UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE TELECOMUNICACIONES

Telecomunicaciones

TITULO DEL TRABAJO DE TITULACIÓN

Detección de Aphidoidea (Pulgón) en plantíos de banano mediante visión por computador y protocolo de transmisión en tiempo real

AUTOR

GUSTAVO RICARDO CARBO LAÍNEZ

PROFESOR TUTOR

ING. GÓMEZ MORALES ÓSCAR, MGT.

LA LIBERTAD – ECUADOR

2024

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, Ph.D.
DIRECTOR DE LA CARRERA

Ing. Gómez Morales Oscar, Mgt.
DOCENTE TUTOR

Ing. Fernando Vinicio Chamba, Mgt.
DOCENTE ESPECIALISTA

Ing. Luis Amaya Fariño, Mgt.
DOCENTE GUÍA UIC

Ing. Corina Gonzabay De La A. Mgt.
SECRETARIA

AGRADECIMIENTO

Las siguientes líneas las dedico a las personas que fueron fundamentales, pues me dieron los ánimos y el aliento para poder culminar mi tema de tesis.

A mi abuelita y hermanas quienes fueron una parte fundamental, puesto que me dieron los alientos a continuar mi carrera a pesar de los percances y problemas, con lo que tuvimos que lidiar, pero por sobre todo a mi Madre, quien siempre quiso que yo culminase mis estudios, para ser una parte fundamental de la sociedad además de todo a mi Tutor académico quien me dio su apoyo, al momento de realizar mi tema de tesis.

DECLARATORIA DE RESPONSABILIDAD

Quien sustenta, **GUSTAVO RICARDO CARBO LAÍNEZ** con C.I. 0927300616, estudiante de la carrera de telecomunicaciones, declaro que el trabajo de Titulación presentado a la unidad de integración Curricular cuyo tema es **'Detección de Aphidoidea (Pulgón) en plantíos de banano mediante visión por computador y protocolo de transmisión en tiempo real'** corresponde y es de la exclusividad de los autores y pertenece al patrimonio intelectual de la Universidad Estatal Península de Santa Elena.

Atentamente

GUSTAVO RICARDO CARBO LAÍNEZ

C.I. 0927300616

APROBACIÓN DEL TUTOR

En mi calidad de Docente Tutor del trabajo de Integración Curricular, **'Detección de Aphidoidea (Pulgón) en plantíos de banano mediante visión por computador y protocolo de transmisión en tiempo real'**, elaborado por Gustavo Ricardo Carbo Laínez, estudiante de la carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la universidad Estatal Península de Santa Elena, previo a la obtención de su título de Ingeniero en Telecomunicaciones me permito declarar que tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos los aspectos y listo para la sustentación de su trabajo.

Atentamente

OSCAR WLADIMIR GOMEZ MORALES

Firmado digitalmente por OSCAR WLADIMIR GOMEZ MORALES
DN: cn=OSCAR WLADIMIR GOMEZ MORALES gn=OSCAR WLADIMIR c=EC l=TAMBILLO ou=Certificado Persona Natural EC (FIRMA) e=oscargomezm@yahoo.com
Motivo:Soy el autor de este documento
Ubicación:
Fecha:2024-11-12 15:14-05:00

ING. GÓMEZ MORALES ÓSCAR, MGT.

Docente Tutor

RESUMEN

El sistema propuesto de visión por computador y transmisión en tiempo real, para la detección de Aphidoidea (Pulgones) en las plantaciones de banano tiene como punto central el sistema de transmisión en tiempo real, dado que el sistema planteado tiene como objetivo la detección de este tipo de insectos los cuales son perjudiciales para los plantíos, considerado uno de los recursos que más impacto tiene en la economía del país.

Para el tema propuesto se tomaron varios puntos, entre los cuales el más rescatable, es la eficiencia al momento de monitorear los cultivos, esto no solo incluye al sistema de visión por computador, sino también al sistema aplicado para la transmisión en tiempo real, por medio de este canal será por donde se enviaran las imágenes para su análisis posterior, estos dos sistemas al aplicarse conjuntamente permiten a los operadores tener un control más eficiente de las Aphidoidea (Pulgones), lo cual se traduce de forma directa en sistemas más eficientes de producción en cultivos, con un mejor resultado en lo que respecta a calidad de producto. Este enfoque combina la automatización de la detección con la capacidad de transferir datos inmediatamente, lo cual es esencial para un control eficaz de plagas en un entorno agrícola.

Palabras clave: Agricultura Inteligente, Aphidoidea, Monitoreo de Cultivos, Transmisión en Tiempo Real, Visión por Computador.

ABSTRACT

The proposed computer vision and real-time transmission system for the detection of Aphidoidea (Aphids) in banana plantations has as its central point the real-time transmission system, given that the proposed system aims to detect this type of insects which are harmful to crops, considered one of the resources that has the greatest impact on the country's economy.

For the proposed topic, several points were taken to highlight, among which the most notable is the efficiency when monitoring crops. This not only includes the computer vision system, but also the system applied for real-time transmission, through this channel will be where the images will be sent for subsequent analysis, these two systems when applied together allow operators to have more efficient control of Aphidoidea (Aphids), which directly translates into more efficient systems. efficient crop production, with a better result in terms of product quality. This approach combines automation of detection with the ability to immediately transfer data, which is essential for effective pest control in an agricultural environment.

Keywords: Smart Agriculture, Aphidoidea, Crop Monitoring, Real-Time Transmission, Computer Vision.

DECLARACIÓN

El presente trabajo de Titulación, de la materia de Integración Curricular cuyo título es '**Detección de Aphidoidea (Pulgón) en plantíos de banano mediante visión por computador y protocolo de transmisión en tiempo real**', declaro que la concepción, análisis y resultados son originales y aportan a la actividad educativa en el are de Telecomunicaciones.

Atentamente

GUSTAVO RICARDO CARBO LAÍNEZ

C.I. 0927300616

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El modelo para la detección de plagas, especialmente pulgones, se desarrolló considerando su uso potencial en el manejo de cultivos, así como el impacto en la eficiencia de los trabajadores en el monitoreo del crecimiento de las plantas. Inicialmente, se evaluaron su idoneidad en el mundo real y su potencial de mejora, además de identificar posibles complicaciones durante la implementación.

En resumen, el primer objetivo es crear una base de datos confiable que los usuarios puedan manipular fácilmente. Esto permitirá realizar cambios si se encuentra otra plaga en cualquier momento, debido a que el sistema de IA, llamado YOLOv5, es muy flexible, puede modificarse fácilmente según las necesidades de los usuarios.
La culminación del proyecto incluye no sólo sistemas de visión por ordenador y modelos entrenados con YOLOv5, sino también el uso de drones y sistemas de conectividad RTMP para transmitir información a través de estos sistemas para su posterior análisis.

El resultado final es un sistema capaz de analizar imágenes de alta resolución para detectar plagas, en particular pulgones, el sistema permitirá a los operadores de las fincas Agroecológicas Esmeralda identificar plagas en sus cultivos en una etapa temprana, facilitando la implementación de métodos de control seleccionados que ayudarán a prevenir pérdidas financieras o de producción.

CAPÍTULO I

1) FUNDAMENTACIÓN

1.2)Antecedentes

La visión por computador es una tecnología ampliamente utilizada en diversos campos, tanto civiles como militares, gracias a sus avanzadas capacidades, como el reconocimiento facial y la detección de objetos. Aunque no es una tecnología reciente, su aplicación se ha expandido significativamente debido a su versatilidad y eficacia. Sin embargo, su principal desafío radica en la necesidad de un entrenamiento previo para identificar objetivos específicos, como personas o animales. Una vez entrenado adecuadamente, el sistema de visión por computador puede analizar y visualizar cualquier elemento detectable dentro de su campo de visión [1].

Para la detección de pulgones, la visión por computadora es una herramienta altamente eficaz, ya que, con suficientes muestras de entrenamiento, este sistema puede identificar cualquier objeto que se le indique buscar, ya sea de manera directa o indirecta, esto significa que el programa puede reconocer tanto al pulgón de forma explícita como a través de características indicativas de su presencia, como patrones en la planta o signos de daño, lo que lo convierte en un método óptimo para el monitoreo y control de estas plagas [2], además es importante señalar que el sistema de visión por computadora está vinculado al protocolo de transmisión en tiempo real (RTMP), que actúa como un puente para el flujo de información a analizar, este protocolo facilita la transmisión continua de datos entre el sistema de detección y la plataforma de análisis, asegurando que las imágenes y videos capturados se procesen en tiempo real, lo que mejora significativamente la eficiencia y la capacidad de respuesta del sistema ante la detección de pulgones u otros objetivos.

El sistema de transmisión en tiempo real es de los sistemas más usados hoy en día en varios ámbitos, principalmente en el ámbito del entretenimiento, pero también para otros sistemas, como lo es el de traslado de información o inclusive de audio y video, a pesar de todo esto, en donde se destaca más en la actualidad en las aplicaciones de recogimiento agrícola, esto pues se existen grandes aplicaciones de la visión por computador en lo agrícola, tanto para cuidado de las plantas como para automatizar los sistemas de cuidado de las plantas [3], [4].

Lo que busca el proyecto es lograr generar un sistema entrelazado entre los dispositivos, permitiendo de esta manera poder hacer que el proyecto logre su cometido de poder detectar, y mostrar si dentro de los cultivos existiera la presencia de pulgón, lo que si llegase a ser el caso los operarios puedan actuar de manera que puedan controlar la plaga antes de que esta se esparza en el resto de los cultivos de banano.

1.3) Descripción del Proyecto

El proyecto de detección de pulgón emplea diversos componentes para garantizar su correcto funcionamiento, entre los cuales destaca un UAV (Vehículo Aéreo No Tripulado) equipado con una cámara, este dron sobrevuela los cultivos capturando imágenes y videos, que son transmitidos a un dispositivo receptor a través del protocolo RTMP (Real Time Messaging Protocol). Este protocolo enlaza el dispositivo con el sistema de visión por computadora, donde se lleva a cabo el análisis de las imágenes y videos capturados, permitiendo así una detección precisa y en tiempo real de los pulgones en los cultivos.

Para llevar a cabo la recolección de material destinada al análisis por el sistema de visión por computadora, el primer paso es crear una base de datos que contenga muestras representativas de pulgones (Aphidoidea), estas muestras servirán como ejemplos de comparación, permitiendo al sistema de visión por computadora aprender y reconocer patrones específicos asociados a los pulgones en los cultivos de banano, esta base de datos es fundamental para entrenar al sistema, asegurando que pueda identificar con precisión la presencia de estos insectos en tiempo real, facilitando la detección temprana y el manejo efectivo de las plagas.

Una vez recolectadas las muestras necesarias, el siguiente paso es entrenar el modelo de reconocimiento dentro del sistema de visión por computadora, este entrenamiento se realiza asegurando que las muestras presenten variaciones entre sí, lo que permite al modelo aprender a identificar el pulgón en diversas condiciones, las diferencias pueden incluir variaciones en iluminación, ángulos de captura, tamaño de los pulgones, estados de desarrollo, y la apariencia del entorno, como diferentes tipos de hojas o daños causados en las plantas. Esto garantiza que el sistema pueda reconocer a los pulgones con precisión, incluso en situaciones que difieran de las imágenes de entrenamiento:

- Imágenes/videos donde la hoja del banano se encuentre con pulgón.
- Imágenes/videos donde la hoja de banano no tenga pulgón.
- Imágenes/videos donde se muestren algunos defectos provocados por el pulgón.

Para asegurar un correcto traslado de información entre los equipos del proyecto, es fundamental establecer un enlace eficiente mediante el protocolo RTMP (Real Time Messaging Protocol), que actúa como un puente entre el dispositivo transmisor y el sistema de visión por computadora. Este enlace es crucial, ya que permite la transmisión continua de fotos y videos capturados hacia el sistema de análisis. Sin esta comunicación, no sería posible procesar las imágenes ni detectar la presencia de pulgones (Aphidoidea), comprometiendo así la efectividad del proyecto en la identificación y monitoreo de la plaga en los cultivos.

1.4) Objetivos del proyecto

1.4.1) Objetivo General

Crear un sistema de detección de Aphidoidea (Pulgón) en plantaciones de banano mediante visión por computador y transmisión en tiempo real para el control efectivo por parte de los agricultores.

1.4.2) Objetivos Específicos

- Investigar las técnicas de visión por computador aplicadas en sistemas agrícolas y protocolos de transmisión en tiempo real mediante una revisión exhaustiva.
- Generar una base de datos de imágenes de Aphidoidea (pulgón) en distintos escenarios para el entrenamiento del modelo.
- Entrenar el algoritmo de visión por computador para la detección de Aphidoidea (pulgón).
- Verificar el correcto funcionamiento del sistema de visión por computador y el protocolo de transmisión, en distintos escenarios para comprobar la exactitud del modelo de detección y transmisión.

1.5) Justificación

La detección de plagas en la producción agrícola de banano mediante la visión por computadora, combinada con protocolos de transmisión en tiempo real, es esencial para reducir pérdidas en este sector vital para la economía del país, la implementación de estos sistemas integrales ofrece importantes beneficios para el control de plagas, particularmente del pulgón (Aphidoidea), proporcionando datos precisos y filtrados que facilitan la toma de decisiones para la prevención y manejo efectivo de las infestaciones, mejorando así la sostenibilidad y productividad del cultivo.

Desde una perspectiva macro, esta propuesta aborda el problema del pulgón mediante el uso de tecnologías emergentes respaldadas por la industria 4.0. Esto no solo facilita la detección específica de esta plaga, sino que también permite identificar otras amenazas o agentes externos que puedan afectar los cultivos de banano y, por ende, su producción, la integración de estas tecnologías avanzadas optimiza el monitoreo y control de las plagas, promoviendo una gestión agrícola más eficiente y resiliente frente a diversos desafíos.

Además, considerando que el protocolo de transmisión en tiempo real incrementa significativamente su relevancia, se abren nuevas líneas de investigación enfocadas en la interconexión de sistemas agrícolas, con el objetivo de optimizar la eficiencia en la toma de decisiones.

De esta manera, desde una perspectiva más amplia, el proyecto tiene un impacto directo tanto en la sostenibilidad como en la seguridad alimentaria, la detección temprana de la presencia de áfidos (pulgones) en los cultivos permite implementar medidas de control menos nocivas, lo que a su vez reduce el uso de pesticidas y fomenta un sistema agrícola más sostenible.

Los pulgones representan una grave amenaza para la producción de cultivos de banano debido a su método de alimentación, que consiste en succionar la savia de partes específicas de la planta, interrumpiendo el flujo de nutrientes, además, su rápida propagación exige una respuesta oportuna. Por esta razón, la implementación de un sistema de visión por computadora en los cultivos de banano permitiría un control más efectivo de esta plaga. A través de la transmisión en tiempo real, los agricultores podrían determinar con precisión cuándo y dónde es necesario aplicar pesticidas.

Con el sistema propuesto, la localización del pulgón se simplifica para los agricultores, permitiéndoles asignar recursos de manera más eficiente, de este modo, la rápida detección de colonias de pulgones facilita y agiliza la implementación de medidas contra plagas, lo que, a largo plazo y dependiendo de la escala, contribuirá a reducir el impacto ambiental.

En resumen, este proyecto no solo enfrenta un problema particular en la agricultura, sino que también fomenta prácticas más sostenibles y eficientes en la gestión de plagas en los cultivos de banano.

1.6) Alcance del Proyecto

El proyecto de detección de pulgones mediante visión por computadora y transmisión en tiempo real tiene como objetivo identificar la presencia de Aphidoidea (pulgones) en los cultivos de banano, esto busca promover una agricultura más sostenible, al reducir los costos asociados a la aplicación de pesticidas, lo que convierte esta práctica en un beneficio económico más que en una pérdida.

Para llevar a cabo el proyecto, es necesario comenzar describiendo el sistema UAV (Vehículo Aéreo No Tripulado), el cual estará equipado con una cámara de resolución mínima de 1920 x 1080 píxeles, todas las imágenes y videos capturados por la cámara serán transmitidos a un dispositivo receptor que, a través del protocolo RTMP (Real-Time Messaging Protocol), reenviará toda la información al sistema principal de análisis.

Como se mencionó anteriormente, el protocolo RTMP (Real-Time Messaging Protocol) o sistema de transmisión en tiempo real se encarga de transferir las imágenes y videos capturados por el UAV desde el dispositivo receptor hasta el sistema de análisis principal, es importante destacar que este protocolo actúa como un puente entre los dispositivos, facilitando la transmisión eficiente de la información entre ambos sistemas.

El sistema de análisis, comúnmente conocido como visión por computadora, es responsable de procesar las imágenes y videos enviados por el UAV (Vehículo Aéreo No Tripulado), para su correcto funcionamiento, este sistema debe ser previamente entrenado con muestras que incluyan la presencia y ausencia de áfidos (pulgones), una vez completado este entrenamiento, el sistema de visión por computadora será capaz de detectar si existe o no la presencia de dicha plaga.

En conclusión, el objetivo del proyecto es brindar apoyo al sector agrícola mediante el desarrollo de una solución tecnológica para la detección de Aphidoidea (pulgón), esta iniciativa promueve prácticas sostenibles en la agricultura, proporcionando a los operarios las herramientas necesarias para mejorar la calidad de los cultivos, la solución permite alertar sobre la presencia de Aphidoidea en los cultivos, lo que facilita su control o erradicación antes de que se convierta en un problema significativo, en resumen, se busca capacitar a los operarios para que utilicen herramientas tecnológicas en la gestión agrícola.

1.7) Metodología

Para el correcto desarrollo de la propuesta se deben tomar en cuenta varios métodos los cuales ayudaran a definir de forma precisa el proyecto, de manera que, al realizarse la recolección, e interpretación de los datos y la formulación de los resultados, estos tengan un mínimo de error, de forma que sean los más fiables posibles.

1.7.1) Metodología de investigación documental

Aquí se exploran los métodos tradicionales y actuales para la detección del pulgón, lo que implica un enfoque de recopilación y síntesis de información relevante [5], además de tomarse en cuenta los sistemas de visión por computador más aplicados dentro de sistemas de cuidado de cultivos a su vez qué se investigó todo en lo que respecta a diversas formas de tratamiento de cultivos.

1.7.2) Metodología de investigación aplicada

Aquí se refleja una metodología aplicada, en la cual se busca resolver un problema práctico a través del uso de tecnologías como YOLOv5, UAVs y protocolos de transmisión en tiempo real (RTMP) [6]. Este tipo de investigación es crucial puesto que revela diferentes formas en que la tecnología ya puede ser aplicada dentro de los sectores tanto agrícola cómo posiblemente también el sector ganadero en la tarde conocida como tecnología 4.0.

1.7.3) Metodología experimental

Se pueden identificar metodologías experimentales ya que dentro del proyecto se necesita realizar un programa el cual utilice el sistema yolov5 para poder obtener los resultados correspondientes en busca de lo requerido es decir el pulgón, en donde por obvias razones se hará uso de una metodología experimental hasta que se obtengan los resultados requeridos.

1.7.4) Metodología de desarrollo tecnológico

La metodología de desarrollo tecnológico aplicada en este proyecto abarca tanto la implementación de sistemas físicos como virtuales para la detección de pulgones, en el ámbito físico, se emplea un dron, mientras que en el entorno virtual se integran sistemas como YOLOv5, Python y OpenCV. Estos componentes juegan un papel clave en el procesamiento de imágenes, constituyendo los elementos principales a considerar para la identificación efectiva de plagas.

CAPÍTULO II

2.1) Marco Contextual

La ubicación elegida para la realización del proyecto de detección de pulgón mediante visión por computadora y un protocolo de transmisión en tiempo real es la finca Agroecológica de Jade, situada en la vía a la costa CERESITA, en este lugar, se tomarán muestras de fotografías y videos para investigar la presencia de pulgón en las plantas, así como para identificar aquellas que no presenten este plaga, el objetivo es generar una base de datos que sirva para el entrenamiento del sistema de visión por computadora.

El lugar fue seleccionado debido a la presencia actual de un problema significativo de pulgón, a pesar de los esfuerzos realizados para abordar esta situación de manera tradicional y sin el uso de pesticidas, se ha encontrado que la principal dificultad radica en la escasez de mano de obra para llevar a cabo una inspección exhaustiva de los cultivos de banano, esto significa que, aunque se haya realizado una limpieza en ciertas áreas de la plantación, la permanencia de algún rastro o incluso un solo huevecillo de pulgón puede dar lugar a un resurgimiento del problema, reiniciando así el ciclo de infestación.

2.2) Marco Conceptual

2.2.1) El pulgón

Las Aphidoidea, comúnmente conocidas como pulgones, son un tipo de insecto fitófago de pequeño tamaño, caracterizado principalmente por su capacidad para alimentarse de las plantas a través de un proceso denominado fagotopia, que consiste en la succión de la savia, esta alimentación es posible gracias a un órgano especializado conocido como estilete bucal, que permite al pulgón atravesar el tejido superficial de las plantas, un problema significativo que puede enfrentar el pulgón es la posibilidad de quedar atrapado si la savia del árbol se solidifica, sin embargo, al igual que los mosquitos, la saliva del pulgón contiene propiedades que previenen la coagulación de la savia, cabe destacar que, al alimentarse de la savia de las plantas, los pulgones generan una sustancia llamada mielada, que es una proteína azucarada y puede atraer a otros insectos [7], [8].

2.2.2) Protocolo de transmisión en tiempo real

El protocolo RTMP (Real-Time Messaging Protocol) es un sistema de transmisión en tiempo real que permite transmitir audio, video y datos entre un usuario y el servidor que almacena la información, este sistema fue desarrollado por Adobe Systems, aunque su uso activo por parte de los usuarios ha disminuido, RTMP sigue funcionando con normalidad, a pesar de ser un sistema de generación de servicio antiguo, se destaca por ser uno de los más eficientes y sencillos de implementar [9].

El protocolo RTMP tiene varias características las cuales son:

- **Transmisión en tiempo real**: Como se mencionó anteriormente, el protocolo RTMP está diseñado para transmisiones en tiempo real, como el streaming y cargas de datos más exigentes, como en el caso de los videojuegos en línea. Si el servicio no funcionara de esta manera, la transferencia de datos se vería comprometida, lo que, a su vez, ralentizaría la calidad del servicio [9].
- **Conexiones persistentes:** La conexión persistente se refiere al mantenimiento continuo del intercambio de información entre el servidor y el cliente, garantizando así que no se produzcan interrupciones en la transferencia de datos [10].
- **Soporte multimedia**: El sistema RTMP permite la transmisión de datos en formatos de video y audio, lo que lo hace adecuado para sistemas multimedia más complejos [10].
- **Flexibilidad en la codificación de datos**: Al admitir diversos tipos de códecs de audio y video, el sistema RTMP ofrece una notable flexibilidad en la transmisión de datos, lo que resulta beneficioso para su aplicación en una variedad de proyectos.
- **Seguridad**: A pesar de que el protocolo RTMP no cuenta con un sistema de cifrado, antes de ser descontinuado por Adobe, se introdujo un sistema similar llamado RTMPS, este protocolo añade la capa de seguridad que falta en RTMP mediante el cifrado de la información durante su transferencia.
- **Adaptabilidad de Bitrate (Dynamic Streaming)**: El protocolo de transmisión en tiempo real tiene la capacidad de adaptarse al ancho de banda de la red en uso, lo que permite que la experiencia de visualización sea más fluida al ajustarse a la velocidad de la red disponible.

- **Uso en la transmisión de videojuegos**: El sistema de transmisión en tiempo real se aplica en diversos sectores, siendo el streaming de videojuegos el más destacado, este enfoque es especialmente relevante en plataformas populares como YouTube y Twitch Gaming, que utilizan este tipo de transmisión para ofrecer contenido en vivo.

El protocolo RTMP fue uno de los sistemas más utilizados durante un tiempo. Sin embargo, ha ido quedando obsoleto debido al desarrollo de nuevos sistemas de transmisión en tiempo real, como el HTTP Live Streaming, aunque este último es más compatible con los sistemas web y ofrece una mayor flexibilidad, ambos protocolos mantienen similitudes en cuanto a configuraciones y funcionamiento.

2.2.3) Vehículo aéreo no tripulado

El UAV (Vehículo Aéreo No Tripulado) es un dispositivo utilizado en diversos campos, tanto militares como civiles, siendo este último el que presenta una mayor demanda, en lo que respecta a los sistemas UAV comerciales, estos dispositivos se caracterizan por tener una cámara integrada, cuya resolución puede variar según las necesidades de los usuarios, es importante tener en cuenta que, aunque los UAV pueden tener un tamaño estándar de entre 300 mm y 500 mm, existen drones pesados que se utilizan en la agricultura para la aplicación de pesticidas, sin embargo, estos últimos se emplean exclusivamente en el sector agrícola, mientras que los civiles suelen optar por drones comerciales, lo que los hace más accesibles [11].

2.3) Marco Teórico

2.3.1) Visión por computador

La visión por computadora es un campo de la inteligencia artificial y la informática que se enfoca en capacitar a las máquinas para interpretar y comprender la información visual del mundo real, este campo utiliza algoritmos y modelos matemáticos para analizar y extraer información relevante de imágenes o secuencias de imágenes, la visión por computadora se aplica ampliamente en diversas industrias y disciplinas, como la medicina, la fabricación, la automoción, la robótica, la seguridad, la realidad aumentada, entre otras [12].

A pesar de que el sistema de visión por computador tiene varias aplicaciones en diferentes campos sus principales puntos.

2.3.1.1) Principios básicos de la visión por computador

Para el correcto funcionamiento de la visión por computador, se debe de tomar en cuenta los siguientes puntos:

- **Sensores y cámaras**: La visión por computadora comienza con la captura de imágenes mediante cámaras y sensores, estos dispositivos transforman la información visual del mundo real en datos digitales que pueden ser procesados por las computadoras [13].
- **Filtrado**: Antes de llevar a cabo un análisis avanzado, las imágenes pueden ser preprocesadas mediante técnicas como el filtrado, las cuales permiten eliminar el ruido o mejorar las características relevantes [13].
- **Estandarización y escalamiento:** Es fundamental asegurarse de que las imágenes cuenten con la escala y el formato adecuados para su análisis, ya que, de lo contrario, podrían presentarse fallas en el entrenamiento del sistema de visión por computadora [13].
- **División de imágenes**: La división, también conocida como segmentación, implica la división de una imagen en áreas o segmentos significativos. Este proceso facilita el análisis de partes específicas de la imagen [13].
- **Identificación de características importantes**: La identificación y extracción de características relevantes de la imagen, tales como bordes, esquinas o texturas, es un proceso crucial, esto puede incluir el uso de operadores de convolución u otras técnicas de identificación que proporcionan un mayor nivel de detalle [14].
- **Identificación y ubicación**: La detección de objetos implica identificar la presencia de objetos específicos en una imagen y, en muchos casos, determinar su ubicación dentro de un área determinada.
- **Clasificación**: La clasificación de patrones en datos visuales permite determinar si una imagen contiene un objeto específico o si pertenece a una categoría particular que se busca identificar [14].
- **Red neuronal convolucional (CNN)**: El aprendizaje profundo, especialmente a través de las Redes Neuronales Convolucionales (CNN), ha revolucionado la visión por computadora al permitir que las máquinas aprendan automáticamente representaciones complejas a partir de datos visuales, sin embargo, el rendimiento de este enfoque depende principalmente de la cantidad de entrenamiento que se haya proporcionado [15].

- **Seguimiento de movimiento:** El rastreo del movimiento de un objeto a lo largo del tiempo se realiza mediante una serie de imágenes, lo que resulta útil en aplicaciones como la vigilancia y la robótica.
- **Salud, automoción, seguridad y más**: La aplicación de los principios de la visión por computadora en industrias específicas no solo hace que el sistema sea más atractivo, sino que también puede resultar en mayores oportunidades de rentabilidad, esto atrae la atención de diversos sectores interesados, quienes buscan integrar estas tecnologías en sus respectivas áreas.

La visión por computadora es un campo en constante evolución, lo que indica que sus principios pueden cambiar según sea necesario para llevar a cabo un análisis efectivo.

2.3.1.2) Algoritmos de procesamiento de imágenes y detección de objetos

Existen diversos sistemas que permiten el procesamiento de imágenes y la detección de objetos, los cuales pueden aplicarse en distintos campos que presentan diversas características, estas características son:

- **Histogram of Oriented Gradients (HOG)**

  El histograma de gradientes orientados (HOG) es un descriptor de características utilizado en visión por computadora y procesamiento de imágenes para la detección de objetos, este método es especialmente popular en el ámbito de la detección de peatones y otras tareas de reconocimiento de objetos, el algoritmo HOG tiene como objetivo capturar la estructura local y la apariencia de los objetos en una imagen mediante el análisis de la distribución direccional de los gradientes [16].

  Se debe de tomar en cuenta que el sistema extrae características que se basen en la orientación de gradientes.

- **Cascada de clasificadores en Adaboost**

  Esta técnica se utiliza en algoritmos de detección de objetos, siendo AdaBoost uno de los algoritmos comúnmente implementados con esta estrategia, la clasificación en cascada es un enfoque eficaz para detectar objetos con alta precisión y velocidad, el objetivo principal de este método es eliminar rápidamente las regiones de imagen que resultan poco prometedoras, lo que

reduce el número de evaluaciones exhaustivas de clasificadores más complejos [17].

- **YOLO (You Only Look Once)**

  El algoritmo YOLO (You Only Look Once) es un sistema moderno de detección de objetos en tiempo real y de código abierto, que emplea una única red neuronal convolucional para identificar objetos en imágenes o videos [18].

- **SSD (Single Shot Multibox Detector)**

  SSD (Single Shot MultiBox Detector) es una arquitectura eficiente para la detección de objetos que utiliza la predicción de múltiples cajas y escalas para abordar la variación en el tamaño y la forma de los objetos. Su diseño ofrece un equilibrio óptimo entre precisión y velocidad, lo que lo hace adecuado para diversas aplicaciones en tiempo real [19].

- **Region-based Convolutional Neural Network**

  La red neuronal convolucional de región (R-CNN) es un enfoque clásico para detectar objetos en imágenes presentado por Ross Girshick, Jeff Donahue, Trevor Darrell y Jitendra Malik en 2014. La arquitectura R -CNN se ha mejorado en versiones posteriores. incluye Fast R-CNN y Faster R-CNN, y sienta las bases para otros enfoques más modernos [20].

Estos son sólo algunos ejemplos de los distintos sistemas de visión por computador que existen y que son usados hoy en día. Además, el aprendizaje profundo, especialmente las redes neuronales convolucionales, ha jugado un papel importante en la mejora de la detección de objetos en imágenes.

Dentro del proyecto, se planea aplicar el sistema de reconocimiento de imágenes YOLO, el cual, como se ha mencionado, es uno de los sistemas más completos y ampliamente utilizados debido a su versatilidad y flexibilidad en el manejo de información relacionada con las imágenes.

2.3.1.3) Herramientas y tecnologías utilizadas en la detección de objetos en imágenes

La detección de objetos en imágenes implica una combinación de herramientas y tecnologías, que abarcan desde bibliotecas y marcos de aprendizaje profundo hasta plataformas de software y herramientas especializadas en procesamiento de imágenes. A continuación, se presentan algunas de las herramientas y tecnologías clave utilizadas en este campo.

- **TensorFlow**

  TensorFlow es una biblioteca de código abierto desarrollada por el equipo de investigación de Google Brain, diseñada para realizar cálculos numéricos, con un enfoque particular en operaciones de aprendizaje automático y aprendizaje profundo, esta biblioteca proporciona un conjunto integral de herramientas y recursos para crear y entrenar modelos de aprendizaje automático, abarcando desde modelos simples hasta arquitecturas complejas de aprendizaje profundo [21].

Ilustración 1 Sistema de aprendizaje profundo [21]

- **PyTorch**

  PyTorch es una biblioteca diseñada para aplicaciones de aprendizaje automático, reconocimiento de imágenes y aprendizaje profundo, desarrollada por el Laboratorio de Investigación de Inteligencia Artificial (FAIR) de Facebook. Está programada en Python, C y CUDA, y se presenta como un software gratuito, de alto nivel y de código abierto, que es compatible con los sistemas operativos Linux, macOS y Microsoft Windows [22].

Ilustración 2 Sistema de aprendizaje autónomo [22]

- **Keras**

  Keras es una interfaz de alto nivel diseñada para crear y entrenar modelos de aprendizaje profundo. Aunque Keras fue desarrollada inicialmente como una biblioteca independiente, a partir de TensorFlow 2.0 se integró directamente en TensorFlow, funcionando como una interfaz de alto nivel. Esta integración permite a los usuarios aprovechar todas las funciones y optimizaciones de

TensorFlow mientras trabajan con la simplicidad y accesibilidad que ofrece Keras [23].

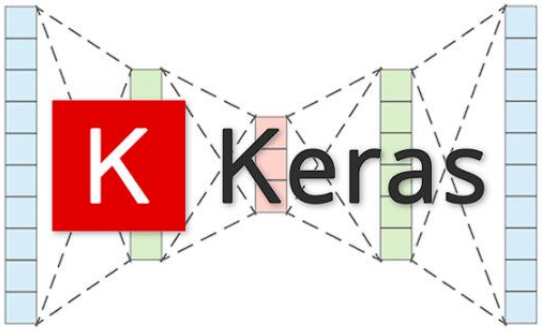


Ilustración 3 Sistema de aprendizaje profundo [23]

- **OpenCV**

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto dedicada al procesamiento de imágenes y la visión por computadora. Desarrollada inicialmente en lenguajes C y C++, OpenCV ofrece una amplia gama de funciones y algoritmos que abarcan áreas como la visión por computadora, el procesamiento de imágenes, el aprendizaje automático y la realidad aumentada [24].

Ilustración 4 Sistema de procesamiento de imágenes [24]

OpenCV es una de las herramientas más utilizadas en entornos de visión por computadora, gracias a sus diversas características. Además, su implementación en lenguajes de programación como C y C++ la convierte en una opción altamente funcional y versátil para los desarrolladores.

2.3.1.4) Bibliotecas de visión por computadora

OpenCV es una de las herramientas más utilizadas en entornos de visión por computadora, gracias a su amplia gama de características, además, su implementación en lenguajes de programación como C y C++ la convierte en una opción altamente funcional y versátil para los desarrolladores:

- **Scikit-Image**

  Scikit-Image es una biblioteca de procesamiento de imágenes que proporciona herramientas para la manipulación de imágenes, así como para la segmentación y otras operaciones que facilitan un procesamiento más eficiente, lo que a su vez mejora la identificación de características dentro de las imágenes [25].

- **Pillow (PIL Fork)**

  Esta biblioteca de procesamiento de imágenes en Python permite una manipulación más directa al admitir diferentes tipos de formatos de imagen, lo que facilita su manejo, además, ofrece la capacidad de guardar los cambios realizados, lo que la convierte en una opción óptima para la manipulación básica de imágenes [26].

- **SimpleCV**

  La biblioteca SimpleCV se basa en OpenCV con el objetivo de proporcionar un entorno más accesible para la creación de sistemas de visión por computadora, esta biblioteca está diseñada para ser utilizada por aquellos usuarios que están comenzando a explorar este campo [27].

- **Mahotas**

  Mahotas es una biblioteca de procesamiento de imágenes que permite una manipulación más detallada de las mismas y ofrece una amplia variedad de técnicas para el procesamiento de características. Esta combinación de capacidades facilita una mejor identificación de objetos en las imágenes [28].

- **OpenCV**

  OpenCV es una de las bibliotecas más utilizadas en Python por diversas razones, entre las más destacadas, ofrece una amplia gama de funciones para el procesamiento de imágenes, detección y seguimiento de objetos, calibración de cámaras, entre otras, estas características hacen de OpenCV la biblioteca más completa y versátil en el ámbito de la visión por computadora [29].

La biblioteca OpenCV se considera la opción, más óptima para la aplicación del proyecto, ya que cuenta con características que se ajustan perfectamente a los requisitos de este, tanto por su capacidad de análisis como por su eficiencia en el procesamiento de imágenes.

2.3.2) Protocolo RTMP

El protocolo RTMP (Real Time Messaging Protocol), también conocido como sistema de transmisión en tiempo real, es uno de los protocolos más utilizados en la actualidad en diversas plataformas de streaming, a pesar de ser un protocolo antiguo, en términos de funcionalidad sigue estando a la altura de muchos de los protocolos más recientes, para los fines del proyecto, RTMP continúa siendo una de las opciones más útiles para establecer un enlace entre sistemas, ya sea para el traslado de datos o para la transmisión de video en tiempo real, entre otras aplicaciones [30].

2.3.3) Agricultura y Plaga

En Ecuador, la importancia del banano trasciende los aspectos culturales y sociales, destacándose también en el ámbito económico, en términos sencillos, constituye uno de los principales pilares de la economía del país, ya que es el producto de exportación más notable de Ecuador, excluyendo el petróleo, además, el cultivo de banano representa una de las principales fuentes de empleo y proporciona un suministro alimentario estable, beneficiándose de un clima que favorece su crecimiento sin interrupciones [31].

Debido a que el banano es una de las plantaciones más extensas en Ecuador, también es uno de los cultivos más afectados por diversas plagas, entre ellas, el pulgón se destaca como una de las infestaciones más comunes en estas plantaciones, ya que este insecto busca fuentes de alimentación abundantes, lamentablemente para los agricultores, el banano se convierte en un objetivo propicio para el pulgón.

Tabla 1 Tipos de pulgón

	Pulgón de algodón	**Pulgón verde de melocotón**	**Pulgón de col**	**Pulgón negro de los cítricos**	**Pulgón verde de trigo**
Coloración	Variable	Verde	verde	Negro	Verde
Principal fuente de alimentación	Algodón Hortalizas	Variedad	Crucíferas	Cítricos	Cereales
Características	Puede afectar a diversas hortalizas	Puede afectar a diversas plantas	Afecta diversas plantas	Afecta diversas plantas cítricas	Puede afectar a diversos cereales, trigos y cebada

Fuente: [32]

A diferencia de otras plagas, cuyo tiempo de incubación y erradicación puede determinarse estacionalmente, el pulgón es una plaga que se presenta de manera constante debido a su alta capacidad reproductiva, lo que la convierte en un desafío difícil de abordar a lo largo del año, esta característica, junto con su método de alimentación, no solo puede afectar la calidad del producto final, sino que también facilita la transmisión de enfermedades entre las plantas, lo que puede perjudicar su calidad de vida.

2.3.4) Ciclo de vida del pulgón

El ciclo de vida de los pulgones es complejo e involucra la reproducción asexual durante el verano, en la cual las hembras aladas dan a luz larvas vivas en lugar de poner huevos, cuando las plantas se ven saturadas por la presencia de pulgones, algunas de las crías desarrollan alas y migran a nuevas plantas, a finales del verano, tanto machos como hembras producen huevos para la reproducción sexual, y las hembras depositan huevos durante todo el invierno, en climas cálidos, la fase de hibernación en forma de huevos puede no ser necesaria, lo que permite una sucesión continua, aunque los pulgones son controlados naturalmente por

depredadores como las mariquitas, en casos de infestaciones severas puede ser necesario recurrir al control químico [7].

Es importante considerar que, debido a que los pulgones producen mielada tras alimentarse, pueden atraer a las hormigas, que adoptan a estos insectos para obtener esta sustancia dulce, en muchos casos, las hormigas protegen a los pulgones con el fin de seguir accediendo a la mielada, lo que complica aún más su erradicación, las hormigas pueden llevarse a los pulgones a sus madrigueras, evitando así que los pesticidas los eliminen [7].

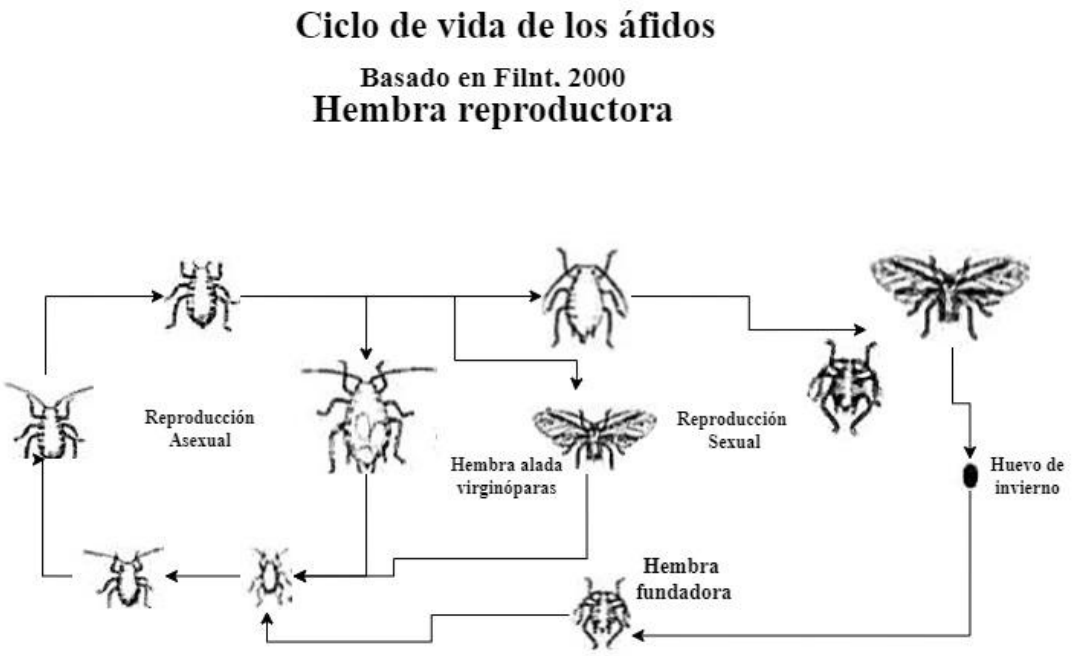


Ilustración 5 Ciclo de vida del pulgón [33]

Como se puede observar en la ilustración 5, el ciclo de vida del pulgón está estrechamente relacionado con la temporada en la que nace, para facilitar su comprensión, a continuación, se presenta una tabla que detalla este ciclo de vida.

Tabla 2 Características del ciclo de vida del pulgón

	Ciclo de vida en invierno	**Ciclo de vida en verano**
característica	Reproducción asexual	Reproducción sexual
Forma de nacimiento	Nacen siendo larvas	Nacen como huevos

Fuente: [33]

2.3.5) Métodos Tradicionales para la detección y control del pulgón

El método tradicional para la detección de pulgón, en diferentes cultivos, principalmente involucran la observación directa de las plantas e inspección visual de los insectos que se encuentren en los cultivos, de igual forma existen diferentes técnicas que pueden ayudar a la fácil detección de la plaga mencionada las cuales son:

- **Observación visual directa**

  Esta técnica consiste en la observación meticulosa de las plantaciones, lo que implica principalmente.

  - **Inspecciones regulares:** Los agricultores y técnicos agrícolas llevan a cabo inspecciones regulares de los cultivos en busca de signos que indiquen la presencia de pulgones [34].
  - **Inspecciones detalladas:** Se realizan revisiones exhaustivas de las hojas y las diferentes partes de la planta para detectar la presencia de pulgones [34].

- **Identificación de síntomas**

  Se busca cualquier indicio que pueda señalar un problema de salud en la planta, como puede ser:

  - **Deformidades de las plantas:** El pulgón, debido a su forma de alimentación, puede provocar deformidades en las plantas, tales como el enrollamiento de las hojas o el descoloramiento de estas [34].
  - **Maleza y Hongo:** La presencia de melaza provoca una secreción pegajosa en las plantas, mientras que la proliferación de pulgones fomenta el crecimiento de hongos negros que se desarrollan sobre esta sustancia [34].

- **Uso de lupa o microscopio**

  El uso de lupas y microscopios está orientado a la identificación del tipo de pulgón que afecta los cultivos, lo que permite determinar el pesticida más adecuado a aplicar, es fundamental considerar características como el tamaño, la forma y el color del pulgón, así como su comportamiento y el daño que causa a las plantas:

  - Antenas.
  - Patas.

- Forma del cuerpo.
- Coloración del pulgón.

En lo que respecta a los métodos tradicionales de eliminación, como se mencionó anteriormente, el uso masivo de pesticidas en los cultivos es la estrategia más común, sin embargo, existen otros métodos eficaces para el control del pulgón, especialmente cuando la infestación es aún incipiente o de menor escala, uno de estos métodos consiste en la utilización de trampas amarillas, que se colocan cerca de las plantas con el fin de atrapar a los pulgones, lo que ayuda a prevenir una mayor propagación de la plaga [34].

- **Monitoreo continuo**

El monitoreo continuo en la agricultura es una práctica esencial para evaluar y gestionar de manera efectiva la salud de los cultivos, en casos individuales de infestación por pulgones, es fundamental que un operador capacitado esté presente en todo momento para observar y evaluar los campos, este proceso implica un examen minucioso de las plantas en busca de signos que indiquen una posible infestación, tales como deformidades en las hojas, la presencia de melaza o daños visibles en las estructuras foliares [35].

Los operadores juegan un papel fundamental en la evaluación del alcance de la infección y son capaces de responder de manera rápida y precisa, la detección temprana de la presencia de pulgones es crucial para implementar estrategias de control adecuadas y prevenir daños significativos a los cultivos, además, el monitoreo continuo ofrece datos valiosos que permiten el análisis de tendencias a largo plazo, lo que contribuye a la toma de decisiones informadas en cuanto a la protección de las plantas [35].

- **Registro y Anotaciones**

El registro y anotaciones, por parte de agricultores, permite de forma menos actual, tener una noción del estado de los cultivos, ya sea por parte del rendimiento e inclusive en lo que respecta al bienestar de los cultivos.

2.3.6) Método actual para la detección de Pulgón

En la actualidad, se han desarrollado diversas técnicas que permiten el monitoreo de los cultivos para detectar la presencia de pulgones, con el objetivo de mantener

un nivel aceptable de control sobre estas plagas, para ello, se aplican las siguientes metodologías.

- **Sensores remotos e Imágenes Satelitales**

  Se implementan sensores que facilitan el monitoreo rápido de extensas áreas de cultivo, permitiendo así la detección temprana de plagas. Además, se utiliza tecnología satelital para retransmitir la información a los operadores responsables del área de cultivo correspondiente. Asimismo, se pueden emplear sensores que evalúan el estrés que experimenta la planta, lo cual puede ser un indicativo de la presencia de pulgones en los cultivos.

- **Drones Agrícolas**

  Se pueden emplear drones equipados con cámaras de mediana o alta resolución para la supervisión de los cultivos, estos drones también pueden estar dotados de sensores que permiten un monitoreo más efectivo de las áreas cultivadas, la operación de estos drones es supervisada de manera directa por un encargado, quien se encarga de vigilar el rendimiento de los cultivos [36].

- **Sistema de visión por computador**

  Se propone implementar un sistema de visión por computadora con el propósito de analizar imágenes y detectar automáticamente la presencia de pulgones, este sistema utilizará algoritmos de aprendizaje automático para identificar patrones asociados con las infestaciones de pulgones, además, se integrarán técnicas avanzadas de procesamiento de imágenes, tales como la segmentación semántica y la extracción de características, con el fin de mejorar tanto la precisión como la velocidad del sistema [37].

- **Sensores y dispositivos móviles**

  Se han desarrollado diversas técnicas para el cuidado de las plantas, entre las cuales se destacan los sensores diseñados para detectar irregularidades en los cultivos, estos sensores pueden ser utilizados por los agricultores para realizar una evaluación rápida del estado de sus cultivos, lo que permite identificar la posible presencia de pulgones, además, existen aplicaciones capaces de determinar el estado de las plantas en función de su apariencia y coloración.

- **Uso del ADN ambiental**

  A pesar de ser una técnica más compleja, esta metodología utiliza el medio ambiente como muestra para la detección de pulgones, no solo es necesario

tomar muestras del entorno, sino también de la tierra y el agua, todos estos elementos pueden ser aplicados para identificar el tipo de plaga presente, tras un análisis, estas muestras se comparan con una base de datos existente sobre pulgones para determinar su presencia en el área [38].

- **Trampas inteligentes**

  Para la implementación de trampas inteligentes, es necesario considerar varios aspectos que facilitarán tanto la detección como la erradicación de plagas, a continuación, se presentan los pasos fundamentales a seguir:

  - Utilización de sensores específicos.
  - Conectividad inalámbrica.
  - Monitoreo continuo.
  - Algoritmos de análisis de datos.
  - Notificaciones inmediatas por la detección.
  - Colocación de sensores en ubicaciones estratégicas.
  - Integración con sistemas de gestión agrícola.
  - Optimización del uso de pesticidas.
  - Almacenamiento de datos a largo plazo.

- **Técnicas de diagnóstico molecular**

  De manera similar al punto anterior, se pueden emplear técnicas de identificación basadas en la detección de material genético (ADN o ARN) de diversas muestras recolectadas, este enfoque permite realizar un análisis preciso para identificar la presencia de pulgones y otros patógenos en los cultivos:

  - **Muestras:**

    Es necesario recoger muestras para el análisis, las cuales pueden incluir tierra, agua, muestras de la planta infectada o incluso el mismo pulgón, este procedimiento es fundamental para llevar a cabo una evaluación precisa de la presencia de plagas y determinar las medidas de control adecuadas [39].

  - **Extracción de ácidos nucleicos**

    En esta etapa, se procede a la extracción del material genético de las muestras recolectadas, asegurando que se mantenga la integridad de

estas para usos posteriores, este paso es crucial para garantizar la precisión en el análisis y la identificación de la plaga.

- **PCR (Reacción en cadena de la Polimerasa)**
  Utiliza tecnología de PCR para amplificar regiones específicas de ADN o ARN de pulgón. Los cebadores (secuencias cortas de ADN) están diseñados para ser complementarios de regiones específicas del material genético de los pulgones [39].
- **Electroforesis**
  La electroforesis es una técnica que se puede emplear para separar y visualizar las cadenas de ADN que han sido amplificadas durante el proceso de PCR (reacción en cadena de la polimerasa).
- **Hibridación Molecular**
  En técnicas avanzadas como la hibridación de ácidos nucleicos, se pueden emplear sondas específicas para pulgones, las cuales se unen selectivamente a secuencias complementarias de ADN o ARN de estos insectos [39].
- **Secuenciación del ADN**
  Si se necesita una identificación más precisa o la confirmación de la especie, se puede emplear la secuenciación de próxima generación (NGS) para obtener la secuencia completa del ADN [39].

- **Análisis Biométrico**
  En este último punto, se implementa el uso de software y herramientas biométricas que facilitan la identificación del tipo de pulgón presente en los cultivos, estas herramientas no solo permiten detectar la especie específica, sino que también proporcionan información adicional que puede ser relevante en función de la situación, como la resistencia a pesticidas o las condiciones ambientales que favorecen su proliferación [40].

La combinación de todos estos métodos permite adoptar un enfoque más integral para la detección temprana del pulgón, así como para el monitoreo y manejo efectivo de las infestaciones, este enfoque holístico no solo optimiza la identificación de la plaga, sino que también contribuye a la implementación de

estrategias de control más precisas y eficientes, garantizando la salud y productividad de los cultivos [41].

2.3.7) Generación del enlace con el protocolo RTMP

Para establecer un servicio de transmisión en tiempo real, el primer paso consiste en preparar el dispositivo que alojará el servidor, esto implica descargar e instalar NGINX, un software que permite la creación de servidores tanto internos como externos. NGINX es una herramienta poderosa que facilita la gestión de flujos de datos, permitiendo la transmisión eficiente de contenido en tiempo real.

Una vez descargado, el servicio de NGINX puede ser ejecutado de forma directa, lo que permitiría la transferencia de información y habilitaría el servidor RTMP, sin embargo, para configurarlo adecuadamente, es necesario seguir algunos pasos adicionales, esto incluye modificar el dominio del servidor, lo que implica cambiar el tipo de direccionamiento para que el dispositivo receptor pueda retransmitir la información al sistema de visión por computadora de manera efectiva.

En forma resumida lo que se debe de hacer para realizar la interconexión entre dispositivos es:

- Crear un punto de acceso en el dispositivo receptor y conectar la computadora a la misma red.
- Instalar NGIXN en la computadora para poder generar el servidor de tipo RTMP.
- Trasmitir las imágenes del dispositivo receptor a la computadora usando la aplicación PRISN Live.
- Ejecutar un script de Python para el monitoreo del puerto 1935.

De forma resumida, estos serían los pasos para poder generar una correcta, interconexión entre los dispositivos, los cuales permitirán el traslado de imágenes/videos entre el UAV y el sistema de visión por computador.

2.3.8) Integración de la visión por computadora y el protocolo RTMP

La visión por computadora es un sistema versátil que se aplica en diversos campos, abarcando desde tareas simples como el reconocimiento facial hasta desafíos más complejos, como la detección de objetos en movimiento y la identificación de acciones, esta capacidad se logra gracias a un entrenamiento

adecuado; a mayor cantidad de ejemplos que el sistema de visión por computadora pueda procesar, mejor será su habilidad para diferenciar y cumplir con las tareas solicitadas.

El protocolo RTMP, como se ha descrito anteriormente en este documento, es un protocolo de transmisión en tiempo real que continúa en uso en plataformas como YouTube y Twitch Gaming, esta perdurabilidad se debe a que, a pesar de la aparición de nuevos protocolos para la transmisión en tiempo real, todos cumplen la misma función fundamental: la transmisión de audio y video de manera eficiente.

Para integrar ambos sistemas, tanto el de análisis como el de transmisión, es fundamental considerar el propósito de su implementación. es decir, es necesario definir para qué se implementará cada sistema, ya que, dependiendo de la respuesta, podría ser necesario realizar ajustes en el sistema de visión por computadora o incluso modificar el sistema de transmisión RTMP, esto asegurará que ambos sistemas se adapten de manera efectiva al trabajo que se les asigne.

En cuanto a la generación del enlace, el proceso es sencillo. Una vez que se ha definido el propósito del sistema, el primer paso es habilitar el servicio de transmisión mediante la ejecución de NGINX, lo que iniciará la transmisión., posteriormente, se debe abrir el sistema de transmisión, que puede ser OBS si se está utilizando un ordenador, o PRINS Live si se trata de un dispositivo transmisor, con estos pasos, la transmisión estará operativa para el traslado de información

2.3.9) La visión por computadora y sus aplicaciones en la agricultura

La visión por computadora presenta diversas aplicaciones en múltiples campos, pero el enfoque de este proyecto se centra en el ámbito agrícola, en este contexto, su principal utilidad radica en el cuidado de las plantas, lo cual implica, de manera simple, la verificación de su estado a través de aspectos como la coloración, la forma y el análisis del suelo, es evidente que cualquier variación o afectación que experimenten los cultivos durante su crecimiento puede resultar en diferentes consecuencias al momento de la cosecha, esto podría llevar a la producción de frutos defectuosos o, incluso, a la inadecuación de los mismos para su comercialización [42].

Como tal la visión por computadora aplicada en la agricultura, se usa para la supervisión de cultivos, pero de una manera más amplia sus aplicaciones son:

- **Mejoramiento de la salud de los cultivos**: Las cámaras utilizadas, en conjunto con el sistema de visión por computadora, son capaces de reconocer y detectar signos de estrés o enfermedades en las plantas, gracias a esta capacidad, el sistema puede alertar a un operario para que verifique el problema, facilitando así su posterior resolución [42].
- **Estimación de rendimiento de los cultivos**: La visión por computadora permite estimar el rendimiento de los cultivos mediante la recolección de datos sobre el crecimiento de las plantas, así como el tamaño de sus frutos [42].
- **Robótica y Automatización**: Los robots utilizados en la cosecha y siembra de diversos cultivos emplean sistemas de visión por computadora para identificar las áreas destinadas a la siembra o cosecha, además, esta tecnología se puede aplicar para el reconocimiento y eliminación de maleza [42].
- **Drones**: Los drones equipados con cámaras pueden integrar sistemas de visión por computadora para, al sobrevolar los cultivos, recopilar datos relacionados con el bienestar de las plantas, la humedad del suelo, entre otros aspectos [42].

La visión por computadora es una de las tecnologías más aplicadas en la actualidad y, a su vez, es altamente combinable con otras tecnologías, como el IoT (Internet de las Cosas), esta integración contribuye a que la agricultura evolucione hacia una etapa conocida como Agricultura 4.0 [42].

2.3.10) Aplicación de la visión por computadora para la detección de plagas

El uso de la visión por computadora para la detección de plagas representa una solución innovadora y eficaz para el monitoreo y manejo de infestaciones, esta tecnología emplea algoritmos avanzados y sistemas de procesamiento de imágenes para identificar automáticamente las irregularidades causadas por las plagas, ofreciendo múltiples ventajas en comparación con los métodos tradicionales de detección, a continuación, se presentan aspectos clave del uso de la visión por computadora en este ámbito [43].

- **Captura de imágenes**: La captura de las imágenes se aplica para poder recoger muestras las cuales será entregadas al sistema como ejemplo.
- **Procesamiento de imágenes**: Las muestras son procesadas, para mejorar la resolución y de esta forma ser viables para su utilización.
- **Segmentación de imágenes**: Se aplican las técnicas de segmentación de imágenes siempre y cuando las mismas no posean la calidad suficiente para ser usadas dentro del sistema de visión por computador.
- **Extracción de características**: En imágenes que indiquen la presencia de plagas, extraiga características clave como color, forma y textura de las áreas identificadas.
- **Entrenamiento del Modelo**: Como se indica, esta parte del proyecto se realiza para poder entrenar el sistema de visión por computadora para obtener mejores resultados.
- **Implementación del Modelo**: El modelo ya finalizado es implementado dentro de un sistema el cual pueda procesar imágenes.
- **Alerta y Decisiones Automatizadas**: En el caso de qué se encuentre la presencia de plaga el sistema puede lanzar una alerta, a los agricultores, también puede habilitarse las opciones para tomar decisiones autónomas para el control de plagas.

CAPITULO III

3.1) Elementos de la propuesta

3.1.1) Elementos físicos de la propuesta

Los elementos físicos son responsables de la recolección de datos para su posterior procesamiento, estos equipos incluyen el vehículo aéreo no tripulado (VANT), el dispositivo receptor y, finalmente, el equipo donde se llevará a cabo el análisis de los datos recolectados, para este análisis, es fundamental considerar las características más adecuadas que se adapten mejor al proyecto, las cuales se detallarán a continuación.

3.1.1.1) Vehículo Aéreo no Tripulado

Los UAV (Vehículos Aéreos No Tripulados) están disponibles en diversas versiones, diseñadas tanto para uso comercial como profesional en distintos campos [44] , estos drones presentan características destacadas que pueden variar según el campo de aplicación, como el militar o el competitivo, aunque ambos tipos son útiles para el proyecto, el dron de tipo comercial se considera el más adaptable debido a su fácil acceso y menor costo, por otro lado, los drones profesionales, especialmente los diseñados para carreras, exhiben un enfoque particular: cuentan con hélices que proporcionan mayor potencia, aunque su duración de vuelo es reducida [45], por otro lado, en el ámbito militar, los drones suelen tener un tamaño mayor, lo que implica una mayor capacidad de carga de batería, sin embargo, este aumento de peso conlleva una reducción en el tiempo de vuelo y también afecta la maniobrabilidad del dispositivo [46].

En contraste, los drones de uso comercial, también conocidos como drones de uso público, ofrecen una mayor accesibilidad en términos económicos, sus características pueden variar considerablemente, aunque la calidad de las cámaras que utilizan depende en gran medida de la inversión realizada en el dron; a mayor costo, generalmente se obtiene un mejor rendimiento, además, es importante tener en cuenta que, al ser drones diseñados para uso personal, su duración de batería suele oscilar entre 20 y 30 minutos [47].

Ilustración 6 Dron de uso comercial Potensic ATOM SE

De forma complementaria, a continuación, se presentará una tabla que detalla algunas características generales de los drones de uso comercial, al finalizar la tabla, se ofrecerá una breve explicación sobre por qué la utilización de estos drones en el proyecto representa un beneficio para futuras aplicaciones en otros sectores, especialmente en el ámbito agrícola, que es el enfoque principal de este estudio.

Tabla 3 Tabla de detalles del dron

Características del dron	
Cuatro Motores del tipo Brushless	7.4 v – 15 v
Cuatro Hélices de Dron	5 pulgadas de diámetro 4.5 pasos
Batería	100mAh a 500mAh
Cámara	3.7 (1s) v a 22.2 v
Placa FC	3.3 v a 5v
Sensores de estabilidad	3.3v a 5v
Control de velocidad (ECS)	3.7 v a 22.2 v

Fuente: [48]

En lo que respecta al funcionamiento de los drones de uso comercial, estos pueden ser utilizados en proyectos relacionados con vigilancia o análisis de datos, gracias a las diversas posibilidades que ofrecen sus distintos modelos, las principales diferencias entre ellos radican en las características de la cámara, que varían según el modelo, así como en el tiempo de duración del vuelo, este último puede experimentar variaciones debido a factores ambientales, como el clima, las condiciones meteorológicas e incluso la resolución de la cámara, en su máxima calidad, el tiempo de vuelo se sitúa entre 15 y 20

minutos, mientras que en configuraciones de resolución más baja, este puede superar el rango de tiempo propuesto [49].

Para facilitar la comprensión de cómo la duración del tiempo de vuelo está directamente relacionada con el tipo de resolución de la cámara utilizada en el dron, la Tabla 4 presentará un esquema que muestra las distintas resoluciones junto con su respectivo tiempo de vuelo, es importante destacar que estos tiempos pueden variar según la marca del dron empleado para las labores de análisis realizadas por los operadores.

Tabla 4 Relación Vuelo / Resolución de la Cámara

Relación tiempo de vuelo/resolución		
Snaptain S5C	720p	10 min
Altair AA108	720p	10 min
HStone	1080p	15 min
Potensic ATOM SE	4K	31 min

Fuente [50]

Como se observa en la Tabla 4, tanto el tiempo de vuelo como la resolución de la cámara pueden variar según el modelo del dron utilizado, entre las marcas más destacadas en términos de calidad y duración de vuelo, se encuentra el DJI Mavic Air, que ofrece una resolución de hasta 4K y un tiempo de vuelo de hasta 20 minutos en alta resolución, asimismo, también se puede considerar el Potensic ATOM SE, que presenta características competitivas en este ámbito [50].

3.1.2) Componentes lógicos

3.1.2.1) Python

Python es un lenguaje de programación considerado de alto nivel debido a su fácil adaptabilidad a diversas tareas, que pueden variar ampliamente, su sintaxis clara y legible facilita la comprensión por parte del usuario del tipo de procesos que se van a realizar en el código. Este lenguaje cuenta con una amplia variedad de aplicaciones en entornos virtuales, y su uso está limitado únicamente por factores externos, como el equipo en el que se ejecute el programa [51].

Ilustración 7 Entorno programático Python [51]

Es importante tener en cuenta que el programa Python opera en un formato equivalente al del CMD (símbolo del sistema), lo que significa que utiliza y ejecuta una interfaz similar a esta, en este entorno, los comandos se ingresan de manera tradicional y constituyen la fuente principal de interacción, sin embargo, esto puede generar un cierto grado de dificultad, ya que al ejecutarse en un entorno CMD, los comandos no proporcionan mensajes de error claros en cuanto a la sintaxis o la concordancia de los mismos, por lo tanto, se sugiere utilizar un entorno alternativo que facilite la interacción con Python, proporcionando una interfaz más amigable y accesible para el usuario.

3.1.2.2) PyCharm

PyCharm es un programa complementario de Python que ofrece un entorno más sencillo y amigable para los usuarios, permitiendo una programación más accesible en comparación con el programa base, es importante destacar que PyCharm actúa como un complemento y no como una versión completa de Python; por lo tanto, no puede funcionar de manera independiente sin la instalación previa de Python [52].

Ilustración 8 Programa Pycharm [53]

La instalación de PyCharm, como complemento de Python, es opcional y depende de las preferencias de cada usuario, sin embargo, se recomienda su instalación, ya que proporciona un entorno más comprensible y accesible, lo que facilita la programación del proyecto.

3.1.2.3) PRISM Live

PRISM Live es una aplicación diseñada para la transmisión de datos provenientes del dispositivo receptor, a través de la cual se enviará toda la información recopilada al sistema de análisis principal, el uso de este programa está principalmente relacionado con su compatibilidad con el protocolo de transmisión en tiempo real, además de ofrecer un entorno sencillo pero completo, con opciones de personalización en lo que respecta a la transmisión.

Ilustración 9 PRISM Live Studio [54]

Cuando se habla de opciones de personalización, nos referimos a la capacidad de agregar las direcciones del protocolo RTMP, a través del cual se transmitirán los datos. Esta funcionalidad facilita la creación de un puente que permite la conexión entre los equipos mencionados, garantizando así el traslado efectivo de la información.

3.1.2.4) OpenCV

El software OpenCV (Open Source Computer Vision Library) es una biblioteca de visión artificial que admite varios lenguajes de programación, incluyendo Python, C++, Java y otros, esta versatilidad permite a los desarrolladores integrar capacidades avanzadas de procesamiento de imágenes y análisis de video en diversas aplicaciones:

- Python.
- Java
- Matlab
- Octave
- JavaScript

Además, el software OpenCV ofrece una amplia compatibilidad con diversos entornos de programación, lo que lo hace altamente funcional en diferentes sistemas operativos, tales como Windows, macOS, GNU/Linux, Android e iOS. La única limitación del programa radica en su ejecución; es decir, el equipo en el que se utilice debe contar con los recursos necesarios para garantizar un funcionamiento fluido y eficiente del software.

Tabla 5 Detalles de uso de Open CV

Espacio de disco duro	Memoria RAM	Procesador	Tarjeta de video
100 MB	4 Gb	Core i5	Amd radeon
500 MB	8 Gb	Core i7	GTX 1050

Fuente: [29]

3.2) Diseño de la propuesta

En lo que respecta al diseño del proyecto, este depende principalmente de la planificación de su implementación. Dado que el área geográfica y el tipo de clima del sitio pueden influir significativamente, es posible que el proyecto requiera aditamentos adicionales para mitigar el impacto de dichos factores. Tras una revisión exhaustiva del sitio seleccionado, se han llegado a las siguientes conclusiones.

- El factor geográfico no tiene una relevancia significativa dentro del proyecto, dado que, al utilizar un UAV (Vehículo Aéreo No Tripulado), este no requiere de una geografía estable para llevar a cabo las grabaciones de video del sitio.
- El factor aéreo adquiere mayor relevancia, ya que el UAV (Vehículo Aéreo No Tripulado) puede verse afectado por ventiscas repentinas que podrían desestabilizarlo, complicando así algunas tomas, sin embargo, según las observaciones del sitio, las corrientes de aire que superan los 13 km/h no son frecuentes, lo que permite estimar un vuelo seguro en esas condiciones.
- Otro de los factores que puede influir en el vuelo del UAV (Vehículo Aéreo No Tripulado) es la abundante flora del área, especialmente la de los bananos, que puede representar un obstáculo, esto se convierte en un riesgo si el operador del dispositivo no tiene el cuidado adecuado al momento de recolectar los datos.

Tomando en cuenta los puntos mencionados anteriormente, el diseño que mejor se adapta a los requisitos del tema de detección de pulgones no es necesariamente complejo; sin embargo, sí considera diversos aspectos que deben evitarse para garantizar que la captura de imágenes y videos sea lo más eficiente posible, minimizando así la posibilidad de errores para el programa encargado de identificar al pulgón.

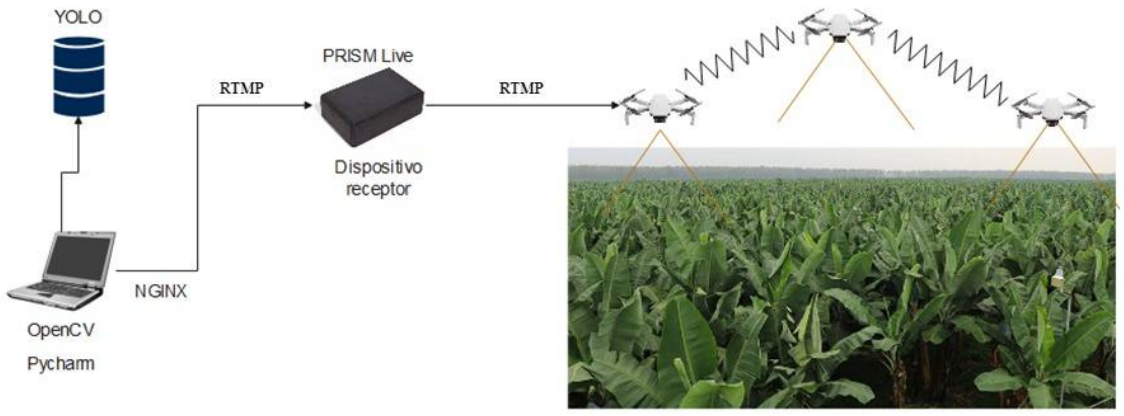


Ilustración 10 Diseño para el tema de detección de pulgón

En la ilustración 10 se pueden observar los diversos componentes que conforman el tema de la tesis sobre la detección de pulgones, estos incluyen la inteligencia artificial (IA) encargada de registrar y detectar la presencia de pulgones, la computadora que genera y procesa toda la información recolectada, y el dispositivo receptor que alberga PRISM Live, responsable del redireccionamiento de la señal junto con el protocolo RTMP. Por último, se encuentra el dron Potensic ATOM SE, que se encarga de recolectar las muestras del entorno.

Como es de suponer, cada componente mostrado en la ilustración es vital para el correcto funcionamiento del proyecto, sin alguno de estos elementos, no sería posible establecer la comunicación entre los dispositivos ni el procesamiento de las imágenes, asimismo, sin el protocolo base de transmisión RTMP, no se podría llevar a cabo la comunicación en tiempo real, lo que afectaría la identificación de los pulgones.

3.2.1) Conexiones

Las conexiones utilizadas en el proyecto son del tipo no guiadas, en primer lugar, es importante destacar que estas conexiones se representan de manera similar a cómo se muestran en la ilustración 11, aunque esta última presenta una versión simplificada, para ofrecer una visión más detallada de las conexiones finales del proyecto, se presenta a continuación la siguiente ilustración.

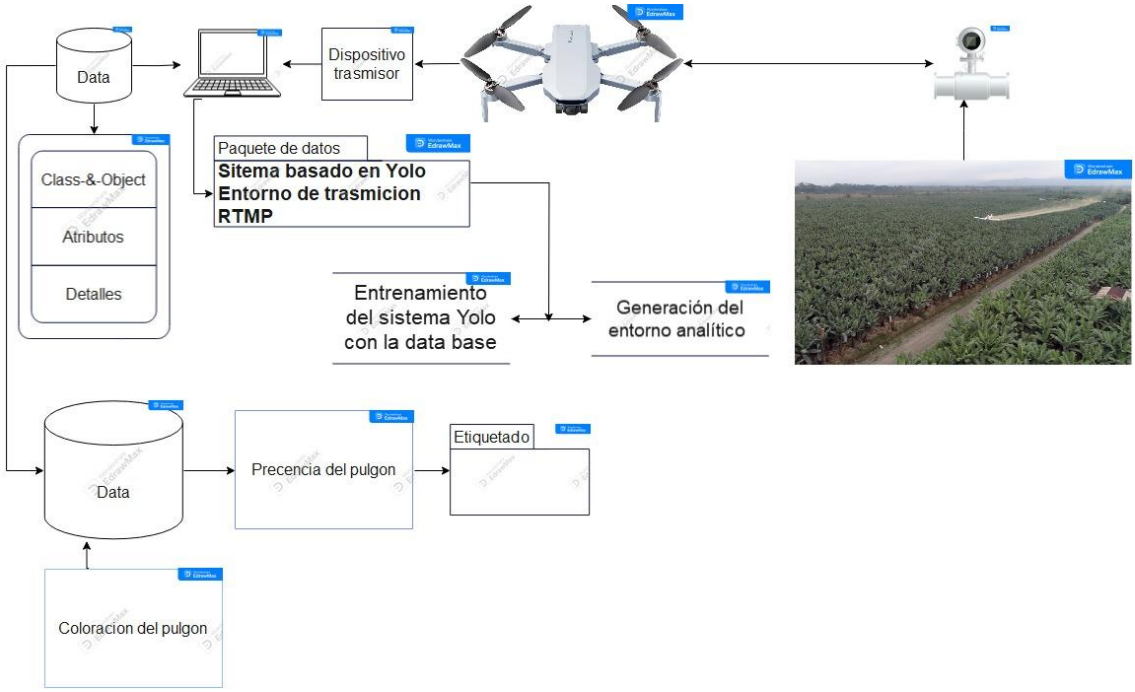


Ilustración 11 Conexiones completas del proyecto de Detección de pulgón en plantíos de banano

Como se puede observar en la ilustración 11, se destacan todas las conexiones y recursos que se utilizan, así como algunos de los puntos más importantes, como las carpetas y bases de datos necesarias. Esta información nos lleva al siguiente punto, que se refiere a los protocolos requeridos para el correcto funcionamiento del procesamiento de las imágenes.

3.2.2) Librerías

Las bibliotecas de información permiten que el programa interactúe de manera más eficiente con diversos componentes, por ejemplo, **sys** se utiliza para gestionar operaciones relacionadas con el sistema y el entorno de ejecución del programa, otra biblioteca clave es **Path**, que proporciona una manera conveniente de trabajar con rutas de archivos y directorios en el sistema de archivos, también está **cv2**, conocida como OpenCV, que se encarga de la captura y procesamiento de información visual, finalmente, la biblioteca **torch** permite al programa acceder a un entorno de aprendizaje profundo.

A continuación, se presenta un análisis más detallado del funcionamiento de cada una de estas bibliotecas.

- **OpenCV (Open Source Computer Vision Library)**: **cv2**, también conocida como OpenCV, es una biblioteca de visión por computadora ampliamente utilizada, proporciona herramientas robustas para la manipulación de imágenes y videos, que incluyen la lectura, escritura y procesamiento de estos medios, además, facilita la detección y reconocimiento de objetos, lo que la convierte en un recurso esencial para proyectos que involucren análisis visual, como la identificación de plagas en cultivos o cualquier tarea de visión artificial avanzada [55].
- **Nympy:** Es una biblioteca fundamental para la computación científica en Python. Proporciona soporte para arreglos multidimensionales y matrices, junto con una amplia colección de funciones matemáticas que permiten realizar operaciones eficientes sobre estos datos. Es particularmente útil en el procesamiento de imágenes, cálculos numéricos avanzados y otras tareas que requieren manipulación de grandes volúmenes de datos en múltiples dimensiones [56].
- **PyTorch:** Es una biblioteca de aprendizaje profundo que facilita la creación y entrenamiento de redes neuronales, proporciona soporte para operaciones tensoriales, autograd (diferenciación automática), y modelos pre entrenados para tareas de visión por computadora y procesamiento del lenguaje natural [22].

- **PIL((Python Imaging Library) y su fork Pillow**: Son bibliotecas diseñadas para la apertura, manipulación y almacenamiento de imágenes en diversos formatos. Permiten convertir imágenes entre diferentes formatos y aplicar transformaciones, como cambios en el tamaño, recortes, rotaciones, ajustes de color, entre otros, facilitando el procesamiento y la edición de imágenes para distintos fines [57].
- **OS (Operating System)**: Es una biblioteca estándar que proporciona una forma de interactuar con el sistema operativo, permite manipular archivos y directorios, acceder a variables de entorno, y realizar otras operaciones relacionadas con el sistema [58].
- **Pathlib**: Es una biblioteca para la manipulación de rutas de archivos de manera orientada a objetos, facilita la creación, modificación y análisis de rutas de archivos y directorios de manera más intuitiva que usando las funciones tradicionales del módulo **'os'** [59].
- **Path:** Es una clase de la biblioteca **'pathlib'** que proporciona una interfaz orientada a objetos para manejar rutas de archivos y directorios, permite realizar operaciones comunes con rutas de manera más legible y estructurada [59].

Estas bibliotecas constituyen una parte esencial del programa, ya que desempeñan funciones cruciales para el procesamiento eficiente de la información recolectada durante el recorrido del dron, es importante destacar que, aunque las bibliotecas actúan como facilitadoras, su correcta implementación y la adecuada aplicación del código han sido fundamentales para garantizar el funcionamiento óptimo del sistema, asegurando así la operatividad y éxito del proyecto.

3.2.3) Datos de Identificación

Para que el programa pueda identificar correctamente lo que se busca, no solo es necesario contar con las bibliotecas adecuadas, sino también disponer de una base de datos que contenga muestras, estas muestras guían al programa sobre qué debe buscar y cómo debe hacerlo, para el proyecto actual, se ha implementado el sistema de visión por computadora YOLO (You Only Look Once), que permite realizar entrenamientos avanzados para la identificación de objetos o personas mediante algoritmos de procesamiento visual.

Aunque el sistema YOLO es ampliamente utilizado en diversos campos, presenta ciertas limitaciones que pueden variar según diferentes factores. Una de las principales deficiencias es la falta de suficientes muestras, lo que impide que YOLO identifique con precisión lo que se busca, por esta razón, al realizar un muestreo, se recomienda contar

con una base de al menos trescientas muestras, estas deben organizarse en dos carpetas principales, cada una conteniendo dos carpetas adicionales, las cuales a su vez tendrán otras dos carpetas en su interior, formando una estructura jerárquica como la siguiente:

- **imagies**: En esta carpeta se almacenan dos carpetas en las cuales, se almacenan como su nombre lo indica las imágenes, que se usan como "train" y "val", que son las imágenes de identificación y validación, que permiten al programa identificar lo que se desea buscar.
- **Labels**: En la carpeta denominada "labels" se almacenan los archivos de texto que contienen las coordenadas específicas que indican la ubicación del pulgón, obtenidas a través del proceso de etiquetado, esta carpeta está organizada en dos subcarpetas: "train" y "val", la subcarpeta "train" contiene las muestras de entrenamiento utilizadas para enseñar al modelo a identificar el objeto, mientras que la subcarpeta "val" almacena las muestras de validación, que permiten al programa realizar comparaciones y ajustar su capacidad para localizar el objeto de interés con mayor precisión.

3.2.4) Etiquetado

El etiquetado de las muestras es una fase crucial del proceso, ya que permite al programa identificar con precisión los elementos que debe buscar, siempre y cuando se cuente con un número adecuado de muestras, sin una cantidad suficiente de datos de referencia, el programa no podrá realizar una detección eficiente y precisa, para garantizar un rendimiento óptimo del sistema de detección, se recomienda contar con al menos trescientas muestras, aunque esta cantidad puede variar según el caso específico [60].

Es importante tener en cuenta que el número de muestras recomendado es un mínimo generalmente aceptado por los profesionales que trabajan con sistemas de visión por computadora, esto se debe a que, además de la cantidad de muestras utilizadas en el entorno YOLO, también influye el número de repeticiones del entrenamiento, el programa permite ajustar o modificar el muestreo de cada una de las muestras proporcionadas, lo que puede mejorar significativamente los resultados obtenidos [60].

Una vez determinado el número mínimo de muestras necesarias para que el programa funcione de manera óptima, a continuación, se presenta una selección de los programas más recomendados para realizar el etiquetado de dichas muestras, estos programas han sido elegidos por su eficacia y facilidad de uso, y se destacan en la comunidad por su

capacidad para gestionar grandes volúmenes de datos de manera eficiente, en particular, el último programa de la lista se considera el más adecuado debido a sus funciones avanzadas y características, que facilitan considerablemente el proceso de etiquetado, haciendo que esta tarea sea más rápida y precisa [60].

- **LabelImg:** Herramienta de código abierto para el etiquetado de imágenes, que permite generar archivos de anotación en formato XML (compatible con Pascal VOC) o TXT (compatible con YOLO) [61].

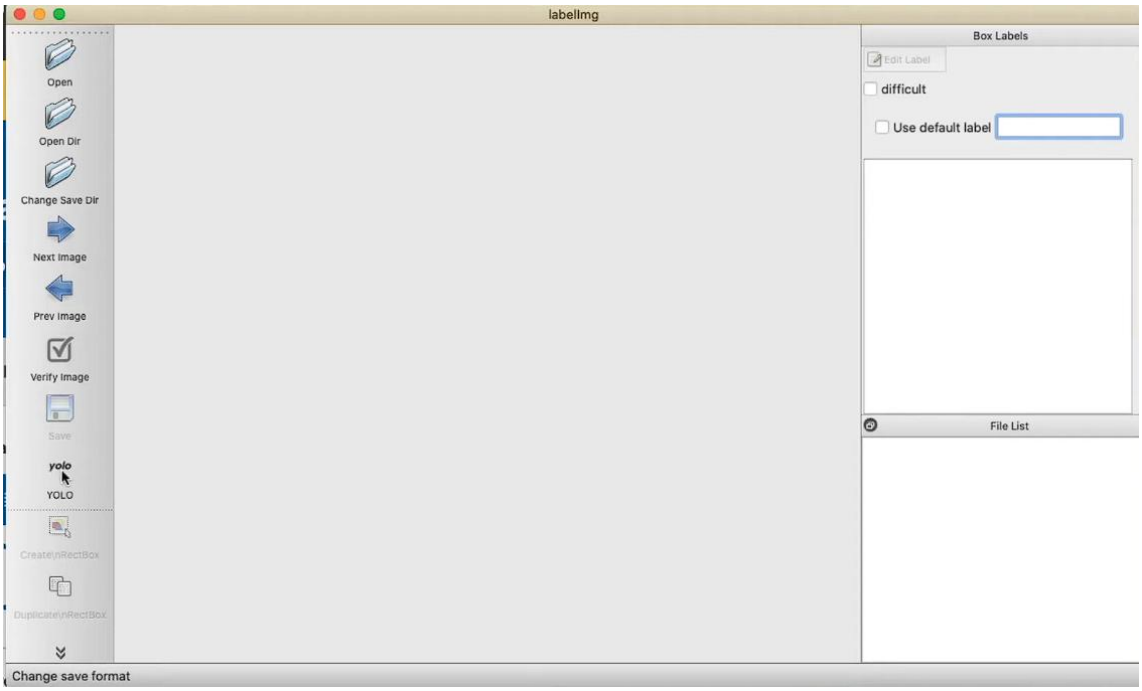


Ilustración 12 Entorno de etiquetado de LabelImg [61]

- **Labelme**: Desarrollada por el MIT, esta herramienta de etiquetado de imágenes es notablemente flexible, permitiendo la creación de anotaciones para diversos propósitos, incluyendo la segmentación semántica [62].

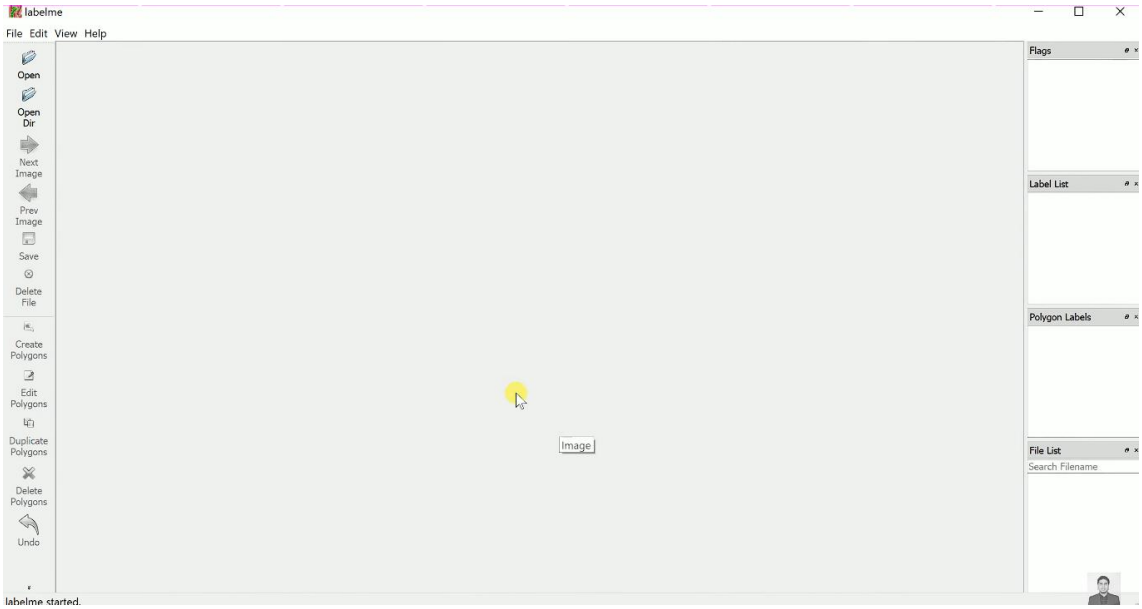


Ilustración 13 Entorno de etiquetado de Labelme [62]

- **VGG Image Annotator (VIA):** Una herramienta ligera de anotación que opera íntegramente en el navegador. Ofrece funcionalidades para etiquetar imágenes en tareas como detección de objetos, segmentación y anotación de puntos [63].

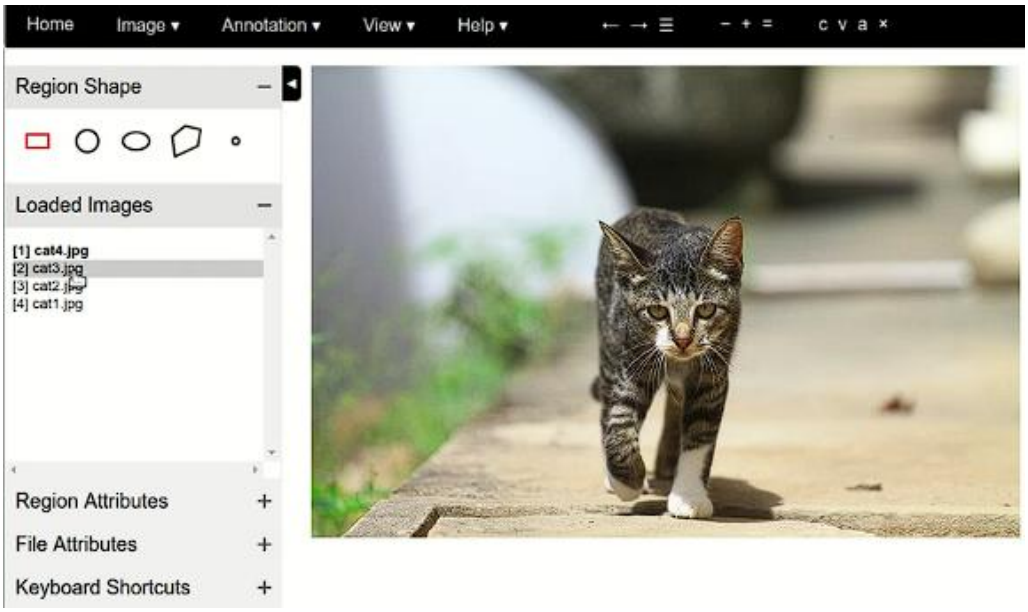


Ilustración 14 Entorno de etiquetado de VGG Image Annotator (VIA) [63]

- **RectLabel**: Esta herramienta de etiquetado de imágenes está especialmente diseñada para usuarios de macOS, siendo compatible con múltiples formatos, incluidos YOLO y Keras [64].

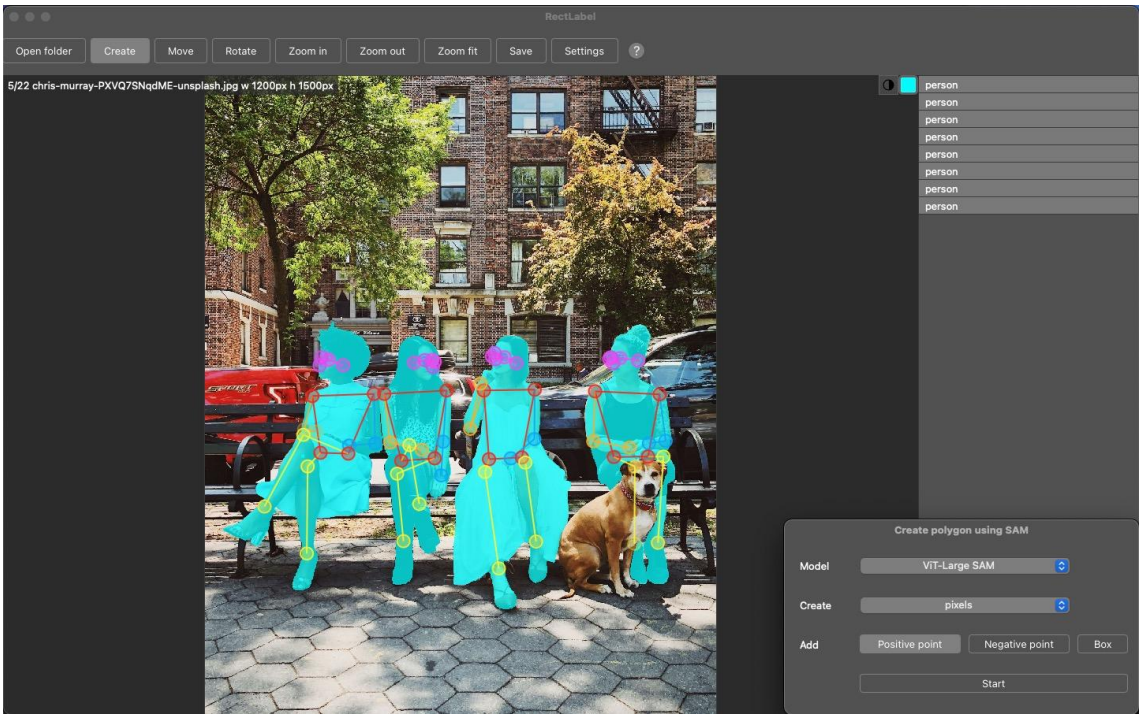


Ilustración 15 Entorno de Etiquetado de RectLabel [64]

- **CVAT (Computer Vision Annotation Tool):** Desarrollada por Intel, CVAT es una herramienta web de anotación sumamente completa, que admite una amplia variedad de tipos de anotaciones, como la detección de objetos y la segmentación [65].

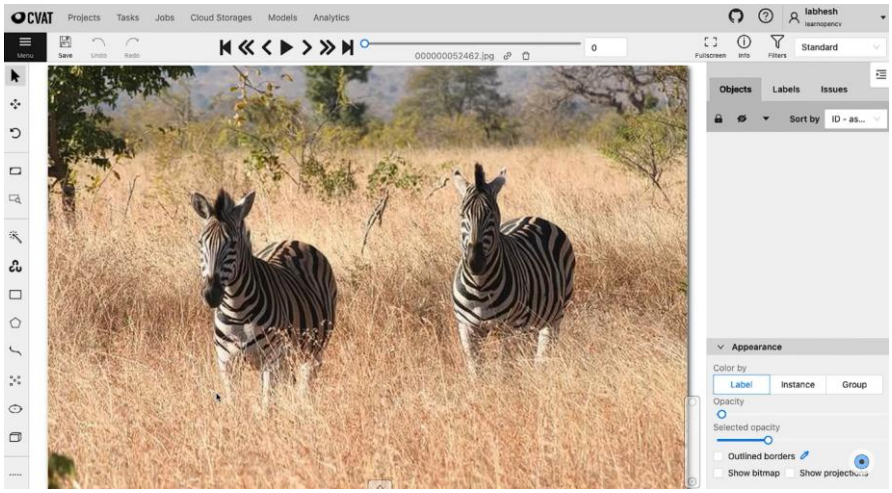


Ilustración 16 Entorno de Etiquetado de CVAT (Computer Vision Annotation Tool) [65]

- **Supervisely**: Plataforma integral para el etiquetado de datos y la creación de conjuntos de datos destinados al entrenamiento de modelos de visión por computadora. Supervisely proporciona herramientas para la anotación colaborativa y es compatible con múltiples formatos [66].

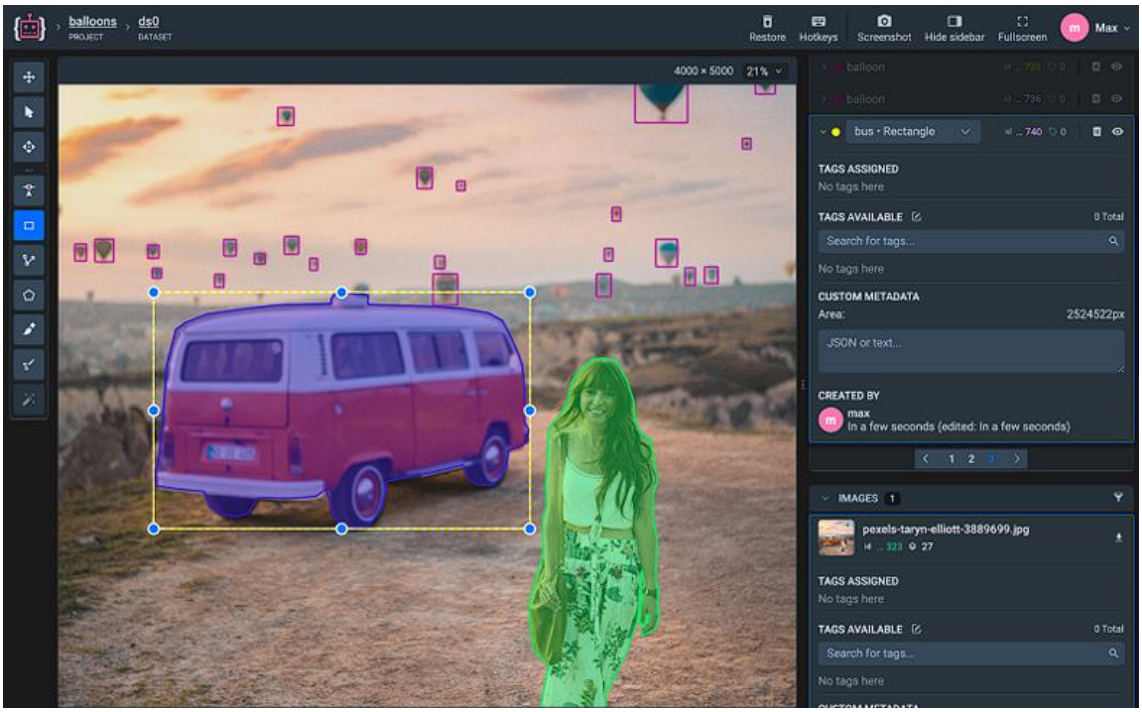


Ilustración 17 Entorno de Etiquetado de supervisely [66]

- **MakeSense.AI**: Herramienta en línea gratuita que facilita el etiquetado de imágenes. Su interfaz es fácil de usar y es compatible con diversos formatos de anotación, incluyendo YOLO [67].

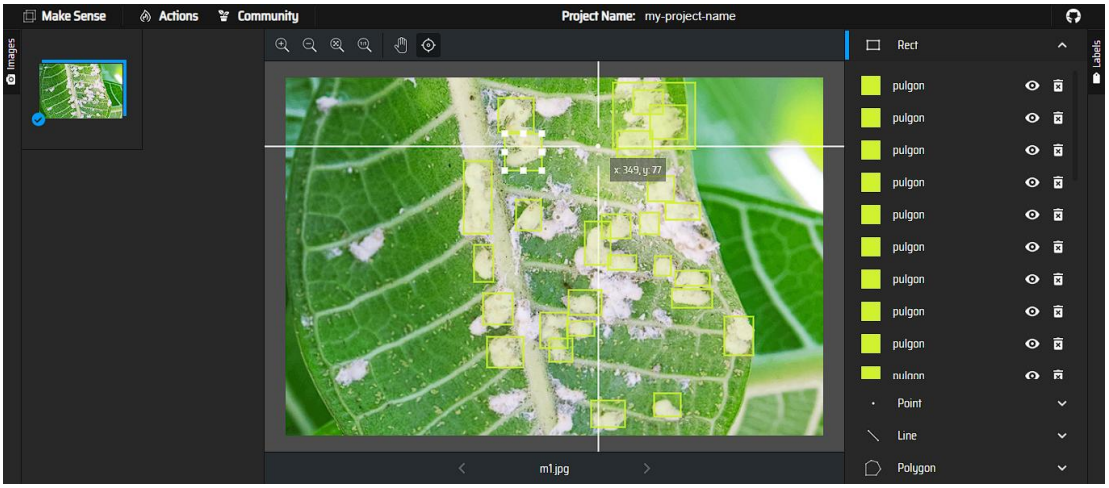


Ilustración 18 Entorno de Etiquetado de makesense.AI [67]

- **RoboFlow**: Plataforma en línea que no solo permite la anotación de imágenes, sino que también facilita la organización de conjuntos de datos y su exportación en diversos formatos listos para el entrenamiento de modelos, como YOLO y TensorFlow [68].

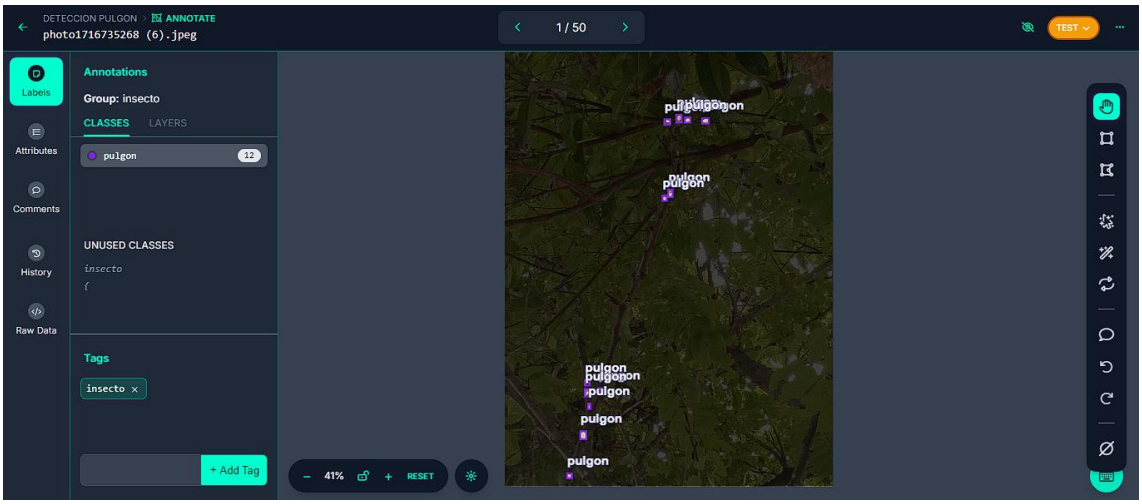


Ilustración 19 Entorno de Etiquetado de roboflow [68]

El programa RoboFlow fue seleccionado para el proyecto actual debido a sus características destacadas, que no solo facilitan su uso, sino que también optimizan la organización de carpetas esenciales para su funcionamiento, la correcta disposición y dirección de estas carpetas son cruciales para el desempeño del programa, ya que una estructura desordenada o ubicaciones incorrectas impedirían al sistema detectar y acceder a la base de datos necesaria para realizar comparaciones y localizar los elementos requeridos [68].

3.2.5) Carga de Muestras en YOLO

Después de haberse realizado el etiquetado de las muestras y haber distribuidos las mismas en las carpetas correspondientes, lo siguiente que se debe de realizar, es la carga de los datos en el sistema de visión por computador mencionado, esto se hace ingresando al siguiente URL.

- https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=BCeFz7Tla7Cs

En este entorno se despliega YOLO, donde es necesario cargar las muestras. Sin embargo, antes de proceder con esta tarea, es imprescindible realizar ciertos ajustes para optimizar el entorno de aprendizaje, es fundamental modificar la configuración de la aceleración para que el sistema aproveche al máximo el hardware disponible, evitando así posibles inconvenientes o interrupciones durante el entrenamiento. A continuación, se ofrecerá una explicación detallada, paso a paso, sobre cómo llevar a cabo esta configuración.

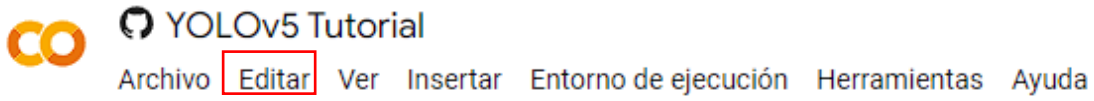


Ilustración 20 Preparación del entorno de entrenamiento de YOLOv5 [69]

En la opción de Editar, nos dirigimos a la opción de configuración de cuaderno, haciendo clic desplegara la siguiente ventana.

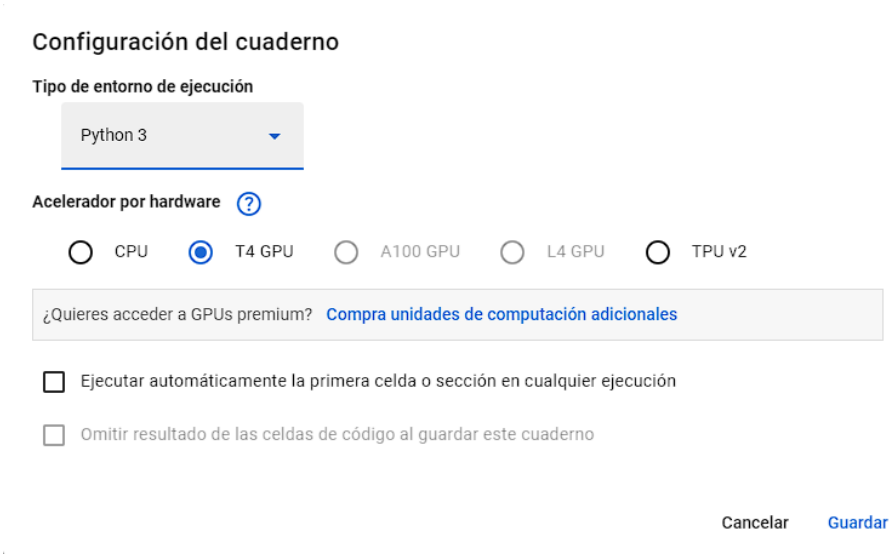


Ilustración 21 Cambio de modo de uso de recursos para la visión por computador [69]

En esta ventana, en el apartado de acelerador por hardware se debe de seleccionar la opción de CPU, para que de esta manera el entorno de Yolov5, haga uso de los recursos disponibles del usuario, para final mente darle a guardar.

Después de realizado esto se debe de ejecutar, el programa dándole clic en el apartado de **Setup,** dando de esta forma inicio el sistema YOLO.

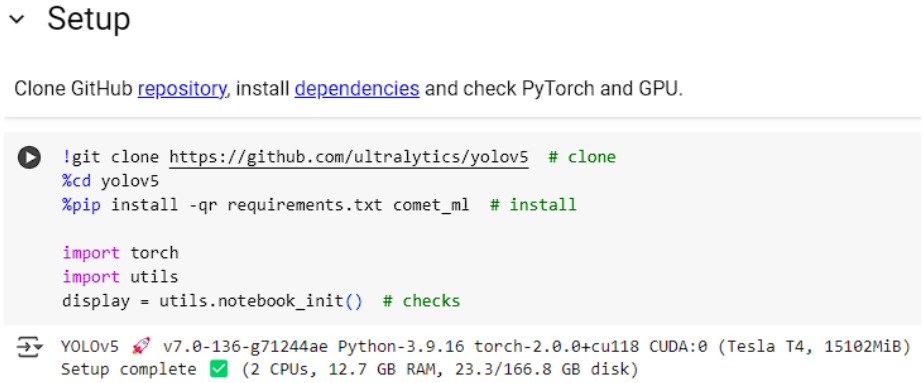


Ilustración 22 Ejecución del modelo YOLOv5 [69]

Una vez se termine de ejecutar, al lado izquierdo del navegador, se desplegará una pestaña emergente dentro del entorno YOLO, donde se encontrarán varias de las carpetas necesarias para el correcto funcionamiento de este.

Ilustración 23 Verificación del correcto arranque deYOLOv5 [69]

Para hacer la importación de los archivos, por debajo de donde se ejecutó el programa se debe de escribir el siguiente comando, en donde se llaman las carpetas de ejecución, aunque algo a tomar en cuenta es que existen dos formas de realizar esta tarea, puesto que dentro del etiquetador usado, se puede realizar el traslado directo de los archivos a YOLO, mediante un código proporcionado directamente por el sistema el cual es el siguiente.

```
[ ]  !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="GZEz31Ic7tFVDvLPl7XB")
     project = rf.workspace("pruebas-kwbwh").project("deteccion-pulgon")
     version = project.version(1)
     dataset = version.download("yolov5")
```

Ilustración 24 Obtención de código para carga de muestras [69]

Este código se debe de ingresar directamente en la parte inferior de donde se ejecutó principal mente el programa.

Ilustración 25 Carga de muestras de roboflow a YOLOv5 [69]

Una vez que se han ingresado los datos, simplemente se debe hacer clic en la flecha de ejecución para que YOLO comience a cargar los archivos previamente etiquetados. Alternativamente, estos archivos también pueden ser cargados directamente en un archivo .zip; sin embargo, después deben ser descomprimidos utilizando la siguiente línea de código.

- !unzip -q /content/nombre del archivo.zip -d /content/

Después de haber completado el proceso de carga de archivos en el programa, el siguiente paso es ejecutar el sistema para que comience el reconocimiento y la generación de la base de datos para el proyecto, para ello, es necesario desplazarse hacia abajo hasta alcanzar la siguiente línea de código.

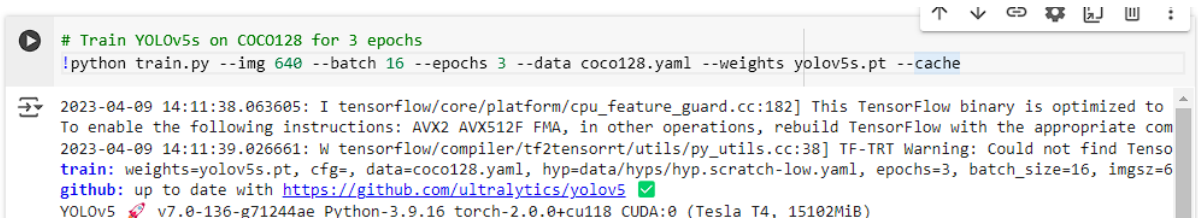


Ilustración 26 Código de entrenamiento para la visión por computador [69]

En esta línea de código lo principal que se debe de cambiar son dos cosas, el número de repeticiones para el entrenamiento y la ubicación de los archivos, que se requieren para el mismo, quedando de la siguiente manera.

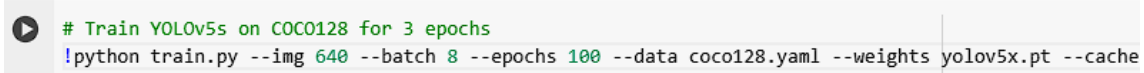


Ilustración 27 Modificación del modelo de entrenamiento de YOLOv5 [69]

El número de repeticiones, o ***epochs***, es un factor crucial, ya que determina cuántas veces el programa repetirá el proceso de entrenamiento para desarrollar un algoritmo de detección confiable. Además, es fundamental configurar correctamente la ruta de los archivos, lo cual implica especificar qué se busca y cómo se debe buscar, en particular, a través del archivo yaml, que está directamente relacionado con el etiquetado de la base de datos proporcionada.

Ya habiendo terminar el proceso de entrenamiento lo que se debe de hacer, es verificar la precisión del programa para lo cual se aplica el siguiente comando por debajo de la ejecución realizada con anterioridad.

- %load_ext tensorboard
- %tensorboard --logdir runs

Después de verificar que las métricas indiquen que el programa tiene una efectividad aceptable, lo siguiente es descargar el archivo de entrenamiento completado, por ello se hace uso del siguiente comando.

- from google.colab import files
- files.download('./runs/train/exp/weights/best.pt')

Tras ejecutar este comando, aparecerá una barra de progreso debajo del mismo, que permitirá visualizar de manera más precisa el avance de la descarga del archivo de entrenamiento completado.

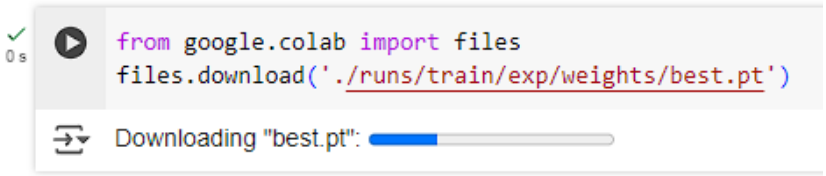


Ilustración 28 Descarga del modelo de entrenamiento ya finalizado [69]

Una vez completados todos estos pasos y con el archivo base de entrenamiento ya disponible, el siguiente paso es generar el código que permitirá ejecutar y analizar dicho archivo. Esto garantizará una correcta detección del objeto, que en este caso es el pulgón, pero antes es importantes entender de forma breve cómo funciona el programa de detección, con respecto al entrenamiento que debe de tener.

3.2.6) Funcionamiento de YOLOv5

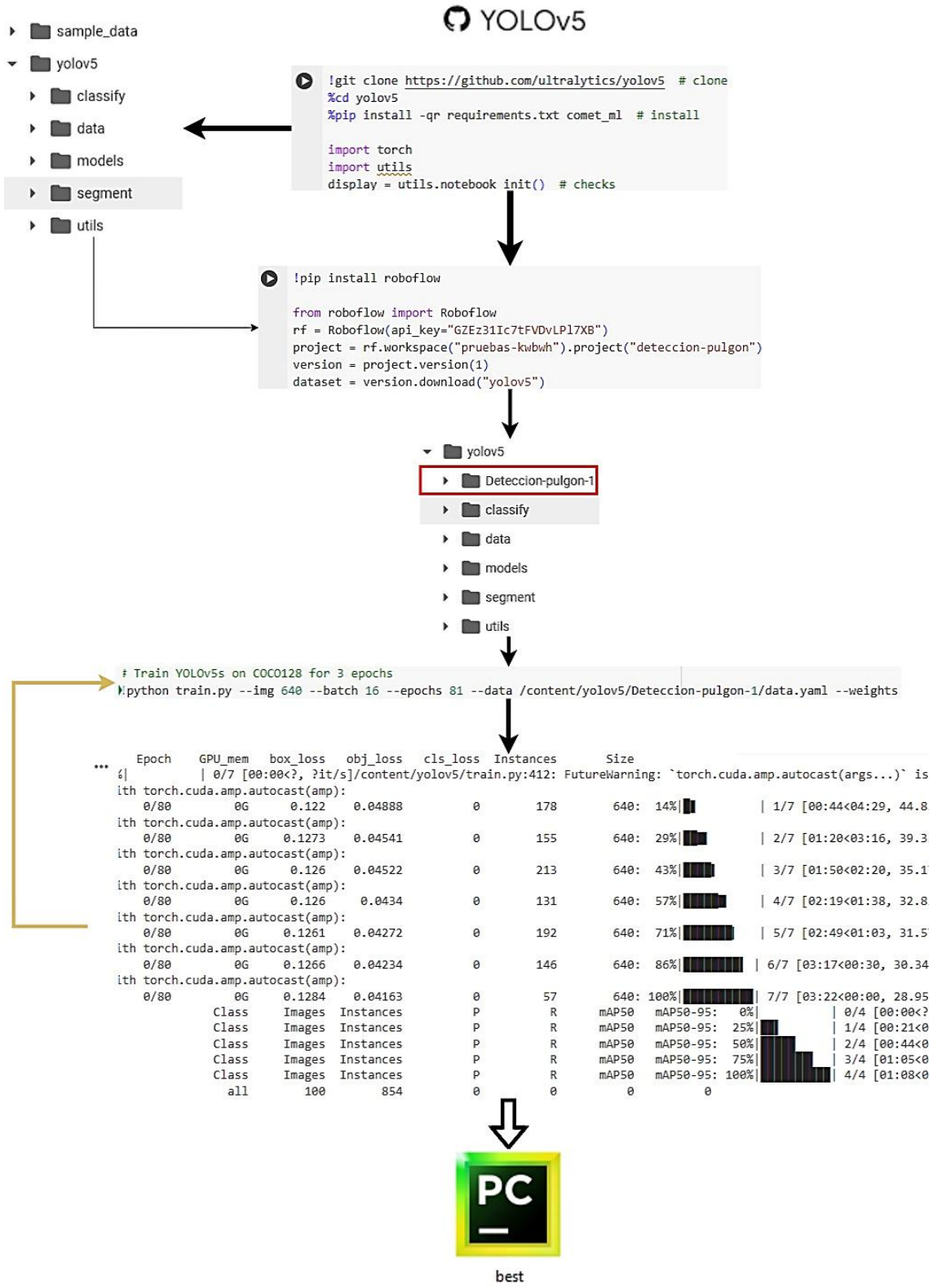


Ilustración 29 Funcionamiento de YOLOv5 [69]

En la ilustración 29 se presenta un ejemplo del funcionamiento del programa de inteligencia artificial y visión por computadora YOLOv5, esta ilustración no solo muestra

cómo se ejecuta el programa, sino también el proceso de carga de datos y el entrenamiento del algoritmo para la detección de elementos específicos, en este caso, el pulgón. Tras la carga inicial de datos, el sistema entra en un ciclo en el que identifica las coordenadas del objeto a detectar, utilizando las muestras proporcionadas previamente, que están organizadas en diferentes carpetas. A partir de estas muestras, se extraen los datos necesarios y se identifican las características relevantes, lo que permite al sistema completar el ciclo de detección de manera eficiente.

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

Ilustración 30 Clonación del repositorio de YOLOv5 [69]

En la ilustración se muestra que el repositorio de YOLOv5 se está clonando desde GitHub usando el comando git clone, también se instalan las dependencias necesarias con pip install.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="GZEz31Ic7tFVDvLPl7XB")
project = rf.workspace("pruebas-kwbwh").project("deteccion-pulgon")
version = project.version(1)
dataset = version.download("yolov5")
```

Ilustración 31 Configuración del entorno de trabajo [69]

Hay un ejemplo de cómo se configura el entorno de trabajo para ejecutar el modelo en un notebook. Esto incluye la importación de bibliotecas y la inicialización de ciertos módulos, probablemente relacionados con la visualización de resultados.

Ilustración 32 Estructura de carpetas [69]

La imagen incluye una vista de la estructura de carpetas del proyecto, que seguí una organización típica de proyectos de aprendizaje profundo, con carpetas para datos, modelos, scripts, y utilidades.

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 81 --data /content/yolov5/Deteccion-pulgon-1/data.yaml --weights
```

Ilustración 33 Entrenamiento del modelo [69]

Como se ve en la ilustración 33 el comando utilizado para entrenar el modelo YOLOv5 en un conjunto de datos específico (COCO128), indicando parámetros como el tamaño de la imagen (--img 640), el tamaño del lote (--batch 16), el número de épocas (--epochs 3), y el archivo de configuración de datos (--data .../Deteccion-piñas/data.yaml).

```
Using 2 dataloader workers
Logging results to runs/train/exp
Starting training for 80 epochs...

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
  0%|          | 0/7 [00:00<?, ?it/s]/content/yolov5/yolov5/train.py:412: FutureWarning: `torch.cuda.amp.autocast(args..
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1241    0.05031          0        166       640:  14%|▉         | 1/7 [00:56<05:38, 56.43
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1252    0.04504          0        160       640:  29%|██        | 2/7 [01:27<03:26, 41.37
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1238    0.04494          0        149       640:  43%|███       | 3/7 [01:57<02:24, 36.08
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1236    0.04384          0        163       640:  57%|████      | 4/7 [02:27<01:41, 33.84
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1249    0.04361          0        180       640:  71%|█████     | 5/7 [02:58<01:05, 32.70
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1258    0.04237          0        127       640:  86%|██████    | 6/7 [03:25<00:30, 30.93s
  with torch.cuda.amp.autocast(amp):
       0/79         0G     0.1266    0.04085          0         23       640: 100%|███████| 7/7 [03:34<00:00, 30.57s
                 Class     Images  Instances          P          R      mAP50   mAP50-95:   0%|          | 0/4 [00:00<?,
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  25%|▊         | 1/4 [00:19<00
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  50%|▌         | 2/4 [00:38<00
                 Class     Images  Instances          P          R      mAP50   mAP50-95:  75%|▊         | 3/4 [00:57<00
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|▌         | 4/4 [01:01<00
                   all        100        854          0          0          0          0
```

Ilustración 34 Registro del progreso del entrenamiento [69]

A la derecha, se muestra una salida del progreso del entrenamiento con métricas como la pérdida de validación (val), precisión, recall, y mAP (mean Average Precision) por clase. Estas métricas son útiles para evaluar el desempeño del modelo a medida que se entrena.

Ilustración 35 Archivo .pt de entrenamiento [69]

Después de haberse realizado el bucle de entrenamiento, de este mismo sistema se extrae el archivo reconocido como best.pt, ah donde este últzimo vendría a ser el resultado del entrenamiento, realizado por Yolov5 para poder detectar lo que vendría a ser para este caso el pulgón.

3.2.7) Generación de Código

El código base debe considerar las bibliotecas mencionadas previamente, ya que estas son fundamentales para el correcto funcionamiento del programa y se deben incorporar al inicio del mismo. Es crucial que tanto el archivo de entrenamiento como el programa que se pretende generar se encuentren en la misma carpeta, pues de lo contrario, el programa podría presentar fallos al no lograr identificar correctamente el archivo requerido.

Después de haber tomado encuentra lo anteriormente mencionado, lo que sigue es la generación del propio código para la cual se hizo de la siguiente manera tomando en cuenta, varios factores que pueden llegar a influir en la propia programación.

```
77  import cv2
78  import numpy as np
79  import torch
80  from PIL import Image
81  import os
82  import pathlib
83  from pathlib import Path
```

Ilustración 36 Importación de librerías dentro del código

Como primer punto para la generación del código lo primero es el llamado a las librerías que van a ser requeridas, puesto que las mismas permitirán al programa ejecutarse de forma correcta, y de manera que no existan conflictos en el sistema al momento de ejecutarse, es importante destacar que no todas las librerías vienen incluidas dentro de Python, por lo cual es recomendable verificarlas y descargarlas/actualizarlas según sea necesario.

```
10  pathlib.PosixPath = pathlib.WindowsPath
```

Ilustración 37 Generación de código para garantizar la compatibilidad de las rutas

Esta parte del código se encarga de garantizar, la compatibilidad de los recursos, como son las librerías, de forma que, si las mismas no son compatibles directamente con Windows, lo que hace este, comando es permitir la compatibilidad de recursos con el entorno virtual asignado, siendo para este caso Windows.

```
13  device = "cuda" if torch.cuda.is_available() else "cpu"
```

Ilustración 38 Línea de código encargada de la configuración los dispositivos de computación

Esta línea de código que configura el dispositivo en el que se ejecutarán las operaciones de computación esta configuración es relevante cuando se trabaja con bibliotecas de aprendizaje profundo como PyTorch.

*relative_model_path = **'C:\\Users\\gr072\\Desktop\\pruebas\\pulgon_detection_project\\best.pt'***

La primera línea se encarga de llamar, al código pre entenado, por ello se debe de verificar la ruta en la cual se encuentra el archivo .tp ya que, sin éste, el programa no podría identificar/buscar lo requerido por el usuario.

*model_yolo=torch.hub.load('**ultralytics/yolov5**','**custom**',path=relative_model_path,force_reload=True)*

La segunda línea se encarga de llamar al modelo YOLOv5 haciendo uso de la API '**torch.hub**' que es una función de '**PyTorch'** que permite cargar modelos desde un repositorio de GitHub o desde una ruta local. '**torch.hub**' se utiliza para facilitar la descarga y carga de modelos preentrenados o personalizados, además de eso el ***'ultralytics/yolov5'*** es un repositorio popular mantenido por Ultralytics que incluye implementaciones de YOLOv5.

El **'custom'** indica que estás cargando un modelo personalizado. Esto es necesario cuando estás cargando un modelo que no es uno de los modelos predefinidos en el repositorio de un GipHub, ademas de esto es necesario la colocación de la ruta de donde se encuentra el archivo para lo cual se hace uso de '**path=relative_model_path**' que especifica la ruta al archivo del modelo que deseas cargar, se utiliza la variable relative_model_path para proporcionar la ruta del archivo del modelo preentrenado, ara final mente hacerse uso de '**force_reload=True**' que fuerza la recarga del modelo incluso si ya está en caché esto asegura que se cargue la versión más reciente del modelo.

Todo esto se encontrará contenido dentro de '**Model_YOLOv5**' que tendrá una instancia del modelo YOLOv5 cargado desde el archivo especificado, puedes utilizar este modelo para realizar inferencias en imágenes o videos.

```
20    classes = []
21    with open('classes.txt', 'r') as f:
22        for line in f:
23            classes.append(line.strip())
```

Ilustración 39 Especificación del nombre de la clase

Este código lee todas las líneas del archivo classes.txt y las almacena en la lista 'classes', eliminando cualquier espacio en blanco adicional que pueda haber en cada línea. La lista 'classes' contendrá todas las etiquetas de las clases que serán utilizadas posteriormente en el programa, por ejemplo, para interpretar las predicciones de un modelo de aprendizaje automático.

classes: Esta línea de código inicia el proceso creando una lista vacía llamada classes. Esta lista tiene un propósito muy específico: servirá como un contenedor donde se almacenarán todas las clases que se leerán desde un archivo llamado classes.txt. Básicamente, esta lista es como un organizador que espera ser llenado con la información importante que se extraerá de ese archivo.

with open('classes.txt', 'r') as f: Esta instrucción realiza una tarea fundamental en el código, ya que se encarga de abrir el archivo classes.txt en modo de lectura, lo cual se indica con la letra 'r'. La acción de abrir el archivo se asocia con la variable f, que actúa como un identificador del archivo durante el proceso de lectura. Un aspecto importante es que el uso de la construcción with garantiza que el archivo se cierre automáticamente una vez que se termina de trabajar con él, lo que es muy útil porque evita problemas como archivos que quedan abiertos por error, incluso si ocurre algún imprevisto durante la ejecución del código.

for line in f: Este bucle for desempeña un papel crucial, ya que itera sobre cada una de las líneas contenidas en el archivo classes.txt. En cada iteración, se toma una línea del archivo y se asigna a la variable line, permitiendo que el código procese esa línea de manera individual. De esta manera, el bucle asegura que se revisen y manejen todas las líneas del archivo, una por una, hasta que no quede ninguna sin procesar.

classes.append(line.strip()): En esta línea del código, se realizan dos operaciones importantes en secuencia. Primero, se utiliza el método '**strip()**' para eliminar cualquier espacio en blanco o carácter de nueva línea que pueda estar presente al principio o al final de la línea actual. Esto es importante porque asegura que la información sea almacenada de manera limpia y sin caracteres innecesarios. Después de limpiar la línea, el método '**append()**' se encarga de agregarla a la lista 'classes'. Cada línea, que en este contexto representa una clase diferente, se almacena en esta lista, creando así un conjunto organizado de todas las clases que estaban listadas en el archivo 'classes.txt'.

```
26    output_dir = 'predicted_output'
27    os.makedirs(output_dir, exist_ok=True)
```

Ilustración 40 Generación del archivo de almacenamiento de resultados

El código verifica si el directorio **'predicted_output'** existe. Si no existe, lo crea. Si ya existe, no hace nada gracias a '**exist_ok=True**'. Esto es útil para asegurarse de que el

programa siempre tenga un directorio en el que guardar los resultados sin preocuparse por la existencia previa del mismo.

El '**output_dir** = '**predicted_output**' usa una variable llamada '**output_dir**' que almacena el nombre del directorio donde se guardarán las imágenes o videos procesados (en este caso, **'predicted_output'**).

La función '**os.makedirs(output_dir, exist_ok=True)**' se utiliza para crear un directorio en el sistema de archivos especificado por '**output_dir'**. El parámetro '**exist_ok=True**' asegura que, si el directorio ya existe, no se producirá un error, permitiendo que el programa continúe sin interrupciones y evitando errores como '**FileExistsError'**.

```
30    def process_frame(frame):
31        img_np = np.array(frame)
```

Ilustración 41 Procesamiento y Anotación del programa

La línea '**def process_frame(frame)**': define una función llamada '**process_frame**' que toma un argumento llamado '**frame'** en Python, la palabra clave '**def'** se utiliza para crear una nueva función, el parámetro '**frame'** representa una imagen o un fotograma de un video, que generalmente se espera esté en formato PIL (Python Imaging Library). Esta función se utiliza en el procesamiento de imágenes, donde cada fotograma o imagen es tratado individualmente para realizar diversas operaciones.

Por otro lado, la línea '**img_np = np.array(frame)**' convierte el argumento '**frame'** en un array de NumPy utilizando '**np.array()'**, NumPy es una biblioteca de Python diseñada para manipular eficientemente arreglos multidimensionales. Al convertir la imagen o fotograma en un array de NumPy, se facilita la aplicación de operaciones matemáticas y manipulaciones complejas de manera más rápida y eficiente. Esta conversión es fundamental, ya que muchos algoritmos de procesamiento de imágenes y aprendizaje automático en Python trabajan mejor con arrays de NumPy que con objetos de imagen de PIL.

```
34    results = model_yolo(img_np)
```

Ilustración 42 Utilización de interferencias por parte de YOLOv5

La línea '**results =**' está asignando la salida de una operación a la variable '**results'**, que almacenará los resultados del modelo YOLOv5 después de realizar una predicción. Este paso es crucial para capturar la información generada por el modelo una vez que ha analizado una imagen en busca de objetos de interés. El modelo utilizado, '**model_yolo'**, es una instancia previamente cargada del modelo YOLOv5, conocido por su capacidad para detectar y localizar objetos en imágenes de manera eficiente.

En este caso, '**model_yolo(img_np)**' se refiere a la ejecución del modelo YOLOv5 sobre la imagen representada por '**img_np'**, que es un array de NumPy, este array se creó a partir de un objeto de imagen PIL, convertido previamente para facilitar su manipulación y análisis mediante algoritmos de procesamiento de imágenes. Cuando el modelo realiza la inferencia sobre '**img_np'**, analiza la imagen para identificar objetos y generar un conjunto de datos de resultados. Estos resultados incluyen detalles como las coordenadas de los cuadros delimitadores (bounding boxes) que rodean los objetos detectados, las clases de estos objetos (por ejemplo, "persona", "automóvil", "animal"), y las puntuaciones de confianza asociadas a cada predicción, indicando cuán seguro está el modelo sobre la presencia de cada objeto.

```
predicted_classes = [classes[int(label)] for label in results.xyxy[0][:, -1]]
```

Ilustración 43 Traducción de los índices de clases numéricas

En esta línea de código, estamos generando una lista de nombres de clases predichas para los objetos detectados en una imagen. Vamos a desglosarlo paso a paso para entender mejor lo que está sucediendo:

results.xyxy[0][:, -1]: Primero, accedemos a los resultados de la detección que nos proporciona el modelo YOLO, estos resultados están organizados en una matriz o tensor, donde cada fila representa una detección diferente. Las columnas de esta matriz contienen información importante como las coordenadas del cuadro delimitador (bounding box) y la etiqueta de clase para cada detección.

[:, -1]: Aquí, usamos la notación de '**slicing**' para seleccionar todas las filas (**:**) pero solo la última columna (**-1**). Esta última columna contiene los índices de clase, que son números enteros que representan las clases de los objetos detectados.

[int(label) for label in results.xyxy[0][:, -1]]: Este es un bucle de lista (list comprehension) que recorre cada etiqueta de clase (label) en la columna seleccionada, convertimos cada etiqueta en un número entero usando '**int(label)**'. Esto es importante porque la información extraída podría estar en un formato diferente, como flotantes, y queremos asegurarnos de que sea un entero para usarlo como índice en nuestra lista de clases.

classes[int(label)]: Usamos el índice de clase para buscar el nombre correspondiente en nuestra lista de nombres de '**clases (classes)**'. Por ejemplo, si '**int(label)**' es '**2**' y '**classes[2]**' es "carro", entonces el nombre "carro" será añadido a nuestra lista de clases predichas.

predicted_classes = [...]: Finalmente, asignamos esta lista de nombres de clases predichas a la variable '**predicted_classes**'. Esta variable ahora contiene una lista de cadenas de texto, cada una representando una clase que el modelo YOLO ha detectado en la imagen.

```
40      img_cv2 = cv2.cvtColor(np.array(frame), cv2.COLOR_RGB2BGR)
```

Ilustración 44 Conversión de imagen de RGB a BGR utilizando OpenCV

Cuando se trabaja con imágenes en Python, es común utilizar la biblioteca PIL (**Python Imaging Library**) o su versión más moderna, Pillow, para cargar y manipular imágenes en el formato RGB (**Red, Green, Blue**), el formato BGR de OpenCV puede parecer inusual porque invierte el orden de los canales de color en comparación con el formato RGB más común. Esta inversión se debe a la forma en que OpenCV fue diseñado originalmente para trabajar con imágenes almacenadas en el estándar de la cámara y las interfaces de video, que a menudo usan BGR.

Para realizar esta conversión, utilizamos la función '**cv2.cvtColor**' de OpenCV, que permite cambiar el espacio de color de una imagen de RGB a BGR. En el contexto del código, primero se convierte el marco o la imagen a un arreglo NumPy para asegurar que esté en un formato compatible con OpenCV. Luego, la función '**cv2.cvtColor(np.array(frame)**', **cv2.COLOR_RGB2BGR)**' se aplica a la imagen para realizar la conversión.

Este paso es crucial para evitar errores y garantizar que los colores se interpreten correctamente durante el procesamiento posterior, como la detección de objetos o el

análisis de características. Sin esta conversión, los colores aparecerían incorrectos en cualquier operación de visualización o análisis, lo que podría llevar a resultados inexactos y a una interpretación incorrecta de los datos visuales.

Por lo tanto, cuando se integra OpenCV con otras bibliotecas de procesamiento de imágenes, es fundamental entender las diferencias en el manejo de los espacios de color y hacer las conversiones necesarias, este conocimiento asegura que las operaciones de procesamiento de imágenes se realicen correctamente, preservando la precisión del análisis y la visualización.

```
43    for idx, (box, conf, cls) in enumerate(zip(results.xyxy[0][:, :4], results.xyxy[0][:, 4], results.
44        x1, y1, x2, y2 = map(int, box)
45        predicted_class = predicted_classes[idx]
```

Ilustración 45 Iteración sobre Predicciones de Detección de Objetos con YOLOv5

Esta línea inicia un bucle que iterará a través de cada detección realizada por el modelo. '**enumerate()**' no solo te permite iterar, sino que también te proporciona un índice (**idx**) para cada elemento en la iteración. Este índice puede ser útil si necesitas hacer referencia a la posición de un elemento en particular

Dentro del bucle, la función '**zip()**' se utiliza para combinar tres conjuntos de datos diferentes en una sola estructura iterable, permitiendo iterar sobre ellos simultáneamente. Estos conjuntos de datos provienen de los resultados generados por el modelo YOLOv5 después de la detección de objetos en una imagen. El primer conjunto, '**results.xyxy[0][:, :4]**', contiene las coordenadas de los cuadros delimitadores (bounding boxes) que especifican la ubicación de cada objeto en la imagen. Cada cuadro se define por cuatro valores: x1, y1 (esquina superior izquierda) y x2, y2 (esquina inferior derecha).

El segundo conjunto, '**results.xyxy[0][:, 4]**', incluye las puntuaciones de confianza del modelo para cada objeto detectado, expresadas como valores entre 0 y 1 estas puntuaciones indican la certeza del modelo acerca de la presencia del objeto en la imagen y su clasificación correcta.

El tercer conjunto, '**results.xyxy[0][:, 5]**', contiene las clases predichas, que son números enteros que corresponden a los índices de las etiquetas de clase, como "0" para "perro" o "1" para "gato". Estos tres conjuntos de datos se combinan para dibujar cuadros delimitadores alrededor de los objetos detectados y etiquetarlos adecuadamente con su clase y confianza asociada.

```
48        color = (255, 0, 0)  # Color
49        cv2.rectangle(img_cv2, (x1, y1), (x2, y2), color, 2)
```

Ilustración 46 Generación de cuadro delimitador y selección de color

En la línea 48, se define una variable llamada '**color**' que se utiliza para especificar el color del cuadro delimitador que se dibujará en la imagen, el color se define como una tupla '**(255, 0, 0)**', que representa un color en formato BGR (azul, verde, rojo) que utiliza OpenCV. En este caso, '**(255, 0, 0)**' indica un color rojo puro porque el valor del canal azul es 255, mientras que los valores de los canales verde y rojo son 0.

Esta la línea 49 utiliza la función '**cv2.rectangle()**' de OpenCV para dibujar un cuadro delimitador en la imagen procesada. Vamos a desglosar los parámetros que se pasan a esta función '**img_cv2**' es la imagen en la que se dibujará el cuadro delimitador. En este caso, es una imagen que ha sido procesada previamente y almacenada en la variable **'img_cv2**'

- (x1, y1): Estas son las coordenadas de la esquina superior izquierda del cuadro delimitador. Es el punto de partida desde donde comenzará a dibujarse el cuadro.
- (x2, y2): Estas son las coordenadas de la esquina inferior derecha del cuadro delimitador. Junto con (x1, y1), definen el área rectangular que se dibujará.

'**Color**' Esta es la variable definida anteriormente que especifica el color del cuadro. Aquí, el cuadro se dibujará en rojo y '**2**' este último parámetro especifica el grosor del borde del cuadro delimitador en píxeles. En este caso, se usa un grosor de 2 píxeles.

```
52        label = f'{predicted_class} {conf:.2f}'
53        label_size, _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, thickness: 2)
54        label_y1 = max(y1, label_size[1] + 10)
55        cv2.putText(img_cv2, label, org: (x1, label_y1), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, c
56
57    return img_cv2
```

Ilustración 47 Agregar una Etiqueta con Texto y Confianza a una Imagen Usando OpenCV

En la línea 52 se está creando una cadena de texto (**label**) que incluye dos partes '**predicted_class**' y '**conf**'. '**predicted_class**' es la clase predicha para el objeto que se está etiquetando, y conf es un valor de confianza asociado con esa predicción, la sintaxis '**:.2f**' asegura que el valor de confianza se muestre con dos decimales. La cadena resultante podría ser algo como "**pulgón 0.87**" si predicted_class es "**pulgón**" y '**conf**' es '**0.87**'.

En la línea 53 se calcula el tamaño (ancho y alto) del texto '**label**' usando la fuente '**cv2.FONT_HERSHEY_SIMPLEX**', con un tamaño de fuente de '**0.5**' y un grosor de '**2**' la función '**cv2.getTextSize**' devuelve el tamaño del texto en píxeles, que se almacena en **'label_size'** la variable '_' se utiliza para descartar el segundo valor de retorno que no se usa en este caso.

La línea 54 se ajusta la posición vertical '**(label_y1)**' para asegurarse de que el texto no se dibuje fuera del área visible, '**y1**' es la coordenada vertical del punto de inicio para la etiqueta, y '**label_size[1] + 10**' es el alto del texto más un margen adicional de 10 píxeles, la función max se usa para elegir el mayor valor entre '**y1'** y **'label_size[1] + 10**', lo que garantiza que la etiqueta se dibuje por encima del área de interés.

Finalmente, en la línea 45 dibuja el texto label en la imagen '**img_cv2**', la posición del texto se define por las coordenadas '**(x1, label_y1)**', la fuente es '**cv2.FONT_HERSHEY_SIMPLEX**', el tamaño de la fuente es '**0.5**', el color del texto es '**color**', y el grosor del texto es '**2**'.

```
60    input_type = int(input("Precione 1 para imagen, 2 para video, 0 para URL: "))
```

Ilustración 48 Ingreso de opción por teclado

El valor ingresado por el usuario se convierte a un número entero utilizando la función '**int**', esto es necesario porque la función '**input**' devuelve un valor de tipo '**str**' (cadena de texto), y necesitamos convertirlo a '**int**' para poder usarlo como un número en el resto del código.

```
62    if input_type == 1:
```

Ilustración 49 Verificación de la entrada para imagen

En la línea 62 se verifica si el usuario selecciono '1', lo que significa que quiere hacer la identificación de una carpeta de imágenes.

```
64        image_folder_path = input("Introduzca la ruta a la carpeta de imágenes: ").strip()
```

Ilustración 50 Solicitud de ubicación del archivo

Para la línea 64 se pide al usuario introducir la ruta de la carpeta en donde se encuentras las imágenes que se van a procesar, tomando en cuenta la posibilidad de errores por mal ingreso de la ruta se hace uso de, el método '**strip()**' se utiliza para eliminar cualquier

espacio en blanco adicional al inicio o al final de la ruta ingresada, asegurando que no haya errores al buscar la carpeta.

```
66    not os.path.exists(image_folder_path):
67     raise FileNotFoundError(f'La ruta de la carpeta de imágenes no existe: {image_folder_path}')
```

Ilustración 51 Comprobación de la existencia de la carpeta

En las líneas 66 y 67 antes de cualquier procesamiento, el programa verifica si la carpeta ingresada realmente existe usando '**os.path.exists()**', si la carpeta no existe, el código genera un error ('FileNotFoundError') y muestra un mensaje indicando que la ruta proporcionada no es válida, esto es para evitar errores más adelante si se intenta acceder a una carpeta inexistente.

```
69        for image_filename in os.listdir(image_folder_path):
```

Ilustración 52 Iteración de archivos

La línea 69 comienza un bucle' **for**' que recorrerá todos los archivos en la carpeta especificada por el usuario, '**os.listdir()**' devuelve una lista de todos los archivos y carpetas en la ruta proporcionada.

```
70        if image_filename.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
```

Ilustración 53 Filtro de archivos aceptados

Dentro del bucle, esta línea verifica si los archivos actuales son de imágenes, lo hace comprobando si el nombre del archivo termina con alguna de las extensiones de imagen comúnmente usadas, como .png, .jpg, .jpeg, .bmp, o .gif. Esta verificación es importante para asegurarse de que solo se procesen archivos de imagen.

```
71            image_path = os.path.join(image_folder_path, image_filename)
72            img = Image.open(image_path).convert('RGB')
```

Ilustración 54 Procesamiento de imágenes

En las líneas 71 y 72 el archivo es una imagen, se crea una ruta completa al archivo usando '**os.path.join()**', luego la imagen se abre con '**Image.open()**' y se convierte a formato RGB con '**.convert('RGB')**'. este paso es necesario para asegurarse de que la imagen esté en un formato adecuado para el procesamiento posterior.

```
73            annotated_img_filename = f'annotated_{os.path.splitext(image_filename)[0]}.jpg'
74            annotated_img_path = os.path.join(output_dir, annotated_img_filename)
```

Ilustración 55 Generación de nombre y ruta

A continuación, el código crea un nuevo nombre de archivo para la imagen procesada, añadiendo el prefijo '**annotated_**' al nombre original (sin la extensión), '**os.path.splitext(image_filename)[0]**' obtiene el nombre del archivo sin su extensión, lo que facilita la creación de un nombre nuevo y limpio, luego, '**os.path.join(output_dir, annotated_img_filename)'** se utiliza para crear la ruta completa donde se guardará la imagen anotada.

```
75      annotated_frame = process_frame(img)
76      cv2.imwrite(annotated_img_path, annotated_frame)
77      print(f'Imagen anotada guardada: {annotated_img_path}')
```

Ilustración 56 Procesamiento y guardado de imágenes

Aquí, la función '**process_frame(img)**' se llama para realizar alguna operación en la imagen, como añadir anotaciones, el resultado es '**annotated_frame**', después '**cv2.imwrite()**' guarda la imagen procesada en la ruta especificada. Finalmente, se imprime un mensaje confirmando que la imagen anotada ha sido guardada correctamente.

```
78      print('Procesamiento de imagen completado.')
```

Ilustración 57 Guardado de imágenes ya etiquetadas

Una vez que se han procesado todas las imágenes de la carpeta, se imprime un mensaje indicando que el procesamiento ha terminado.

```
80      elif input_type == 2:
```

Ilustración 58 Verificación de la entrada para video

En la línea 80 se verifica si el usuario selecciono '2', lo que significa que quiere hacer la identificación de un video.

```
82      video_path = input("Introduzca la ruta al archivo de video: ").strip()
```

Ilustración 59 Solicitud de ingreso para video

Para la línea 82 se pide al usuario introducir la ruta de la carpeta en donde se encuentra el video que se van a procesar, tomando en cuenta la posibilidad de errores por mal ingreso

de la ruta, se hace uso de, el método '**strip()**' se utiliza para eliminar cualquier espacio en blanco adicional al inicio o al final de la ruta ingresada, asegurando que no haya errores al buscar la carpeta.

```
84    if not os.path.isfile(video_path):
85        raise FileNotFoundError(f'El archivo de video no existe: {video_path}')
```

Ilustración 60 Verificación de la existencia del archivo de video

Antes de hacer cualquier procesamiento, el código verifica si el archivo de video proporcionado realmente existe usando '**os.path.isfile(video_path)**', si el archivo no existe, se genera un error (**FileNotFoundError**) para alertar al usuario de que no se puede encontrar el archivo de video en la ruta especificada. Esto es una buena práctica para evitar errores posteriores en el código.

```
87        cap = cv2.VideoCapture(video_path)
88        frame_width = int(cap.get(3))
89        frame_height = int(cap.get(4))
```

Ilustración 61 Configuración de captura de video

Aquí, '**cv2.VideoCapture(video_path)**' abre el archivo de video para su lectura, luego, el código obtiene el ancho y el alto de los cuadros del video usando '**cap.get(3)**' y '**cap.get(4)**', respectivamente. Estas líneas aseguran que el video de salida tenga las mismas dimensiones que el video de entrada.

```
90    out = cv2.VideoWriter(os.path.join(output_dir, 'video anotado.avi'), cv2.VideoWriter_fourcc(*'XVI
```

Ilustración 62 Configuración de escritura de video

El código configura un objeto '**cv2.VideoWriter**' para escribir el video procesado. Esto implica definir el nombre del archivo de salida ('**video anotado.avi**'), el codec de video a utilizar ('**XVID'**), la tasa de fotogramas (30.0 fps), y las dimensiones del video (anchura y altura de los cuadros). '**cv2.VideoWriter_fourcc(*'XVID')**' especifica el codec utilizado para la compresión del video.

```
92    while cap.isOpened():
93        ret, frame = cap.read()
94        if not ret:
95            break
```

Ilustración 63 Procesamiento de cuadro por cuadro del video

Este bloque '**while**' procesa el video cuadro por cuadro, '**cap.isOpened()**' comprueba si la captura de video se ha inicializado correctamente y está lista para leer. Dentro del bucle, '**cap.read()**' lee cada cuadro del video. La variable '**ret**' es un valor booleano que indica si el cuadro fue leído correctamente. Si no hay más cuadros para leer ('**ret**' es '**False**'), el bucle se rompe, terminando el procesamiento.

```
96      # Convertir a imagen PIL
97      frame_pil = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
98      annotated_frame = process_frame(frame_pil)
99      out.write(annotated_frame)
```

Ilustración 64 Conversión y Anotación de los fotogramas

Para cada cuadro leído, primero se convierte de un formato de imagen de OpenCV (BGR) a un formato compatible con PIL (RGB) usando '**cv2.cvtColor()**' y '**Image.fromarray()**', esto es necesario si la función '**process_frame**' está diseñada para trabajar con imágenes PIL.

Luego, '**process_frame(frame_pil)**' se utiliza para realizar cualquier procesamiento o anotación en el cuadro, el resultado ('**annotated_frame**') es la versión anotada del cuadro original. Finalmente, '**out.write(annotated_frame)**' escribe el cuadro anotado en el archivo de video de salida.

```
101     cap.release()
102     out.release()
103     print('Procesamiento de video completado.')
```

Ilustración 65 Liberación de recursos

Una vez que se han procesado todos los cuadros del video, se liberan los recursos utilizados. '**cap.release()**' cierra el archivo de video de entrada, y '**out.release()**' cierra el archivo de video de salida. Finalmente, se imprime un mensaje indicando que el procesamiento del video ha sido completado.

```
105     elif input_type == 0:
```

Ilustración 66 Verificación de la entrada para Video en tiempo real

En la línea 105 se verifica si el usuario selecciono '0', lo que significa que quiere hacer la identificación de un video en tiempo real a través de un sistema de red.

```
107     cap = cv2.VideoCapture('http://192.168.0.100:8080/video')
```

Ilustración 67 Captura de video en la red

Aquí, '**cv2.VideoCapture()**' se utiliza para abrir una fuente de video. En este caso, la fuente es una URL que apunta a un flujo de video (streaming) desde una cámara IP o un servidor de video que se ejecuta en una red local. La URL **'http://192.168.0.100:8080/video'** representa la dirección IP de la cámara o el servidor, y el puerto **8080** es el que está sirviendo el video.

```
109    while True:
110        ret, frame = cap.read()
111        if not ret:
112            break
```

Ilustración 68 Bucle para la captura y procesamiento de video

Este bucle '**while**' es el núcleo del procesamiento en tiempo real, '**cap.read()**' lee un cuadro del flujo de video y devuelve dos valores: '**ret**' y '**frame**'.

- '**ret**' es un valor booleano que indica si la lectura del cuadro fue exitosa o no. Si no se puede leer un cuadro (por ejemplo, si el video se detiene o hay un error de red), '**ret**' será '**False**', y el bucle se romperá.
- '**Frame**' es la imagen del cuadro capturado, que se almacenará en memoria para su posterior procesamiento.

```
114    frame_pil = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

Ilustración 69 Conversión de cuadro a imagen PIL

Una vez que se captura un cuadro, este se convierte de un formato de imagen de OpenCV (BGR) a un formato de imagen PIL (RGB) utilizando '**cv2.cvtColor()**', OpenCV usa el formato BGR por defecto, pero muchas bibliotecas de procesamiento de imágenes, como PIL, usan RGB. Esta conversión es necesaria porque la función de procesamiento '**process_frame**' que se usa más adelante está diseñada para trabajar con imágenes en formato PIL.

```
115    annotated_frame = process_frame(frame_pil)
```

Ilustración 70 Procesamiento del Cuadro Capturado

Aquí, la función '**process_frame(frame_pil)**' toma el cuadro convertido como entrada y aplica algún tipo de procesamiento o anotación. En este contexto, el procesamiento podría ser algo como la detección de objetos usando un modelo de YOLO o cualquier otro tipo

de análisis de imagen, el resultado, '**annotated_frame**', es el cuadro procesado con las anotaciones aplicadas.

```
116     cv2.imshow( winname: 'Detección YOLO en tiempo real', annotated_frame)
```

Ilustración 71 Visualización del Cuadro Procesado

Finalmente, el cuadro anotado se muestra en una ventana utilizando '**cv2.imshow()**', el título de la ventana es **'Detección YOLO en tiempo real'**, lo que sugiere que el procesamiento podría estar relacionado con la detección de objetos en tiempo real utilizando el modelo YOLO (You Only Look Once), aunque la función exacta del procesamiento puede variar, '**cv2.imshow()**' abre una ventana de GUI que permite ver el flujo de video procesado a medida que ocurre en tiempo real.

```
119     if cv2.waitKey(1) & 0xFF == ord('q')
120         break
```

Ilustración 72 Salir del bucle de procesamiento de imagen, video y video en tiempo real

El fragmento de código '**cv2.waitKey(1) & 0xFF == ord('q')**' se utiliza para detectar si el usuario presiona la tecla 'q' durante la visualización de una ventana de OpenCV. La función '**cv2.waitKey(1)**' espera 1 milisegundo para una pulsación de tecla, lo que permite al programa continuar casi de inmediato después de verificar la entrada del usuario. Esta operación de espera es prácticamente no bloqueante y es ideal para aplicaciones en tiempo real, como el procesamiento de video.

El operador '**& 0xFF**' se utiliza para asegurar que solo se consideren los últimos 8 bits del valor devuelto por '**cv2.waitKey(1)**', ya que esta función puede devolver un valor de 32 bits en algunas plataformas. La comparación con '**ord('q')**', que devuelve el valor ASCII de **'q'**, verifica si la tecla presionada es efectivamente la letra **'q'**, si se presiona **'q'**, la comparación devuelve '**True**', lo que generalmente se utiliza para terminar un bucle o cerrar una ventana de visualización.

```
122     cap.release()
123     cv2.destroyAllWindows()
124     print('Procesamiento en tiempo real completo.')
```

Ilustración 73 Liberación de Recursos y Cierre de Ventanas

El código '**cap.release()**' cierra la fuente de video previamente abierta con '**cv2.VideoCapture()**', liberando recursos y memoria del sistema para evitar bloqueos,

luego, '**cv2.destroyAllWindows()**' cierra todas las ventanas abiertas por '**cv2.imshow()**', asegurando que no queden ventanas abiertas que puedan consumir recursos innecesariamente. Finalmente, '**print('Procesamiento en tiempo real completo.')**' proporciona retroalimentación al usuario, indicando que el procesamiento en tiempo real ha concluido correctamente y que los recursos han sido liberados con éxito.

```
126     else:
127         print("Tipo de entrada no válido. Por favor, introduzca 1, 2, or 0.")
```

Ilustración 74 Manejo de Entradas No Válidas

La sección '**else**': se ejecuta cuando el usuario proporciona un tipo de entrada que no es ni 1, 2, ni 0, lo cual indica que la entrada no es válida para seleccionar entre procesar una imagen, video o URL, el mensaje '**print("Tipo de entrada no válido. Por favor, introduzca 1, 2, o 0.")**' instruye al usuario a corregir su entrada para elegir un valor válido en la próxima ocasión.

```
129     print('Procesamiento completo.')
```

Ilustración 75 Culminación del programa

En esta línea se da por finalizado el programa y con ello cualquier proceso que se haya ejecutado con el mismos liberando, cualquier recurso usado y aplicado durante el proceso.

3.2.8) Que es NGINX

Nginx (pronunciado "engine-ex") es un software de servidor web de código abierto que también se puede utilizar como proxy inverso, equilibrador de carga, servidor de correo y más. Fue creado por Igor Sysoev en 2004 para resolver un problema de C10k que los servidores web tradicionales como Apache tenían problemas para manejar de manera eficiente (es decir, manejar 10.000 conexiones simultáneas) [70].

Ilustración 76 Herramienta de generación de servicio [71]

Entre sus características más resaltables se encuentran los siguientes:

- **Servidor web:** La función principal de Nginx es servir contenido web, como páginas HTML y archivos multimedia, utilizando el protocolo HTTP. Es conocido por su alto

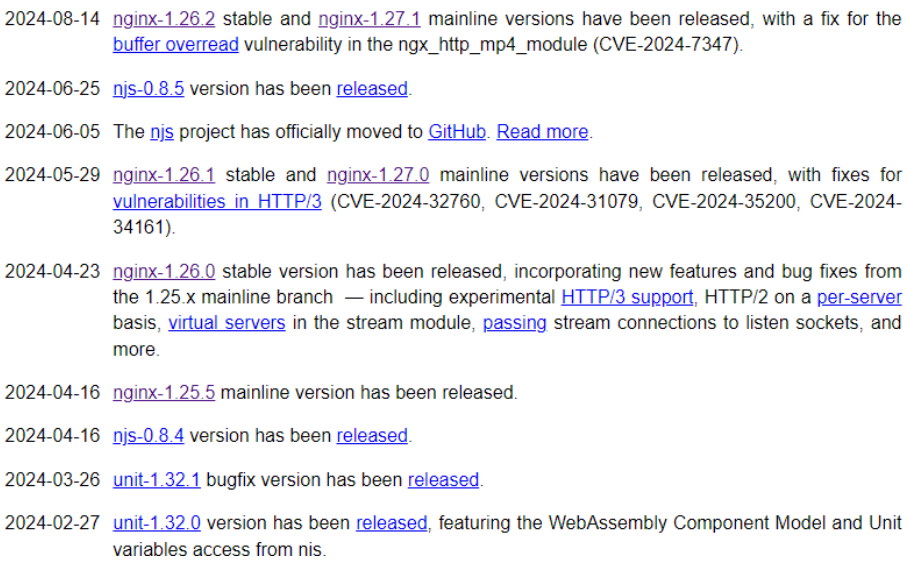
rendimiento, eficiencia y capacidad para manejar múltiples conexiones simultáneas [72].

- **Proxy inverso:** Nginx puede actuar como un proxy inverso, lo que significa que puede recibir solicitudes de los clientes y reenviarlas a uno o más servidores backend (como servidores de aplicaciones o bases de datos). Luego envía la respuesta del backend al cliente, se utiliza para mejorar la seguridad, el equilibrio de carga y mejorar la capacidad de respuesta del sistema [72].
- **Equilibrador de carga:** Nginx puede distribuir el tráfico entrante entre múltiples servidores backend para garantizar que ningún servidor esté sobrecargado, lo que ayuda a mejorar la disponibilidad y el rendimiento del sistema [72].
- **Servidor de correo:** Nginx también se puede configurar como servidor de correo para protocolos como SMTP, IMAP y POP3 además de permitir el procesamiento eficiente y seguro del tráfico de correo electrónico [72].
- **Almacenamiento en caché de contenido:** Nginx puede almacenar en caché contenido estático y dinámico para reducir la carga en los servidores backend y mejorar el tiempo de respuesta del usuario [72].
- **Compatibilidad con HTTP/2 y gRPC:** Nginx admite el protocolo HTTP/2, lo que proporciona conexiones más rápidas y eficientes, y puede manejar solicitudes de gRPC, un sistema RPC de alto rendimiento [72].

3.2.9) Generación de Servicio RTMP

Para generar el protocolo RTMP lo primero que se debe de realizar es la descarga de la herramienta conocida como NGINX, la cual permitirá realizar/generar el servicio de transmisión en tiempo real esto mediante la ejecución en segundo plano, de un servicio de transmisión y recepción de señal, el cual será explicado más adelante por lo mientras hoy se dará una breve explicación de cómo se debe de instalar correctamente la herramienta NGINX.

Para realizarse la correcta instalación de la herramienta NGINX, se debe de ingresar a la página oficial con el mismo nombre, y buscar la versión que se desee instalar, para lo cual en este caso se tomó la versión nginx 1.7.12.1 Lizard, esto puesto que es una de las versiones con una mayor facilidad de no respecto a su configuración, no necesariamente debe ser la misma versión puesto que existen versiones más actuales que permiten la modificación directa de parte de su código para permitir la correcta generación del servicio.



2024-08-14 nginx-1.26.2 stable and nginx-1.27.1 mainline versions have been released, with a fix for the buffer overread vulnerability in the ngx_http_mp4_module (CVE-2024-7347).

2024-06-25 njs-0.8.5 version has been released.

2024-06-05 The njs project has officially moved to GitHub. Read more.

2024-05-29 nginx-1.26.1 stable and nginx-1.27.0 mainline versions have been released, with fixes for vulnerabilities in HTTP/3 (CVE-2024-32760, CVE-2024-31079, CVE-2024-35200, CVE-2024-34161).

2024-04-23 nginx-1.26.0 stable version has been released, incorporating new features and bug fixes from the 1.25.x mainline branch — including experimental HTTP/3 support, HTTP/2 on a per-server basis, virtual servers in the stream module, passing stream connections to listen sockets, and more.

2024-04-16 nginx-1.25.5 mainline version has been released.

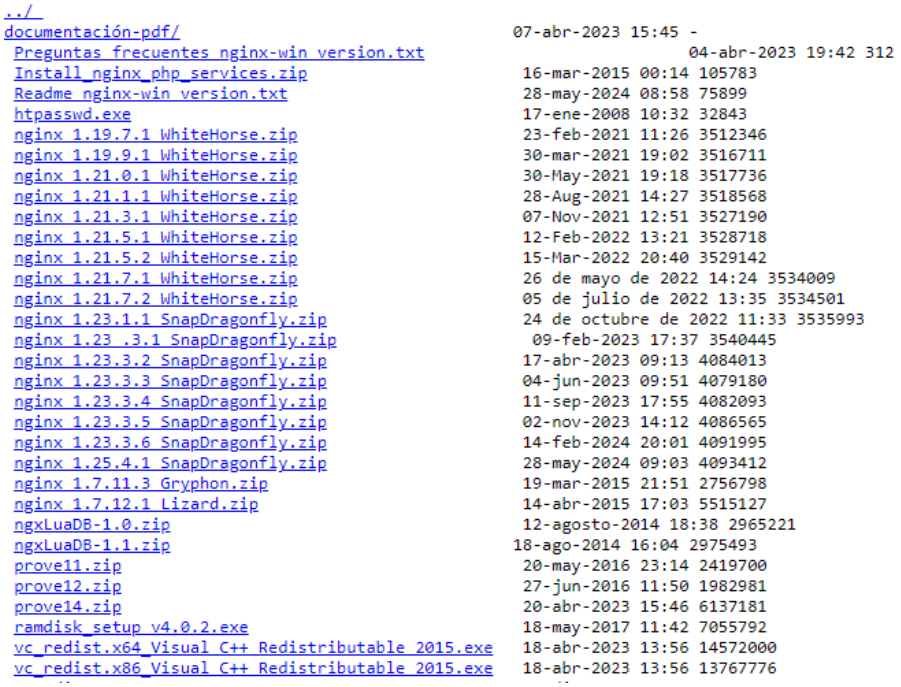
2024-04-16 njs-0.8.4 version has been released.

2024-03-26 unit-1.32.1 bugfix version has been released.

2024-02-27 unit-1.32.0 version has been released, featuring the WebAssembly Component Model and Unit variables access from nis.

Ilustración 77 Versiones más actuales de descarga de nginx [73]

La página mostrada no es necesariamente la principal para las descargas de las versiones puesto que dentro de la misma página web oficial existe un apartado que permite verificar mejor las versiones en la cual se puede encontrar la versión que se está utilizando para el proyecto.



../	
documentación-pdf/	07-abr-2023 15:45 -
Preguntas frecuentes nginx-win version.txt	04-abr-2023 19:42 312
Install_nginx_php_services.zip	16-mar-2015 00:14 105783
Readme nginx-win version.txt	28-may-2024 08:58 75899
htpasswd.exe	17-ene-2008 10:32 32843
nginx 1.19.7.1 WhiteHorse.zip	23-feb-2021 11:26 3512346
nginx 1.19.9.1 WhiteHorse.zip	30-mar-2021 19:02 3516711
nginx 1.21.0.1 WhiteHorse.zip	30-May-2021 19:18 3517736
nginx 1.21.1.1 WhiteHorse.zip	28-Aug-2021 14:27 3518568
nginx 1.21.3.1 WhiteHorse.zip	07-Nov-2021 12:51 3527190
nginx 1.21.5.1 WhiteHorse.zip	12-Feb-2022 13:21 3528718
nginx 1.21.5.2 WhiteHorse.zip	15-Mar-2022 20:40 3529142
nginx 1.21.7.1 WhiteHorse.zip	26 de mayo de 2022 14:24 3534009
nginx 1.21.7.2 WhiteHorse.zip	05 de julio de 2022 13:35 3534501
nginx 1.23.1.1 SnapDragonfly.zip	24 de octubre de 2022 11:33 3535993
nginx 1.23 .3.1 SnapDragonfly.zip	09-feb-2023 17:37 3540445
nginx 1.23.3.2 SnapDragonfly.zip	17-abr-2023 09:13 4084013
nginx 1.23.3.3 SnapDragonfly.zip	04-jun-2023 09:51 4079180
nginx 1.23.3.4 SnapDragonfly.zip	11-sep-2023 17:55 4082093
nginx 1.23.3.5 SnapDragonfly.zip	02-nov-2023 14:12 4086565
nginx 1.23.3.6 SnapDragonfly.zip	14-feb-2024 20:01 4091995
nginx 1.25.4.1 SnapDragonfly.zip	28-may-2024 09:03 4093412
nginx 1.7.11.3 Gryphon.zip	19-mar-2015 21:51 2756798
nginx 1.7.12.1 Lizard.zip	14-abr-2015 17:03 5515127
ngxLuaDB-1.0.zip	12-agosto-2014 18:38 2965221
ngxLuaDB-1.1.zip	18-ago-2014 16:04 2975493
prove11.zip	20-may-2016 23:14 2419700
prove12.zip	27-jun-2016 11:50 1982981
prove14.zip	20-abr-2023 15:46 6137181
ramdisk_setup_v4.0.2.exe	18-may-2017 11:42 7055792
vc_redist.x64_Visual C++ Redistributable 2015.exe	18-abr-2023 13:56 14572000
vc_redist.x86_Visual C++ Redistributable 2015.exe	18-abr-2023 13:56 13767776

Ilustración 78 Versiones de descarga de nginx [73]

Ambas versiones de ambas páginas oficiales de lo que viene a ser la herramienta, serán dejadas a continuación para futuras referencias i o utilizaciones en otros proyectos.

- https://nginx.org/
- http://nginx-win.ecsds.eu/download/

Una vez completada la descarga del programa, es necesario proceder con la modificación directa de los archivos de configuración, esto se debe a que la herramienta no se instala de manera convencional en el disco duro, sino que se ejecuta directamente como un archivo ejecutable, por lo tanto, es imprescindible acceder a la ruta en la que se descargó o registró el programa para realizar los ajustes correspondientes, pero de forma general siendo algo como lo que se puede ver a continuación.

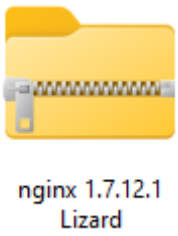


Ilustración 79 Descompresión del archivo nginx

Como lo indica la ilustración 79 lo primero que se debe de realizar es la descompresión del archivo para su posterior, modificación.

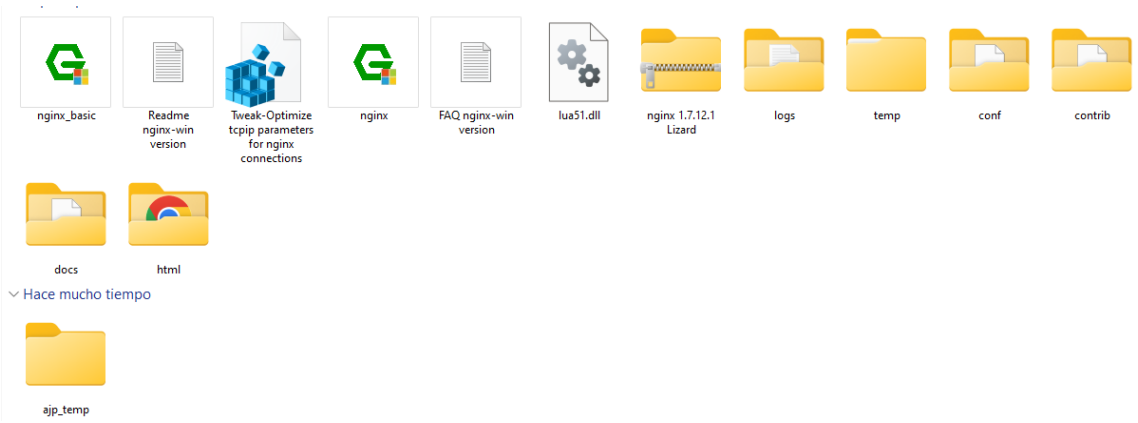


Ilustración 80 Identificación de la carpeta a modificar

Después de haberse descomprimido el archivo, hoy se debe de abrir y buscar la carpeta que diga con y dentro de esta buscar el siguiente documento.

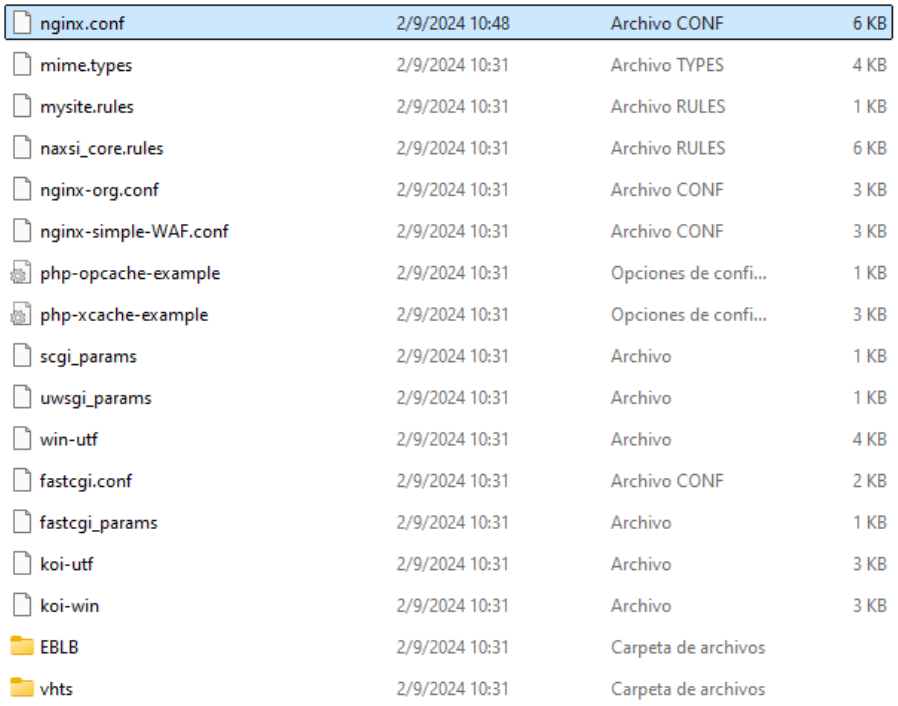


Ilustración 81 Modificación del archivo generador de servicio

El archivo nginx.conf, es el archivo al cual se le va a configurar o más bien alterar una parte de la línea de código interno, esto ya que, mediante esta nueva línea, se le permitirá al programa poder ejecutar un servicio de transmisión en tiempo real.

```
    # HTTPS server
    #
    #server {
    #    listen       443 ssl spdy;
    #    server_name  localhost;

    #    ssl                  on;
    #    ssl_certificate      cert.pem;
    #    ssl_certificate_key  cert.key;

    #    ssl_session_timeout  5m;

    #    ssl_prefer_server_ciphers On;
    #    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    #    ssl_ciphers ECDH+AESGCM:ECDH+AES256:ECDH+AES128:ECDH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!eNULL:!MD5:!DSS:!EXP:!ADH:!LOW:!MEDIUM;

    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #}

}
rtmp {
        server {
                listen 1935;
                chunk_size 20480;

                application live {
                        live on;
                        record off;
                }
        }
}
```

Ilustración 82 Colocación de nueva línea de ejecución

Dentro de lo que viene hacer el archivo, al final de todo se debe de generar la siguiente línea de código, que permitirá la apertura de un servicio de transmisión en tiempo real.

```
}
rtmp {
        server {
                listen 1935;
                chunk_size 20480;

                application live {
                        live on;
                        record off;
                }
        }
}
```

Ilustración 83 Código de apertura de transmisión

Después de agregar este fragmento de código al final del perfil, guarde los cambios y continúe ejecutando el servicio de transmisión. Esto permitirá activar el sistema, permitiendo la transmisión y recepción de datos. En este caso, el dispositivo actúa como intermediario, facilitando la comunicación entre el emisor y el receptor de la señal. De esta forma, los servicios de transporte se establecen de forma inmediata y se garantiza la correcta coordinación de las señales.

Ya después de haberse modificado todo lo anteriormente dicho de forma correspondiente sólo quedaría ejecutar el archivo de NGINX, lo cual haría que como tal el programa comience con la transmisión y recepciones de toda la información que posea el identificador IP y la contraseña designada por el usuario.

CAPITULO IV

4.1) Resultados

4.1.1) Resultados de Conexión

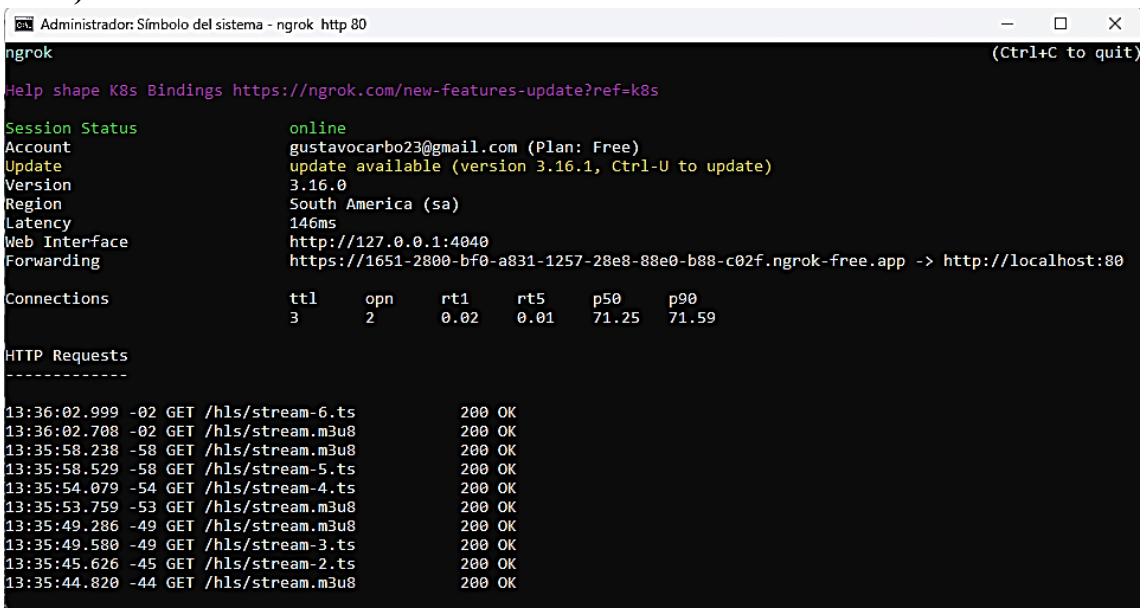


Ilustración 84 Proceso de transmisión entre dispositivos

Para verificar el correcto traslado de información entre los dispositivos, se utiliza la herramienta conocida como Ngrok, la cual cumple con dos propósitos principales. En primer lugar, genera un certificado de funcionalidad de la transmisión; en segundo lugar, actúa como un sistema visualizador que permite monitorear el flujo de datos, un ejemplo de esto se muestra en la ilustración 84, donde se puede observar en la parte inferior de la pantalla, justo después del texto "HTTP Requests", las líneas de código que siguen registran el traslado de los datos, mientras que, a la derecha, donde aparece "200 OK", se indica la cantidad de paquetes transferidos correctamente, en caso de fallos en la transmisión, se mostraría un mensaje de error, como se ve, en la siguiente ilustracion.

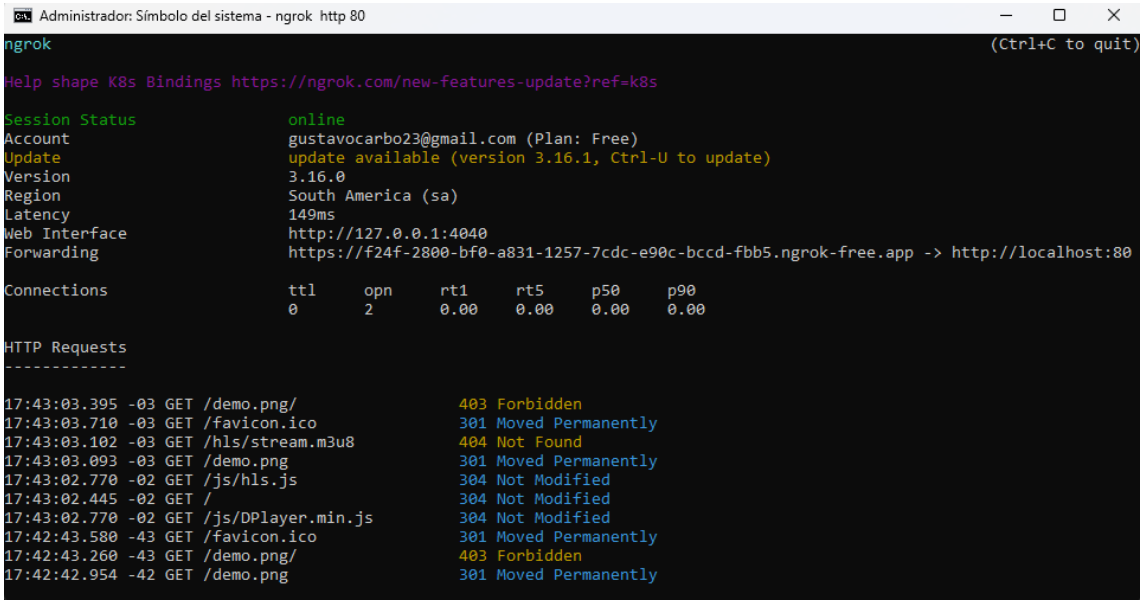


Ilustración 85 Error de transmisión

En la ilustración 85, como se mencionó previamente, en caso de pérdida de conexión o interrupción del enlace por donde se moviliza la información, el CMD mostrará mensajes de error indicando la desconexión, además, se señalará la interrupción en el traslado de los paquetes, lo que obligará al administrador del sistema a restablecer la red. Esto implicará detener el proceso de transferencia y reiniciar todos los sistemas conectados a la red propietaria, donde se generan los paquetes que deben ser enviados.

4.1.2) Resultados Prácticos de las capturas

Ilustración 86 Detección de Pulgón en sistema externo

En la ilustración 86 se muestra un ejemplo del resultado final del programa de detección de pulgón mediante visión por computadora, utilizando el protocolo RTMP para la transmisión de información hacia el equipo de análisis, el resultado es la imagen presentada, donde, con el apoyo de YOLOv5, el programa es capaz de detectar el pulgón de manera rápida y precisa en las imágenes enviadas por el UAV.

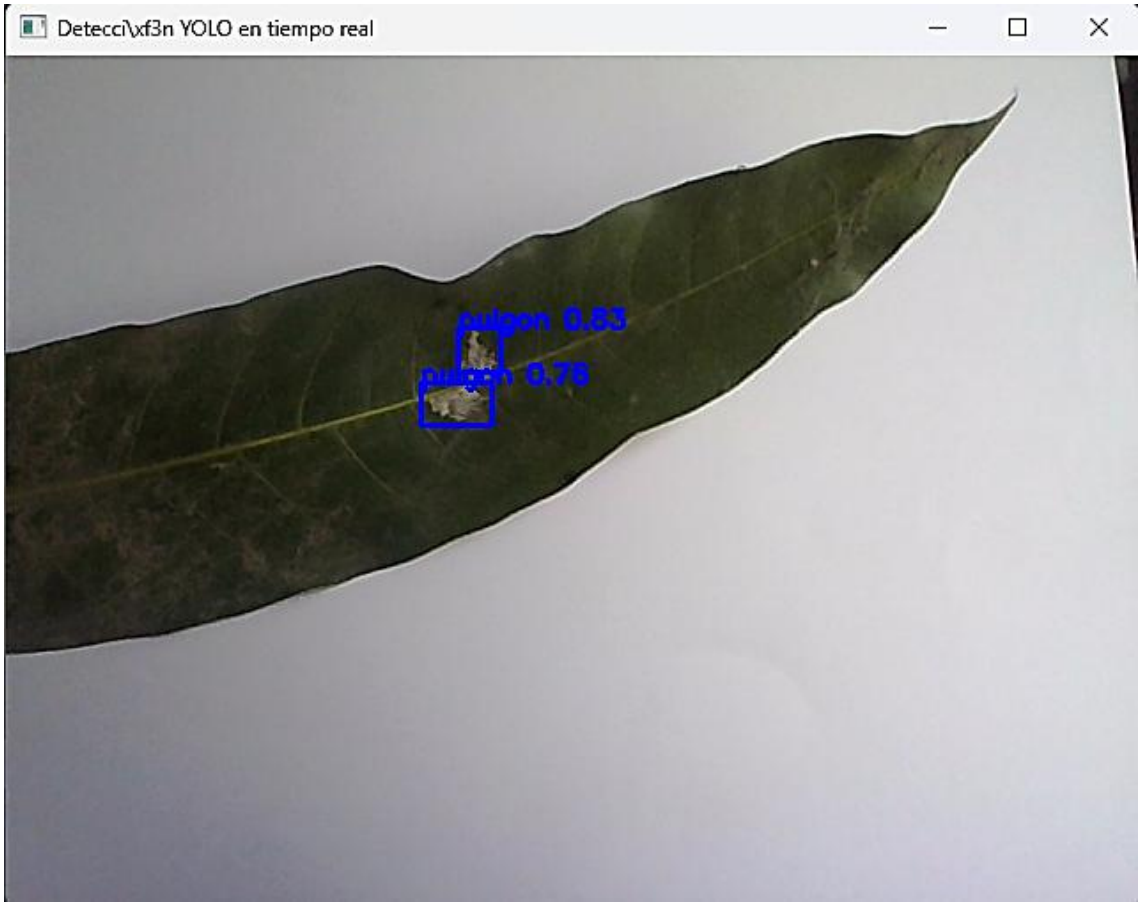


Ilustración 87 Detección de Pulgón en sistema interno

En la ilustración 87 se muestra cómo se probó el sistema de detección de pulgón en un entorno controlado e interno. Como se puede observar, el resultado fue exitoso, ya que el programa detectó únicamente la presencia del pulgón en la zona designada, sin identificar ningún otro elemento en la imagen.

4.1.3) Graficas Resultantes del entrenamiento

Después de haberse realizado el entrenamiento y generado el sistema de transmisión y recepción, lo que queda es verificada la precisión del sistema, es por ello que a continuación se presentará en los gráficos resultantes del entrenamiento y con ello una breve explicación del significado de cada uno de estos para de esta forma tener una mejor idea de la calidad, del programa diseñado.

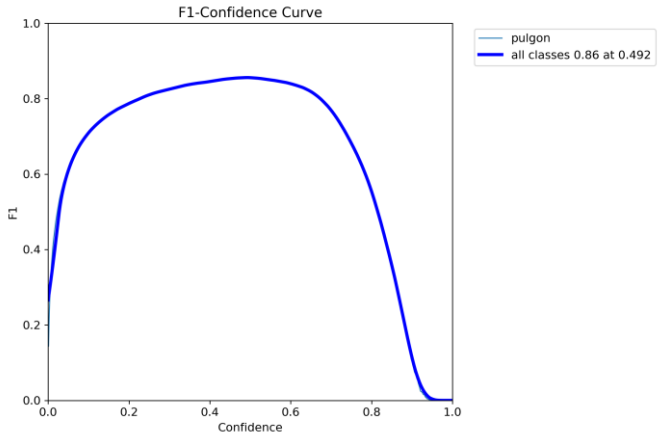


Ilustración 88 Curva F1-Confidence

Como se puede ver en la ilustración 84 se puede ver el modelo de aprendizaje donde en el eje Y, representa la métrica armónica de la precisión y el recall, estas métricas, se encargan de equilibrar el modelo en caso de un falso positivo o negativo, mientras que en el eje X, se puede ver un inicio bajo, lo que indica una falsa detección, que se va arreglando a medida que el proceso continuo, hasta la finalización del programa donde existe una caída progresiva, debido a lo ya mencionado, es decir la culminación del entrenamiento.

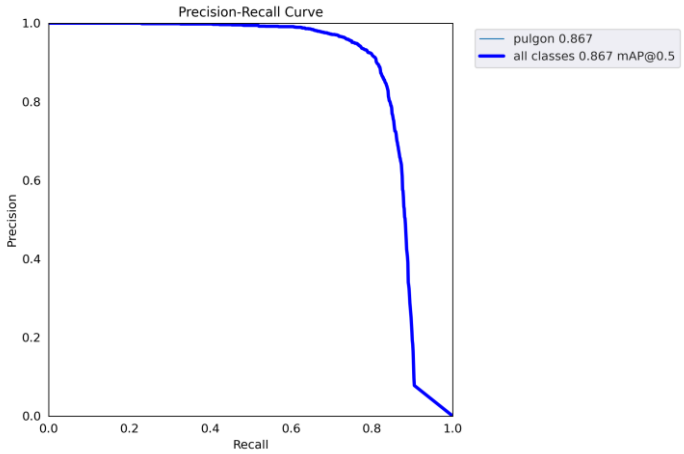


Ilustración 89 Curva Precision-Recall (PR)

En la ilustración 85 se puede ver como la confianza del modelo inicia desde un punto alto, en el eje Y, mientras que en el eje X recall, muestra como la capacidad del modelo para identificar correctamente la presencia de lo indicado(proporción de verdaderos positivos entre todas las observaciones positivas reales), al finalizar el programa, nuevamente se puede ver una caída abrupta a partir de 0.9 lo que al igual que la gráfica anterior indica, no solo la finalización del programa y del entrenamiento sino también que como tal, el

identificador se vuelve menos selectivo y por ende posee menos coherencia en la detección de objetos.

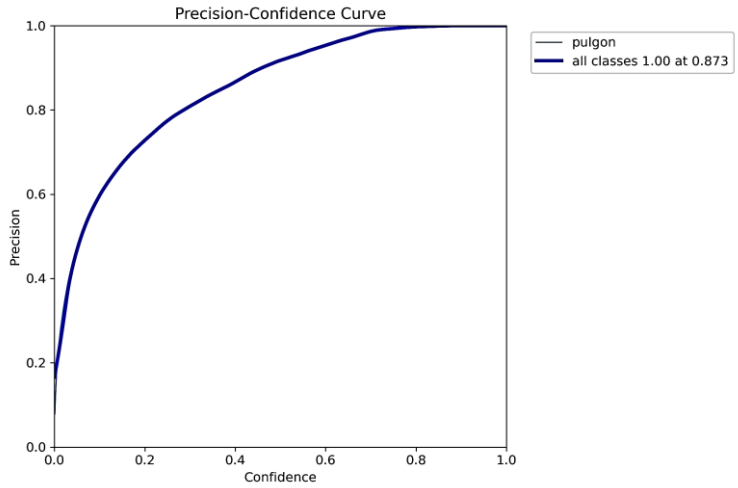


Ilustración 90 Curva de Precisión-Confianza

En la ilustración 86 se puede ver además de los ejes X y Y, que representan la confidencia y la precisión respectivamente, una curvatura que empieza desde abajo la cual indica un bajo nivel de confianza, la cual va aumentado de forma gradual hasta llegar a su punto máximo del 100% siendo este un aproximado de 0.873, lo que significa que el modelo tiene una muy reducido índice de error por falsas detecciones.

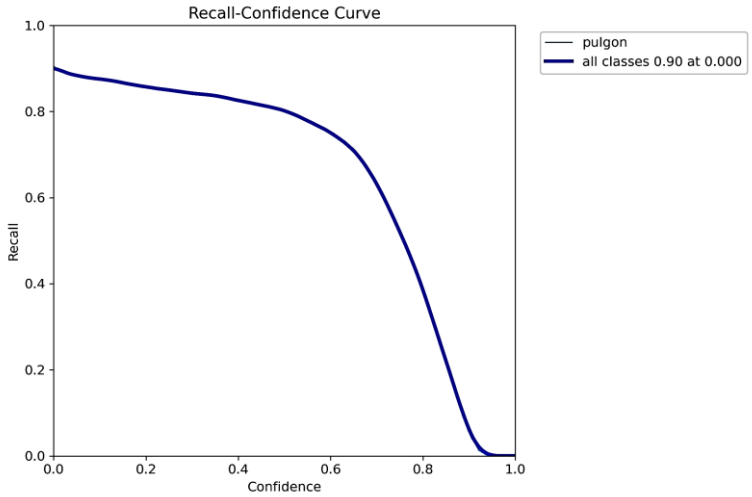


Ilustración 91 Curva de Recall-Confianza

En la ilustración 87, se puede ver el nivel de *recall*, que a pesar de que termina siendo cero, comienza con un índice de detección alto, junto a la confianza, donde se puede determinar que entre mayor sea el índice del recall el índice de predicción se reduce y

viceversa, por lo que se puede concluir que el *recall* y la confianza del modelo son inversamente proporcionales, ya que como lo ya mencionado en un índice de confianza alto, el modelo se vuelve más estricto con la detección, teniendo menos detecciones, pero si ocurre lo contrario, habría más detecciones, siendo estas falsos positivos.

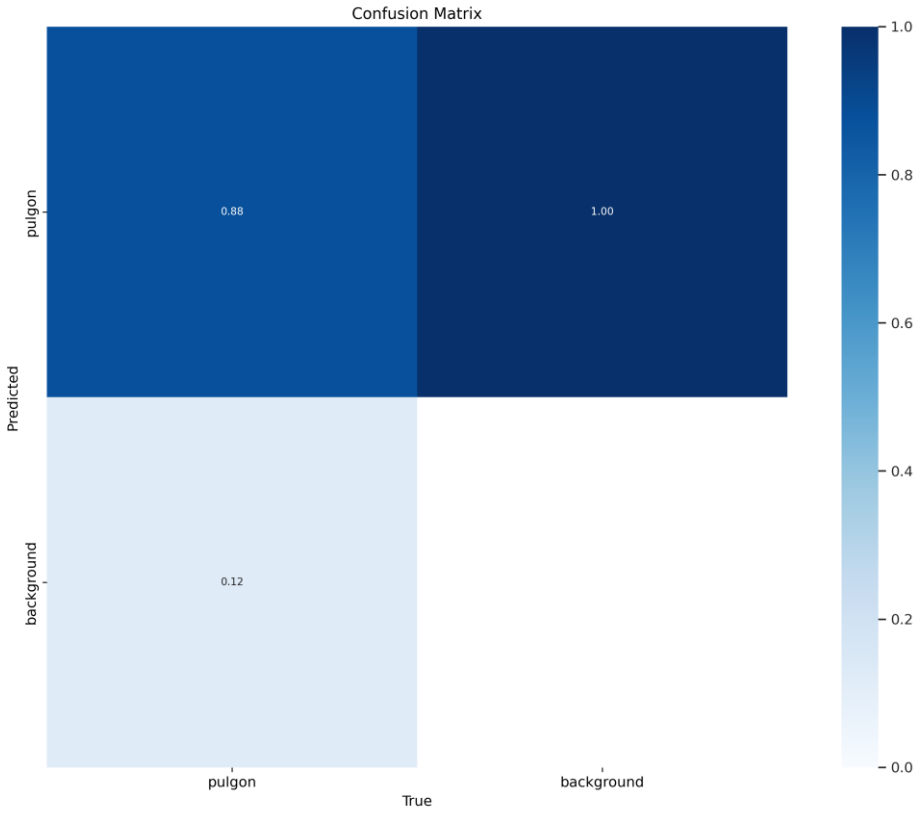


Ilustración 92 Matriz de Confusión

En la ilustración 88 se observa la finalización del entrenamiento del modelo YOLOv5, donde se evalúan los aciertos y errores en la detección del objetivo, los resultados muestran una precisión del 0.88, lo que indica que el modelo detectó correctamente el 88% de los casos analizados, por otro lado, se registró un 0.12 de errores, atribuibles a falsos positivos, esto sugiere que, en un 12% de los casos, el modelo identificó erróneamente el objetivo, lo cual podría estar relacionado con dificultades del modelo para detectar pulgones en ciertos fondos durante el proceso de entrenamiento, en resumen, el análisis muestra un desempeño general del 100%, con una tasa de aciertos del 88% y una tasa de error del 12%.

4.1.4) Graficas del modelo final del entrenamiento

Después de haberse explicado los resultados de cada uno de los modelos de entrenamiento, a continuación, se mostrara de forma completa el resultado final, del algoritmo y la precisión de este, explicando a su vez el significado de cada una de las grafías, a la vez que se da un breve resumen del porque son llamadas así y de su importancia en el análisis del correcto funcionamiento del modelo de detección basado en YOLOv5.

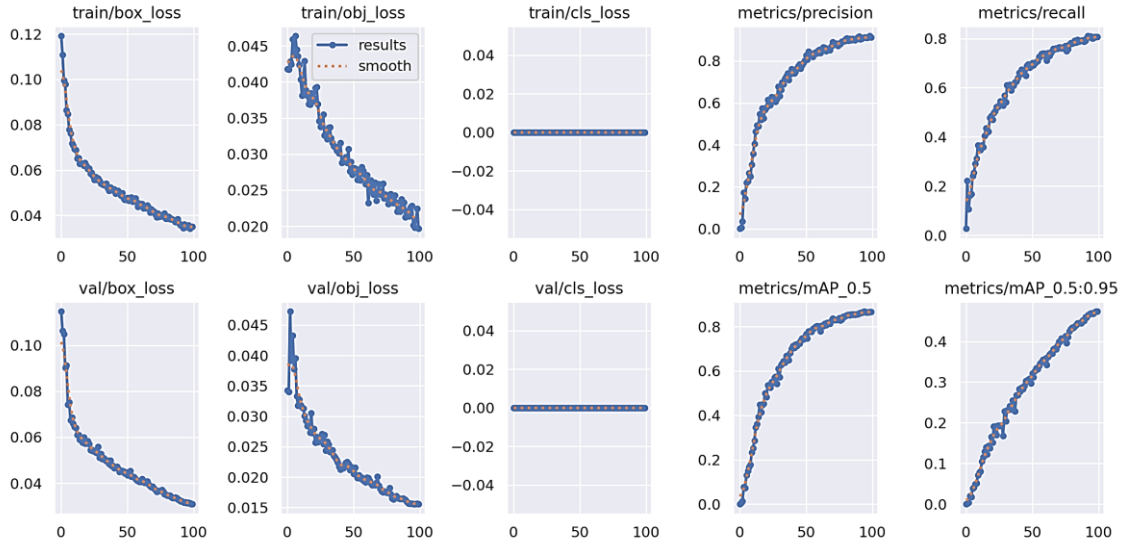


Ilustración 93 Evolución de las pérdidas durante el entrenamiento de YOLOv5

1. **Train/Box_loss:** En la primera grafica de la izquierda superior, se muestran las pérdidas de la caja de predicción durante su modelo, lo cual indica que a medida que avanza el entrenamiento el modelo mejora su detección.
2. **Train/obj_loss:** Las pérdidas relacionadas con la detección de objetos también disminuyen gradualmente, lo que refleja que el modelo aprende a detectar con mayor precisión la presencia de objetos en las imágenes.
3. **Train/cls_loss:** En la gráfica de, en medio se puede ver una línea recta constante en 0, la cual es debido a que como tal el modelo solo se dedica a la detección de un solo objeto, puesto que la tarea que se requiere solo hace uso de una sola clase la cual es Pulgón.
4. **Metric/precisión:** Esta grafica muestra como a medida que el entrenamiento avanza el tiempo, el modelo tiene una mayor capacidad de detección, volviéndose en temas simples un sistema más rápido y apto, para la detección.

5. **Metric/recall:** La grafica Metric, como el modelo mejora su aprendizaje para la detección de objetos, siendo este mismo el pulgón, pero con lo mostrado con una alta eficiencia.
6. **Val/box_loss:** De manera similar a la pérdida de caja durante el entrenamiento, la pérdida de validación para la localización de objetos también se reduce, lo que indica que el modelo se generaliza bien al material invisible.
7. **Val/obj_loss:** La pérdida de detección de objetos en la validación es similar a la del entrenamiento, lo que indica que el rendimiento general del modelo es bueno.
8. **Val/cls_loss:** Al igual que la gráfica 3, en esta no se ve que haya un valor, siendo este 0, lo que indica al igual que la gráfica anterior, que no hay más de una detección en el modelo de entrenamiento.
9. **Metrics/mAP_0.5**: El valor de precisión media (mAP) en un umbral de 0,5 también mejora durante el entrenamiento, lo que indica una mejora en la capacidad del modelo para detectar objetos con una precisión razonable.
10. **Metrics/mAP_0.5:0.95**: Finalmente, el mAP bajo el rango de umbral de 0,5 a 0,95 también muestra una tendencia creciente, lo que significa que el modelo es capaz de realizar una detección precisa incluso en condiciones de evaluación más estrictas.

4.1.5) Conclusiones

- En los sistemas agrícolas, se utilizan comúnmente técnicas de reconocimiento como la segmentación de imágenes (Image Segmentation), el reconocimiento de objetos (Object Recognition) y la detección de características (Feature Detection), ya que permiten obtener resultados más precisos. De estas técnicas, la más empleada suele ser el reconocimiento de objetos, debido a que YOLO ofrece un entorno amigable para el usuario y es fácilmente interpretable a nivel visual. Por otro lado, en la transmisión en tiempo real, los principales protocolos utilizados son RTMP, RTP, RTSP, SRT y FLV, cada uno con tiempos de latencia relativamente bajos. Por ejemplo, el RTMP presenta una latencia de 2 a 5 segundos, mientras que el RTP puede tener entre 20 y 150 milisegundos. El RTSP presenta latencias de 1 a 3 segundos, el SRT varía entre 100 milisegundos y 1 segundo, y el FLV puede llegar a 2 a 5 segundos de latencia. Aunque el RTMP es el protocolo más antiguo, sigue siendo uno de los más utilizados hoy

en día debido a su facilidad de uso y adaptabilidad a diversos entornos de transmisión en tiempo real.

- Para la generación de la base de datos, existen varias herramientas, como **LabelImg**, **VGG Image Annotator**, **Makesense.AI** y **Roboflow**, estas plataformas se utilizan para crear una base sólida sobre la cual se llevará a cabo el entrenamiento del modelo, en este caso, se utilizó Roboflow, con aproximadamente 300 muestras, cada una con una resolución de
  píxeles, aunque habitualmente se podrían utilizar más muestras para obtener resultados más precisos, en este proyecto se optó por trabajar con imágenes de alta calidad, lo que permite al sistema identificar la plaga (pulgón) de manera más eficiente.
- Para el entrenamiento, se utilizó el modelo **YOLOv5**, en su versión **x**, debido a su alta eficiencia y bajos requerimientos de recursos, tanto durante el entrenamiento como en su implementación dentro del programa, las especificaciones empleadas en el modelo son cruciales, ya que parámetros incorrectos generarían resultados gráficos erróneos, en este caso, se utilizaron un batch size de 8 y 100 epochs, lo que resultó en un 88% de precisión y un 12% de errores, al probar un entrenamiento con 80 epochs y un batch size de 16, los resultados fueron significativamente peores, con más del 50% de errores y apenas un 40% de precisión, esto llevó a la conclusión de que un ciclo de entrenamiento extenso, combinado con un tamaño de lote de 8, proporciona los mejores resultados posibles para la cantidad de muestras disponibles. Modificar estos parámetros, ya sea aumentando o disminuyendo, resultaría en una reducción de la calidad en el conjunto de datos final.
- Tras realizar todas las pruebas, se obtuvieron las siguientes conclusiones, la latencia de transmisión a distancias de 3 a 4 metros varía entre 135 y 153 milisegundos, afectando únicamente la transmisión del UAV al dispositivo receptor, sin embargo, la transmisión del receptor a la computadora para su procesamiento presenta una latencia de 3 a 5 segundos, la cual puede aumentar a un máximo de 7 u 8 segundos cuando la distancia alcanza los 60 metros, la detección de plagas mediante videos e imágenes procesados es 100% efectiva. El procesamiento en tiempo real se ve afectado por la carga adicional de mantener la conexión entre el dron y el sistema de visión por computadora, generando un retraso total de 7 a 8 segundos con una variación adicional de 50

a 70 milisegundos durante la simulación, no obstante, el programa sigue siendo efectivo en la detección de plagas, específicamente del pulgón.

4.1.6) Recomendaciones

- El equipo requerido para el proyecto debe ser de gama media alta o alta para garantizar que pueda realizar los procesos requeridos de manera fluida y sin demoras excesivas o notorias.
- Es recomendable que el equipo posea una tarjeta gráfica, la cual se encargue del procesamiento de imágenes mientras que la CPU, se encarga del mantenimiento de los múltiples procesos.
- Sea lo más específico posible al etiquetar imágenes, el etiquetado incorrecto en algunas muestras puede afectar negativamente los resultados del entrenamiento, reduciendo la precisión del modelo.
- Cuando se recopilan muestras para crear una biblioteca, deben tener una resolución mínima de 1280 x 720 para garantizar una calidad suficiente para obtener una muestra clara y representativa de lo que se debe determinar.
- Después de configurar la red y comenzar la transmisión, la computadora emisora debe usar la misma dirección IP que la computadora que viene de la red. Esto es necesario para garantizar que el tráfico se enrute correctamente a su destino previsto.

[1] M. de C. H. CRUZ and M. de C. H. CRUZ, “LA VISIÓN POR COMPUTADORA Y LAS FUTURAS APLICACIONES TECNOLÓGICAS EN DIVERSOS ESCENARIOS,” *Revista de la Academia del Guerra del Ejército Ecuatoriano*, vol. 12, no. 1, p. 5, Jul. 2021, doi: 10.24133/age.n12.2019.13.

[2] “El futuro del campo mexicano en la agricultura digital - Reporte Indigo.” Accessed: Nov. 13, 2023. [Online]. Available: https://www.reporteindigo.com/reporte/el-futuro-del-campo-mexicano-en-la-agricultura-digital/

[3] “Alojamiento de servidores RTMP: Qué es y cómo acceder a él.” Accessed: Oct. 16, 2023. [Online]. Available: https://www.dacast.com/es/blog-es/rtmp-servidor-hosting/

[4] “Así es la historia de la Visión por ordenador - LISA Insurtech.” Accessed: Oct. 09, 2023. [Online]. Available: https://lisainsurtech.com/asi-es-la-historia-de-la-vision-por-ordenador/

[5] “Investigación exploratoria: tipos, metodología y ejemplos.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.lifeder.com/investigacion-exploratoria/

[6] “Investigación descriptiva: características, técnicas, ejemplos.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.lifeder.com/investigacion-descriptiva/

[7] “Los enemigos naturales de los pulgones. Servicio Regional de Investigación y Desarrollo Agroalimentario.” Accessed: Dec. 11, 2023. [Online]. Available: http://www.serida.org/publicacionesdetalle.php?id=4811

[8] “Los pulgones - Las plagas - cyclamen.com - Enfermedades.” Accessed: Oct. 07, 2023. [Online]. Available: https://www.cyclamen.com/es/profesional/enfermedades/1/10

[9] “¿Qué es RTMP? .” Accessed: Dec. 17, 2023. [Online]. Available: https://www.dacast.com/es/blog-es/rtmp-real-time-messaging-protocol/

[10] “Protocolo de transmisión en tiempo real .” Accessed: Dec. 17, 2023. [Online]. Available: https://es.wikipedia.org/wiki/Protocolo_de_transmisi%C3%B3n_en_tiempo_real

[11] “¿Qué son y para qué sirven los drones? - UAVLatam.” Accessed: Dec. 20, 2023. [Online]. Available: https://uavlatam.com/que-son-los-drones-para-que-sirven/

[12] “Visión por Computador Qué es, Aplicaciones y Objetivos.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.edsrobotics.com/blog/vision-computador-que-es/

[13] “¿Qué es la visión por computadora? Sistemas de visión artificial y aplicaciones: ejemplos de soluciones — RecFaces.” Accessed: Dec. 17, 2023. [Online]. Available: https://recfaces.com/es/articles/vision-computador-soluciones

[14] “Visión por computadora.” [Online]. Available: http://www.cs.ucf.edu/courses/cap6411/book.pdf

[15] “Introducción a la visión por computadora: qué es y cómo funciona - Brita Inteligencia Artificial.” Accessed: Dec. 17, 2023. [Online]. Available: https://brita.mx/introduccion-a-la-vision-por-computadora-que-es-y-como-funciona/

[16] “HOG (Histogram of Oriented Gradients): An Overview | by Mrinal Tyagi | Towards Data Science.” Accessed: Dec. 16, 2023. [Online]. Available: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f

[17] V. Hugo and E. Zendrero, “INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY Adaboost bayesiano para detección de rostros.”

[18] “YOLO: Real-Time Object Detection.” Accessed: Dec. 16, 2023. [Online]. Available: https://pjreddie.com/darknet/yolo/

[19] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, Dec. 2015, doi: 10.1007/978-3-319-46448-0_2.

[20] “Region Based Convolutional Neural Networks.” Accessed: Dec. 16, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Region_Based_Convolutional_Neural_Networks

[21] “¿Qué es TensorFlow y para qué sirve?” Accessed: Dec. 16, 2023. [Online]. Available: https://www.incentro.com/es-ES/blog/que-es-tensorflow

[22] “Aprende sobre PyTorch con cursos online | edX.” Accessed: Dec. 16, 2023. [Online]. Available: https://www.ibm.com/mx-es/topics/pytorch

[23] “Keras Documentation,” *keras.io*, Accessed: Dec. 16, 2023. [Online]. Available: https://keras.io/#why-this-name-keras

[24] “Some Amazing Applications of OpenCV Library - Analytics Vidhya.” Accessed: Dec. 16, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/09/some-amazing-applications-of-opencv-library/

[25] S. Van Der Walt *et al.*, “Scikit-image: Image processing in python,” *PeerJ*, vol. 2014, no. 1, 2014, doi: 10.7717/PEERJ.453.

[26] “Pillow (PIL Fork) 10.1.0 documentation.” Accessed: Dec. 16, 2023. [Online]. Available: https://pillow.readthedocs.io/en/stable/

[27] “SimpleCV.” Accessed: Dec. 16, 2023. [Online]. Available: http://simplecv.org/

[28] “Mahotas: Open source software for scriptable computer vision,” *J Open Res Softw*, vol. 1, no. 1, p. e3, Jul. 2013, doi: 10.5334/JORS.AC.

[29] “What is OpenCV? The Complete Guide (2024) - viso.ai.” Accessed: Dec. 16, 2023. [Online]. Available: https://viso.ai/computer-vision/opencv/

[30] “¿Qué es el RTMP? Una visión general del protocolo - IONOS.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/rtmp/

[31] “Banano, Origen e Influencia en la Economía Ecuatoriana – GoldGreen.” Accessed: Dec. 17, 2023. [Online]. Available: https://goldgreen.com.ec/banano-origen/

[32] “8 tipos de pulgones.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.jardineriaon.com/tipos-de-pulgones.html

[33] “El impacto del Cambio Climático en los pulgones y en el uso de pesticidas - FuturCrop.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.futurcrop.com/el-impacto-del-cambio-climatico-en-los-pulgones-y-en-el-uso-de-pesticidas/

[34] “▷ ✓ 10 formas efectivas para eliminar los pulgones en planta ◁- Plantasmanía 🌱 🌱.” Accessed: Dec. 11, 2023. [Online]. Available: https://www.plantasmania.com/pulgones-amarillos/

[35] “Pulgones en trigo: recomendaciones para el monitoreo - Agrolatam.” Accessed: Dec. 11, 2023. [Online]. Available: https://www.agrolatam.com/nota/pulgones-en-trigo-recomendaciones-para-el-monitoreo/

[36] “Drones en la agricultura: conoce qué son y para qué sirven.” Accessed: Dec. 11, 2023. [Online]. Available: https://agrotendencia.tv/agropedia/cultivos/drones-en-la-agricultura-tipos-que-son-para-que-sirven/

[37] “VISION POR COMPUTADOR - JAVIER GONZALEZ JIMENEZ - 9788428326308.” Accessed: Dec. 11, 2023. [Online]. Available: https://www.agapea.com/libros/VISIoN-POR-COMPUTADOR-9788428326308-i.htm

[38] “ADN ambiental: La genética al servicio de la protección de los humedales y su biodiversidad – Noticias UACh.” Accessed: Dec. 12, 2023. [Online]. Available: https://diario.uach.cl/adn-ambiental-la-genetica-al-servicio-de-la-proteccion-de-los-humedales-y-su-biodiversidad/

[39] B. M. J. Farfán, “BIOLOGÍA MOLECULAR APLICADA AL DIAGNÓSTICO CLÍNICO,” *Revista Médica Clínica Las Condes*, vol. 26, no. 6, pp. 788–793, Nov. 2015, doi: 10.1016/J.RMCLC.2015.11.007.

[40] “Agrobio – Control biológico de plagas y polinización natural Pulgón - Control biológico de áfidos | Agrobío.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.agrobio.es/informacion/plagas/pulgon/

[41] “Aplicación de la visión por computadora en la agricultura y la agricultura de precisión.” Accessed: Nov. 11, 2023. [Online]. Available: https://ichi.pro/es/aplicacion-de-la-vision-por-computadora-en-la-agricultura-y-la-agricultura-de-precision-133798432692829

[42] “Aplicación de la visión por computadora en la agricultura y la agricultura de precisión.” Accessed: Dec. 13, 2023. [Online]. Available: https://ichi.pro/es/aplicacion-de-la-vision-por-computadora-en-la-agricultura-y-la-agricultura-de-precision-133798432692829

[43] J. Gómez-Camperos, H. Jaramillo, and G. Guerrero-Gómez, “Técnicas de procesamiento digital de imágenes para detección de plagas y enfermedades en cultivos: una revisión,” *Ingeniería y Competitividad*, vol. 24, no. 1, Oct. 2021, doi: 10.25100/iyc.v24i1.10973.

[44] “▷ Tipos de drones : Clasificación, nombres y Usos 2021 - Novodrone.” Accessed: Mar. 21, 2024. [Online]. Available: https://novodrone.com/content/13-tipos-de-drones

[45] “Todo lo que necesitas saber sobre los drones de carreras.” Accessed: Mar. 21, 2024. [Online]. Available: https://umilesgroup.com/drones-de-carreras/

[46] “▷ ¿Cuánto tiempo puede volar un dron? Planeta Tecnológico de los Drones.” Accessed: Mar. 21, 2024. [Online]. Available: https://drones.comoescoger.com/cuanto-tiempo-puede-volar-un-dron/

[47] “14 drones con los mejores tiempos de vuelo [grado profesional y hobby],” *https://volandocondrones.com/*, Accessed: Mar. 21, 2024. [Online]. Available: https://volandocondrones.com/mejores-tiempos-de-vuelo/

[48] “Todas las Partes de los Drones. Explicadas al Detalle - Esenziale.” Accessed: Mar. 21, 2024. [Online]. Available: https://esenziale.com/tecnologia/partes-drone/

[49] “14 drones con los mejores tiempos de vuelo [grado profesional y hobby].” Accessed: Mar. 21, 2024. [Online]. Available: https://volandocondrones.com/mejores-tiempos-de-vuelo/

[50] “14 drones con los mejores tiempos de vuelo [grado profesional y hobby],” *https://volandocondrones.com/*, Accessed: Mar. 22, 2024. [Online]. Available: https://volandocondrones.com/mejores-tiempos-de-vuelo/

[51] “Welcome to Python.org.” Accessed: Mar. 22, 2024. [Online]. Available: https://www.python.org/

[52] “PyCharm: el IDE de Python para la ciencia de datos y el desarrollo web de JetBrains.” Accessed: Mar. 22, 2024. [Online]. Available: https://www.jetbrains.com/es-es/pycharm/

[53] “PyCharm: el IDE de Python para la ciencia de datos y el desarrollo web.” Accessed: Aug. 22, 2024. [Online]. Available: https://www.jetbrains.com/es-es/pycharm/

[54] “PRISM Live Studio.” Accessed: Mar. 22, 2024. [Online]. Available: https://prismlive.com/en_us/

[55] “OpenCV Tutorial: A Guide to Learn OpenCV in Python.” Accessed: Sep. 25, 2023. [Online]. Available: https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/

[56] “La librería Numpy | Aprende con Alf.” Accessed: Aug. 22, 2024. [Online]. Available: https://aprendeconalf.es/docencia/python/manual/numpy/

[57] “Pillow (PIL Fork) 10.4.0 documentation.” Accessed: Aug. 22, 2024. [Online]. Available: https://pillow.readthedocs.io/en/stable/

[58] “os — Interfaces misceláneas del sistema operativo — documentación de Python - 3.10.13.” Accessed: Aug. 22, 2024. [Online]. Available: https://docs.python.org/es/3.10/library/os.html

[59] “pathlib — Object-oriented filesystem paths — Python 3.12.5 documentation.” Accessed: Aug. 22, 2024. [Online]. Available: https://docs.python.org/3/library/pathlib.html

[60] “Anotación de imágenes: casos de uso, técnicas y tipos clave [2024].” Accessed: Aug. 22, 2024. [Online]. Available: https://es.shaip.com/blog/image-annotation-for-computer-vision/

[61] “GitHub - HumanSignal/labelImg: LabelImg is now part of the Label Studio community. The popular image annotation tool created by Tzutalin is no longer actively being developed, but you can check out Label Studio, the open source data labeling tool for images, text, hypertext, audio, video and time-series data.” Accessed: Aug. 22, 2024. [Online]. Available: https://github.com/HumanSignal/labelImg

[62] “GitHub - labelmeai/labelme: Image Polygonal Annotation with Python (polygon, rectangle, circle, line, point and image-level flag annotation).” Accessed: Aug. 22, 2024. [Online]. Available: https://github.com/labelmeai/labelme

[63] “Visual Geometry Group - University of Oxford.” Accessed: Aug. 22, 2024. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/software/via/

[64] “RectLabel | Ryo Kawamura.” Accessed: Aug. 22, 2024. [Online]. Available: https://rectlabel.com/

[65] “CVAT.” Accessed: Aug. 22, 2024. [Online]. Available: https://www.cvat.ai/

[66] “Supervisely: unified OS for computer vision.” Accessed: Aug. 22, 2024. [Online]. Available: https://supervisely.com/

[67] “Make Sense.” Accessed: Aug. 22, 2024. [Online]. Available: https://www.makesense.ai/

[68] “photo1716735268 (6).jpeg > Deteccion pulgon.” Accessed: Aug. 22, 2024. [Online]. Available: https://app.roboflow.com/pruebas-kwbwh/deteccion-pulgon/images/55Tuy1TslzkSsJyA3uEs?queryText=&pageSize=50&startingIndex=0&browseQuery=true

[69] “YOLOv5 Tutorial - Colab.” Accessed: Aug. 22, 2024. [Online]. Available: https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=iiS7caT81d05

[70] “¿Qué Es NGINX y Cómo Funciona? NGINX explicado para principiantes.” Accessed: Sep. 25, 2023. [Online]. Available: https://kinsta.com/es/base-de-conocimiento/que-es-nginx/

[71] “nginx.” Accessed: Nov. 28, 2023. [Online]. Available: https://nginx.org/en/

[72] “¿Qué Es NGINX Y Cómo Funciona? - Guía Completa.” Accessed: Sep. 25, 2023. [Online]. Available: https://www.hostinger.es/tutoriales/que-es-nginx

[73] “nginx news.” Accessed: Sep. 04, 2024. [Online]. Available: https://nginx.org/