,2
captura_06.jpg	1	0	0	1	1	0,5
captura_07.jpg	1	0	2	2	0,33333333	0,2
captura_08.jpg	1	0	1	3	0,5	0,2
captura_09.jpg	3	0	0	1	1	0,75
captura_01.jpg	6	0	0	5	1	0,54545455
captura_02.jpg	4	0	2	7	0,66666667	0,30769231
captura_03.jpg	2	0	1	10	0,66666667	0,15384615
captura_04.jpg	4	0	0	8	1	0,33333333
captura_05.jpg	1	0	0	9	1	0,1
captura_06.jpg	6	0	0	5	1	0,54545455
captura_01.jpg	0	0	0	3	0	0
captura_02.jpg	1	0	0	2	1	0,33333333
captura_03.jpg	3	0	0	0	1	1
captura_04.jpg	3	0	0	0	1	1
captura_05.jpg	4	0	0	0	1	1
captura_06.jpg	1	0	0	3	1	0,25
captura_07.jpg	3	0	1	0	0,75	0,75
captura_08.jpg	1	0	1	0	0,5	0,5
captura_09.jpg	1	0	0	0	1	1
captura_01.jpg	2	0	0	9	1	0,18181818
captura_02.jpg	2	0	1	9	0,66666667	0,16666667
captura_03.jpg	3	0	0	4	1	0,42857143
captura_04.jpg	0	0	2	9	0	0
captura_05.jpg	2	0	0	7	1	0,22222222
captura_07.jpg	5	0	0	5	1	0,5
captura_01.jpg	2	0	0	2	1	0,5
captura_02.jpg	5	0	0	4	1	0,55555556

captura_03.jpg	3	0	2	5	0,6	0,3
captura_04.jpg	6	0	1	5	0,85714286	0,5
captura_05.jpg	4	0	0	8	1	0,33333333
captura_06.jpg	9	0	0	4	1	0,69230769
captura_07.jpg	3	0	0	8	1	0,27272727
captura_08.jpg	1	0	0	8	1	0,11111111
captura_09.jpg	2	0	3	8	0,4	0,15384615
captura_10.jpg	3	0	1	13	0,75	0,17647059
captura_01.jpg	2	0	0	2	1	0,5
captura_02.jpg	5	0	0	4	1	0,55555556
captura_03.jpg	3	0	2	5	0,6	0,3
captura_04.jpg	6	0	1	5	0,85714286	0,5
captura_05.jpg	4	0	0	8	1	0,33333333
captura_01.jpg	8	0	0	9	1	0,47058824
captura_02.jpg	7	0	0	7	1	0,5
captura_03.jpg	12	0	0	8	1	0,6
captura_04.jpg	3	0	0	9	1	0,25
captura_05.jpg	5	0	0	5	1	0,5
captura_06.jpg	1	0	0	8	1	0,11111111
captura_07.jpg	1	0	0	8	1	0,11111111
captura_08.jpg	5	0	0	5	1	0,5
captura_09.jpg	5	0	0	6	1	0,45454545
captura_10.jpg	5	0	0	6	1	0,45454545
captura_11.jpg	5	0	0	5	1	0,5
captura_12.jpg	9	0	0	6	1	0,6
captura_01.jpg	5	0	0	5	1	0,5
captura_02.jpg	8	0	0	4	1	0,66666667
captura_03.jpg	6	0	1	4	0,85714286	0,54545455
captura_01.jpg	2	0	0	5	1	0,28571429
captura_02.jpg	1	0	0	7	1	0,125
captura_03.jpg	1	0	0	8	1	0,11111111
captura_04.jpg	1	0	0	9	1	0,1
captura_05.jpg	3	0	0	7	1	0,3
captura_06.jpg	2	0	0	8	1	0,2
captura_07.jpg	4	0	0	7	1	0,36363636
captura_08.jpg	1	0	0	6	1	0,14285714
captura_09.jpg	0	0	0	7	0	0
captura_10.jpg	2	0	0	8	1	0,2
captura_11.jpg	4	0	0	4	1	0,5
captura_12.jpg	2	0	0	4	1	0,33333333
captura_13.jpg	3	0	0	3	1	0,5
captura_14.jpg	3	0	0	5	1	0,375
captura_15.jpg	2	0	0	6	1	0,25

captura_01.jpg	3	0	0	6	1	0,33333333
captura_02.jpg	4	0	0	5	1	0,44444444
captura_03.jpg	3	0	0	4	1	0,42857143
captura_04.jpg	3	0	0	5	1	0,375
captura_05.jpg	4	0	0	2	1	0,66666667
captura_06.jpg	3	0	0	4	1	0,42857143
captura_07.jpg	5	0	0	3	1	0,625
captura_08.jpg	4	0	0	2	1	0,66666667
captura_09.jpg	6	0	1	2	0,85714286	0,66666667
captura_10.jpg	2	0	0	6	1	0,25
captura_11.jpg	3	0	0	3	1	0,5
captura_12.jpg	4	0	0	4	1	0,5
captura_13.jpg	5	0	0	4	1	0,55555556
captura_14.jpg	6	0	0	3	1	0,66666667
captura_15.jpg	3	0	0	2	1	0,6
	298	0	29	439	0,88457672	0,4141902

Openpose :

Tabla 10.Openpose (imágenes positivas)

Imagen	VP	VN	FP	FN	Precisión	Exactitud
captura_01.jpg	0	0	0	1	0	0
captura_02.jpg	0	0	0	3	0	0
captura_03.jpg	0	0	0	3	0	0
captura_04.jpg	0	0	0	1	0	0
captura_05.jpg	0	0	0	2	0	0
captura_06.jpg	0	0	0	2	0	0
captura_07.jpg	2	0	0	1	1	0,66666667
captura_08.jpg	2	0	0	2	1	0,5
captura_09.jpg	2	0	0	2	1	0,5
captura_01.jpg	6	0	0	5	1	0,54545455
captura_02.jpg	4	0	0	7	1	0,36363636
captura_03.jpg	8	0	0	4	1	0,66666667
captura_04.jpg	7	0	0	5	1	0,58333333
captura_05.jpg	5	0	0	5	1	0,5
captura_06.jpg	5	0	0	6	1	0,45454545
captura_01.jpg	2	0	0	1	1	0,66666667
captura_02.jpg	2	0	0	1	1	0,66666667
captura_03.jpg	2	0	3	1	0,4	0,33333333
captura_04.jpg	2	0	0	1	1	0,66666667
captura_05.jpg	2	0	0	2	1	0,5

captura_06.jpg	0	0	0	4	0	0
captura_07.jpg	0	0	0	3	0	0
captura_08.jpg	0	0	0	1	0	0
captura_09.jpg	0	0	0	1	0	0
captura_01.jpg	2	0	1	9	0,66666667	0,16666667
captura_02.jpg	3	0	0	8	1	0,27272727
captura_03.jpg	0	0	0	7	0	0
captura_04.jpg	2	0	0	7	1	0,22222222
captura_05.jpg	1	0	0	8	1	0,11111111
captura_06.jpg	1	0	0	9	1	0,1
captura_01.jpg	2	0	0	2	1	0,5
captura_02.jpg	2	0	0	7	1	0,22222222
captura_03.jpg	3	0	0	5	1	0,375
captura_04.jpg	2	0	0	9	1	0,18181818
captura_05.jpg	2	0	0	10	1	0,16666667
captura_06.jpg	7	0	0	6	1	0,53846154
captura_07.jpg	3	0	0	8	1	0,27272727
captura_08.jpg	3	0	0	6	1	0,33333333
captura_09.jpg	4	0	0	6	1	0,4
captura_01.jpg	1	0	0	15	1	0,0625
captura_02.jpg	2	0	0	2	1	0,5
captura_03.jpg	2	0	0	7	1	0,22222222
captura_04.jpg	3	0	0	5	1	0,375
captura_05.jpg	2	0	0	9	1	0,18181818
captura_06.jpg	2	0	0	10	1	0,16666667
captura_01.jpg	0	0	0	17	0	0
captura_02.jpg	3	0	0	11	1	0,21428571
captura_03.jpg	4	0	1	16	0,8	0,19047619
captura_04.jpg	0	0	0	12	0	0
captura_05.jpg	0	0	0	10	0	0
captura_06.jpg	1	0	0	8	1	0,11111111
captura_07.jpg	3	0	2	6	0,6	0,27272727
captura_08.jpg	3	0	0	7	1	0,3
captura_09.jpg	3	0	0	8	1	0,27272727
captura_01.jpg	3	0	0	8	1	0,27272727
captura_02.jpg	1	0	0	9	1	0,1
captura_03.jpg	1	0	0	14	1	0,06666667
captura_04.jpg	2	0	0	8	1	0,2
captura_05.jpg	5	0	0	7	1	0,41666667
captura_06.jpg	3	0	0	7	1	0,3
captura_01.jpg	4	0	0	3	1	0,57142857
captura_02.jpg	6	0	1	2	0,85714286	0,66666667
captura_03.jpg	6	0	0	3	1	0,66666667

captura_04.jpg	5	0	1	5	0,83333333	0,45454545
captura_05.jpg	5	0	0	5	1	0,5
captura_06.jpg	5	0	0	5	1	0,5
captura_07.jpg	5	0	1	6	0,83333333	0,41666667
captura_08.jpg	0	0	0	7	0	0
captura_09.jpg	0	0	0	7	0	0
captura_01.jpg	0	0	0	10	0	0
captura_02.jpg	0	0	0	8	0	0
captura_03.jpg	0	0	0	6	0	0
captura_04.jpg	0	0	0	6	0	0
captura_05.jpg	0	0	0	8	0	0
captura_06.jpg	0	0	0	8	0	0
captura_01.jpg	3	0	0	6	1	0,33333333
captura_02.jpg	1	0	0	8	1	0,11111111
captura_03.jpg	1	0	0	6	1	0,14285714
captura_04.jpg	2	0	0	6	1	0,25
captura_05.jpg	3	0	0	3	1	0,5
captura_06.jpg	0	0	0	7	0	0
captura_07.jpg	2	0	0	6	1	0,25
captura_08.jpg	2	0	0	4	1	0,33333333
captura_09.jpg	2	0	0	6	1	0,25
captura_01.jpg	3	0	0	5	1	0,375
captura_02.jpg	3	0	0	3	1	0,5
captura_03.jpg	5	0	0	3	1	0,625
captura_04.jpg	4	0	0	5	1	0,44444444
captura_05.jpg	3	0	0	6	1	0,33333333
captura_06.jpg	1	0	0	4	1	0,2
	203	0	10	534	0,7221164	0,26807305

YOLO :

Tabla 11. YOLO (Imágenes positivas)

Imagen	VP	VN	FP	FN	Precisión	Exactitud
sin_captura_01.jpg	0	0	0	1	0	0
sin_captura_02.jpg	2	0	0	1	1	0,66666667
sin_captura_03.jpg	1	0	0	2	1	0,33333333
sin_captura_04.jpg	1	0	0	0	1	1
sin_captura_05.jpg	0	0	0	2	0	0
sin_captura_06.jpg	1	0	0	1	1	0,5
sin_captura_07.jpg	2	0	0	1	1	0,66666667
sin_captura_08.jpg	3	0	0	1	1	0,75
sin_captura_09.jpg	4	0	0	0	1	1

sin_captura_10.jpg	7	0	0	4	1	0,63636364
sin_captura_11.jpg	5	0	0	6	1	0,45454545
sin_captura_12.jpg	8	0	0	4	1	0,66666667
sin_captura_13.jpg	10	0	0	2	1	0,83333333
sin_captura_14.jpg	9	0	0	1	1	0,9
sin_captura_15.jpg	8	0	0	3	1	0,72727273
sin_captura_01.jpg	2	0	0	1	1	0,66666667
sin_captura_02.jpg	3	0	0	0	1	1
sin_captura_03.jpg	2	0	0	1	1	0,66666667
sin_captura_04.jpg	3	0	0	0	1	1
sin_captura_05.jpg	4	0	0	0	1	1
sin_captura_06.jpg	4	0	0	0	1	1
sin_captura_07.jpg	3	0	0	0	1	1
sin_captura_08.jpg	1	0	0	0	1	1
sin_captura_09.jpg	1	0	0	0	1	1
sin_captura_10.jpg	8	0	0	3	1	0,72727273
sin_captura_11.jpg	10	0	0	1	1	0,90909091
sin_captura_12.jpg	7	0	0	0	1	1
sin_captura_13.jpg	9	0	0	0	1	1
sin_captura_14.jpg	9	0	0	0	1	1
sin_captura_15.jpg	7	0	0	3	1	0,7
sin_captura_01.jpg	4	0	0	0	1	1
sin_captura_02.jpg	6	0	0	3	1	0,66666667
sin_captura_03.jpg	5	0	0	3	1	0,625
sin_captura_04.jpg	9	0	0	2	1	0,81818182
sin_captura_05.jpg	7	0	0	5	1	0,58333333
sin_captura_06.jpg	12	0	0	1	1	0,92307692
sin_captura_07.jpg	3	0	0	8	1	0,27272727
sin_captura_08.jpg	6	0	0	3	1	0,66666667
sin_captura_09.jpg	7	0	0	3	1	0,7
sin_captura_10.jpg	5	0	1	11	0,83333333	0,29411765
sin_captura_11.jpg	4	0	0	0	1	1
sin_captura_12.jpg	6	0	0	3	1	0,66666667
sin_captura_13.jpg	5	0	0	3	1	0,625
sin_captura_14.jpg	9	0	0	2	1	0,81818182
sin_captura_15.jpg	7	0	0	5	1	0,58333333
sin_captura_01.jpg	6	0	0	11	1	0,35294118
sin_captura_02.jpg	4	0	0	10	1	0,28571429
sin_captura_03.jpg	9	0	0	11	1	0,45
sin_captura_04.jpg	2	0	0	10	1	0,16666667
sin_captura_05.jpg	3	0	0	7	1	0,3
sin_captura_06.jpg	3	0	0	6	1	0,33333333
sin_captura_07.jpg	5	0	0	4	1	0,55555556

sin_captura_08.jpg	8	0	0	2	1	0,8
sin_captura_09.jpg	5	0	0	6	1	0,45454545
sin_captura_10.jpg	4	0	0	7	1	0,36363636
sin_captura_11.jpg	6	0	0	4	1	0,6
sin_captura_12.jpg	9	0	0	6	1	0,6
sin_captura_13.jpg	10	0	0	0	1	1
sin_captura_14.jpg	8	0	0	4	1	0,66666667
sin_captura_15.jpg	7	0	0	3	1	0,7
sin_captura_01.jpg	7	0	0	0	1	1
sin_captura_02.jpg	8	0	0	0	1	1
sin_captura_03.jpg	7	0	0	2	1	0,77777778
sin_captura_04.jpg	8	0	0	2	1	0,8
sin_captura_05.jpg	10	0	0	0	1	1
sin_captura_06.jpg	9	0	0	1	1	0,9
sin_captura_07.jpg	8	0	0	3	1	0,72727273
sin_captura_08.jpg	6	0	0	1	1	0,85714286
sin_captura_09.jpg	7	0	0	0	1	1
sin_captura_10.jpg	7	0	0	3	1	0,7
sin_captura_11.jpg	8	0	0	0	1	1
sin_captura_12.jpg	6	0	0	0	1	1
sin_captura_13.jpg	6	0	0	0	1	1
sin_captura_14.jpg	7	0	0	1	1	0,875
sin_captura_15.jpg	4	0	0	4	1	0,5
sin_captura_01.jpg	7	0	0	2	1	0,77777778
sin_captura_02.jpg	6	0	0	3	1	0,66666667
sin_captura_03.jpg	7	0	1	0	0,875	0,875
sin_captura_04.jpg	6	0	0	2	1	0,75
sin_captura_05.jpg	6	0	0	0	1	1
sin_captura_06.jpg	5	0	0	2	1	0,71428571
sin_captura_07.jpg	8	0	2	0	0,8	0,8
sin_captura_08.jpg	6	0	1	0	0,85714286	0,85714286
sin_captura_09.jpg	8	0	0	0	1	1
sin_captura_10.jpg	8	0	0	0	1	1
sin_captura_11.jpg	5	0	0	1	1	0,83333333
sin_captura_12.jpg	7	0	0	1	1	0,875
sin_captura_13.jpg	9	0	0	0	1	1
sin_captura_14.jpg	8	0	0	1	1	0,88888889
sin_captura_15.jpg	5	0	0	0	1	1
	527	0	5	210	0,97072751	0,74279829

7.2.2 Imágenes negativas

Extracción de características:

Tabla 12.Extracción de características (Imágenes negativas)

Imagen	VP	VN	FP	FN	Exactitud
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	0	1	0	0
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	0	1	0	0
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	0	1	0	0
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
captura_01.jpg	0	0	0	0	0
captura_02.jpg	0	1	0	0	1
captura_03.jpg	0	1	0	0	1
captura_04.jpg	0	0	1	0	0
captura_05.jpg	0	1	0	0	1
captura_06.jpg	0	1	0	0	1
captura_07.jpg	0	0	0	0	0

captura_08.jpg	0	1	0	0	1
captura_09.jpg	0	1	0	0	1
captura_10.jpg	0	1	0	0	1
captura_11.jpg	0	1	0	0	1
captura_08.jpg	0	1	0	0	1
captura_09.jpg	0	1	0	0	1
captura_10.jpg	0	1	0	0	1
captura_11.jpg	0	1	0	0	1
captura_01.jpg	0	1	0	0	1
captura_02.jpg	0	1	0	0	1
captura_03.jpg	0	1	0	0	1
captura_04.jpg	0	0	1	0	0
captura_05.jpg	0	0	1	0	0
captura_06.jpg	0	1	0	0	1
captura_07.jpg	0	0	1	0	0
captura_08.jpg	0	1	0	0	1
captura_09.jpg	0	1	0	0	1
captura_10.jpg	0	1	0	0	1
captura_11.jpg	0	0	1	0	0
captura_12.jpg	0	1	0	0	1
captura_13.jpg	0	1	0	0	1
captura_14.jpg	0	0	1	0	0
captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	0	1	0	0
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	0	1	0	0
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1

sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	0	1	0	0
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1

Openpose:

Tabla 13. Openpose (Imágenes negativas)

Imagen	VP	VN	FP	FN	Exactitud
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1

sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1

sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
	0	90	0	0	1

YOLO:

Tabla 14. YOLO (Imágenes negativas)

Imagen	VP	VN	FP	FN	Exactitud
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1

sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1

sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
sin_captura_01.jpg	0	1	0	0	1
sin_captura_02.jpg	0	1	0	0	1
sin_captura_03.jpg	0	1	0	0	1
sin_captura_04.jpg	0	1	0	0	1
sin_captura_05.jpg	0	1	0	0	1
sin_captura_06.jpg	0	1	0	0	1
sin_captura_07.jpg	0	1	0	0	1
sin_captura_08.jpg	0	1	0	0	1
sin_captura_09.jpg	0	1	0	0	1
sin_captura_10.jpg	0	1	0	0	1
sin_captura_11.jpg	0	1	0	0	1
sin_captura_12.jpg	0	1	0	0	1
sin_captura_13.jpg	0	1	0	0	1
sin_captura_14.jpg	0	1	0	0	1
sin_captura_15.jpg	0	1	0	0	1
	0	90	0	0	1

UNIVERSIDAD CESMAG NIT: 800.109.387-7 VIGILADA MINEDUCACIÓN	**CARTA DE ENTREGA TRABAJO DE GRADO O TRABAJO DE APLICACIÓN – ASESOR(A)**	**CÓDIGO:** AAC-BL-FR-032
		VERSIÓN: 1
		FECHA: 09/JUN/2022

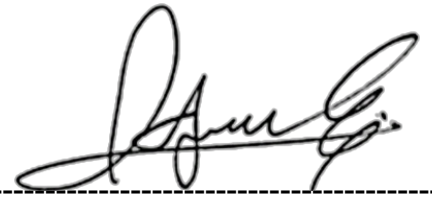
San Juan de Pasto, 20 de Noviembre del 2025

Biblioteca
REMIGIO FIORE FORTEZZA OFM. CAP.
Universidad CESMAG
Pasto

Saludo de paz y bien.

Por medio de la presente se hace entrega del Trabajo de Grado / Trabajo de Aplicación denominado Estudio Comparativo de Algoritmos de Visión Computacional Orientados al Control de Iluminación Publica, presentado por el (los) autor(es) Christian Eduardo Bastidas Carlosi y Jhon Alexander Urbina Burbano del Programa Académico Ingeniería Electrónica al correo electrónico biblioteca.trabajosdegrado@unicesmag.edu.co. Manifiesto como asesor(a), que su contenido, resumen, anexos y formato PDF cumple con las especificaciones de calidad, guía de presentación de Trabajos de Grado o de Aplicación, establecidos por la Universidad CESMAG, por lo tanto, se solicita el paz y salvo respectivo.

Atentamente,



José Camilo Eraso Guerrero
C.C: 1.085.290.125
Programa académico: Ingeniería Electrónica
Teléfono de contacto: 318 793 0965
Correo electrónico: jceraso@unicesmag.edu.co

INFORMACIÓN DEL (LOS) AUTOR(ES)	
Nombres y apellidos del autor: Christian Eduardo Bastidas Carlosi	**Documento de identidad:** 1.004.236.371
Correo electrónico: cebastidas.6371@unicesmag.edu.co	**Número de contacto:** 3117776606
Nombres y apellidos del autor: Jhon Alexander Urbina Burbano	**Documento de identidad:** 1.004.543.019
Correo electrónico: Urbinajhonalex1@gmail.com	**Número de contacto:** 3195469679
Nombres y apellidos del autor:	**Documento de identidad:**
Correo electrónico:	**Número de contacto:**
Nombres y apellidos del asesor: José Camilo Eraso Guerrero	**Documento de identidad:** 1.085.290.125
Correo electrónico: jceraso@unicesmag.edu.co	**Número de contacto:** 318 793 0965
Título del trabajo de grado: Estudio Comparativo de Algoritmos de Visión Computacional Orientados al Control de Iluminación Publica	
Facultad y Programa Académico: Facultad de ingeniería, Ingeniería Electrónica	

En mi (nuestra) calidad de autor(es) y/o titular (es) del derecho de autor del Trabajo de Grado o de Aplicación señalado en el encabezado, confiero (conferimos) a la Universidad CESMAG una licencia no exclusiva, limitada y gratuita, para la inclusión del trabajo de grado en el repositorio institucional. Por consiguiente, el alcance de la licencia que se otorga a través del presente documento, abarca las siguientes características:

a) La autorización se otorga desde la fecha de suscripción del presente documento y durante todo el término en el que el (los) firmante(s) del presente documento conserve (mos) la titularidad de los derechos patrimoniales de autor. En el evento en el que deje (mos) de tener la titularidad de los derechos patrimoniales sobre el Trabajo de Grado o de Aplicación, me (nos) comprometo (comprometemos) a informar de manera inmediata sobre dicha situación a la Universidad CESMAG. Por consiguiente, hasta que no exista comunicación escrita de mi(nuestra) parte informando sobre dicha situación, la Universidad CESMAG se encontrará debidamente habilitada para continuar con la publicación del Trabajo de Grado o de Aplicación dentro del repositorio institucional. Conozco(conocemos) que esta autorización podrá revocarse en cualquier momento, siempre y cuando se eleve la solicitud por escrito para dicho fin ante la Universidad CESMAG. En estos eventos, la Universidad CESMAG cuenta con el plazo de un mes después de recibida la petición, para desmarcar la visualización del Trabajo de Grado o de Aplicación del repositorio institucional.
b) Se autoriza a la Universidad CESMAG para publicar el Trabajo de Grado o de Aplicación en formato digital y teniendo en cuenta que uno de los medios de publicación del repositorio institucional es el internet, acepto(amos) que el Trabajo de Grado o de Aplicación circulará con un alcance mundial.

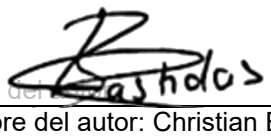
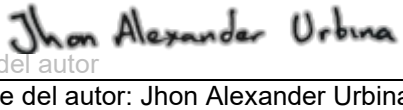
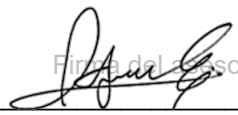
c) Acepto (aceptamos) que la autorización que se otorga a través del presente documento se realiza a título gratuito, por lo tanto, renuncio(amos) a recibir emolumento alguno por la publicación, distribución, comunicación pública y/o cualquier otro uso que se haga en los términos de la presente autorización y de la licencia o programa a través del cual sea publicado el Trabajo de grado o de Aplicación.
d) Manifiesto (manifestamos) que el Trabajo de Grado o de Aplicación es original realizado sin violar o usurpar derechos de autor de terceros y que ostento(amos) los derechos patrimoniales de autor sobre la misma. Por consiguiente, asumo(asumimos) toda la responsabilidad sobre su contenido ante la Universidad CESMAG y frente a terceros, manteniéndose indemne de cualquier reclamación que surja en virtud de la misma. En todo caso, la Universidad CESMAG se compromete a indicar siempre la autoría del escrito incluyendo nombre de(los) autor(es) y la fecha de publicación.
e) Autorizo(autorizamos) a la Universidad CESMAG para incluir el Trabajo de Grado o de Aplicación en los índices y buscadores que se estimen necesarios para promover su difusión. Así mismo autorizo (autorizamos) a la Universidad CESMAG para que pueda convertir el documento a cualquier medio o formato para propósitos de preservación digital.

NOTA: En los eventos en los que el trabajo de grado o de aplicación haya sido trabajado con el apoyo o patrocinio de una agencia, organización o cualquier otra entidad diferente a la Universidad CESMAG. Como autor(es) garantizo(amos) que he(hemos) cumplido con los derechos y obligaciones asumidos con dicha entidad y como consecuencia de ello dejo(dejamos) constancia que la autorización que se concede a través del presente escrito no interfiere ni transgrede derechos de terceros.

Como consecuencia de lo anterior, autorizo(autorizamos) la publicación, difusión, consulta y uso del Trabajo de Grado o de Aplicación por parte de la Universidad CESMAG y sus usuarios así:

- Permito(permitimos) que mi(nuestro) Trabajo de Grado o de Aplicación haga parte del catálogo de colección del repositorio digital de la Universidad CESMAG por lo tanto, su contenido será de acceso abierto donde podrá ser consultado, descargado y compartido con otras personas, siempre que se reconozca su autoría o reconocimiento con fines no comerciales.

En señal de conformidad, se suscribe este documento en San Juan de Pasto a los 20 días del mes de Noviembre del año 2025

 Firma del autor	 Firma del autor
Nombre del autor: Christian Eduardo Bastidas Carlosi	Nombre del autor: Jhon Alexander Urbina Burbano
Firma del autor	Firma del autor
Nombre del autor:	Nombre del autor:
 Firma del asesor Nombre del asesor: José Camilo Eraso Guerrero	