DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA AUTÓNOMO DE DETECCIÓN Y REPULSIÓN DE AVES EN UN CULTIVO DE FRESAS MEDIANTE EL USO DE UN DRON COMERCIAL.

CÓDIGO DE PROYECTO: PG-19-1-11

FÉLIX DAVID GÓMEZ MARÍN

CÓDIGO: 1611340

IDENTIFICACIÓN: C.C. 1020841018

HAROLD MURCIA HERNÁNDEZ

CÓDIGO: 1511177

IDENTIFICACIÓN: C.C. 1020836767

UNIVERSIDAD PILOTO DE COLOMBIA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA MECATRÓNICA

BOGOTÁ, D.C.

2019

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA AUTÓNOMO DE DETECCIÓN Y REPULSIÓN DE AVES EN UN CULTIVO DE FRESAS MEDIANTE EL USO DE UN DRON COMERCIAL.

FÉLIX DAVID GÓMEZ MARÍN

CÓDIGO: 1611340

IDENTIFICACIÓN: C.C. 1020841018

HAROLD MURCIA HERNÁNDEZ

CÓDIGO: 1511177

IDENTIFICACIÓN: C.C. 1020836767

PROYECTO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO MECATRÓNICO DE LA UNIVERSIDAD PILOTO DE COLOMBIA

DIRECTOR:

M.Sc. MARCO ANTONIO JINETE GÓMEZ

M.Sc. en Automatización Industrial

Ing. en Ingeniería Electrónica

CODIRECTOR:

PhD. RUBÉN DARÍO HERNÁNDEZ BELEÑO

PhD. en Ingeniería Mecatrónica

M.Sc. en Ingeniería Mecánica

Ing. en Ingeniería Mecatrónica

UNIVERSIDAD PILOTO DE COLOMBIA

FACULTAD DE INGENIERÍA

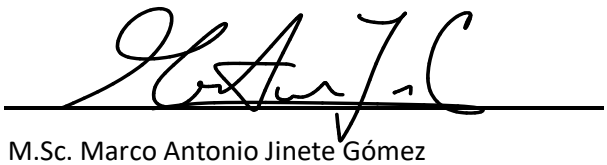
PROGRAMA DE INGENIERÍA MECATRÓNICA

BOGOTÁ, D.C.

2019

NOTA DE ACEPTACIÓN

Una vez realizada la revisión metodológica y técnica del documento final de proyecto de grado, doy constancia de que los estudiantes han cumplido a cabalidad con los objetivos propuestos, cumple a cabalidad con los Lineamientos de Opción de Grado vigentes del programa de Ingeniería Mecatrónica y con las leyes de derechos de autor de la República de Colombia, por tanto, se encuentran preparados para la defensa del mismo ante un jurado evaluador que considere idóneo el Comité de Investigaciones del Programa de Ingeniería Mecatrónica de la Universidad Piloto de Colombia.



M.Sc. Marco Antonio Jinete Gómez

Director del Proyecto

DEDICATORIA

Primero que todo, a Dios, quien me otorgó el don de la inteligencia, y me dio las capacidades necesarias para culminar este proyecto y todo lo que me he propuesto en la vida, por eso y más, le doy gracias a él.

Le agradezco a mis papás, Félix y Janneth, por darme las herramientas para ser quien soy hoy en día, por llenarme de sabiduría, por apoyarme siempre que los necesito, por darme los recursos para poder explotar mis capacidades, por brindarme la oportunidad de enriquecerme con el aprendizaje continuo y por ser siempre los mejores amigos que puedo tener.

Ofrezco un enorme agradecimiento a mi hermanita Sofía, por tenerme paciencia y entender que no podía estar siempre para ella durante el desarrollo del proyecto, por ser el motor de mi vida, por demostrarme que la vida siempre tiene espacio para una risa al día, razón que fue esencial para disfrutar todo el desarrollo del proyecto y de mis estudios universitarios.

Agradezco a mi abuela Nubia, por siempre estar presente cuando la necesito, por no dejarme decaer cuando las adversidades de la vida me hacen querer no continuar, por sus enseñanzas, que fueron vitales para concluir con este capítulo de mi vida. Además, le doy gracias a mi familia porque su apoyo fue esencial para mi desarrollo personal y profesional.

A mis amigos, quienes siempre estuvieron dispuestos a escucharme y apoyarme, con quienes siempre había momento para reír y disfrutar, gracias por acompañarme durante todo el proceso universitario y el desarrollo del proyecto final. También, a todos los docentes que por un momento creyeron en mí y me brindaron todo el conocimiento que me podían otorgar, sin el cual no estaría culminando esta extensa fase de mi vida.

Félix David Gómez Marín

Agradezco a mis papás Milena y Mauricio, quienes a pesar de vivir un poco lejos no dudan ni un segundo en venir cuando los necesito.

Un agradecimiento muy especial a mis abuelos Miryam y Jesús, quienes me han apoyado a nivel personal desde el primer día en la universidad.

Harold Murcia Hernández

AGRADECIMIENTOS

Agradecemos a nuestro director del proyecto, el M.Sc. Marco Antonio Jinete Gómez, quien nos brindó su apoyo y asesoramiento a lo largo de todo el proyecto, quien sin pensarlo dos veces y a pesar de no disponer de mucho tiempo aceptó ser nuestro tutor cuando lo necesitábamos.

Agradecemos a nuestro codirector, el PhD. Rubén Darío Hernández Beleño por su gestión y asesoramiento, y por promover en nosotros la importancia de la investigación; su experiencia y sus consejos fueron claves para el desarrollo de este proyecto.

A la Universidad Piloto de Colombia por permitirnos contar con sus recursos para el desarrollo del proyecto, y por generar en nosotros aprendizaje continuo para el enriquecimiento profesional y personal a lo largo de los estudios.

A la Dirección de Investigaciones de la Universidad Piloto de Colombia por el patrocinio de diversos elementos del proyecto.

Le ofrecemos un agradecimiento muy especial al Ing. Efraín Velásquez y a su familia, por su hospitalidad, por recibirnos siempre con la mejor actitud, y por brindarnos el espacio para la implementación del proyecto.

Agradecemos al PhD. Luis Felipe Herrera Quintero por sus asesorías en clases, su continuo interés en nuestros avances, por sus recomendaciones, ya que fueron muy constructivas y tomadas en cuenta, además de su conocimiento técnico que fue de gran ayuda.

Además, agradecemos al M.Sc. Félix Roberto Gómez Devia por ayudarnos con la metodología empleada para el desarrollo del proyecto de grado, y con la redacción y la implementación de las normativas empleadas en el documento.

CONTENIDO

pág.

LISTA DE CUADROS

pág.

LISTA DE FIGURAS

pág.

INTRODUCCIÓN

Actualmente, la tecnología facilita cientos de tareas, acortando tiempos, costos y esfuerzos. El sector agrícola no es una excepción; las nuevas tecnologías, cada vez más eficientes, económicas y amigables con el medio ambiente, permiten a los agricultores automatizar procesos que suelen ser desgastantes, complicados e, incluso, peligrosos. Sin embargo, existen problemáticas que el costo de tratarlas es superior a las pérdidas y, por lo tanto, los agricultores al no encontrar soluciones alternativas se ven obligados a asumir las pérdidas que las problemáticas generan. Este es el caso de una finca productora de fresas ubicada en el municipio de Guasca en el altiplano cundiboyacense, que ve reducida su producción semanal en aproximadamente 20% por los daños ocasionados por las aves, según lo manifiesta el Ingeniero Agrónomo y dueño de la finca; Efraín Velásquez.

A partir de la necesidad identificada en los cultivos aledaños a la ciudad de Bogotá, en donde las aves pueden dañar las plantas, los frutos, los sembrados e incluso propagar enfermedades, se propone desarrollar un sistema que permita detectar y repeler las aves en un cultivo de fresas sin lastimarlas.

El proceso consiste en un *UAV (V*ehículo Aéreo No Tripulado, por sus siglas en inglés), que por medio de una red sensórica a lo largo del cultivo, pueda localizar las aves y se desplace a su ubicación para repelerlas. Para esto, el agricultor está dispuesto a acomodar un espacio delimitado para la realización de pruebas prácticas. Además, se propone realizar en un plazo de nueve semanas.

RESUMEN

El presente documento, correspondiente al proyecto de grado para la carrera de Ingeniería Mecatrónica de los autores de este.

En el presente documento se propone una forma de resolver el problema de las aves en los cultivos de fresas con el uso de un sistema de visión artificial; se comparan diferentes métodos para la detección de las aves con el uso de una base de datos tomada directamente en el terreno de implementación. Además, se explica el proceso de preprocesamiento para el algoritmo de detección de aves basado en diferencias de imágenes, y el de entrenamiento para el sistema de visión artificial basado en YOLOV3; por consiguiente, se mencionan los pasos para la implementación del proceso de detección seleccionado y su uso para dirigir el dron a repeler las aves dentro del cultivo.

Posteriormente, se expone el proceso de desarrollo del algoritmo capaz de detectar y guiar el dron a la estación de aterrizaje. Y, se evidencian los resultados obtenidos luego de la implementación de lo mencionado en el terreno de pruebas previamente seleccionado; donde se determina que el sistema es capaz de ejecutar todo el proceso sin intervención humana dentro de una autonomía de aproximadamente 2 horas.

Palabras Clave: *visión artificial, procesamiento digital de imágenes.*

ABSTRACT

The current paper, which corresponds to the project for the mechatronic engineering degree of the authors of this one.

This paper proposes a way to solve the problem of birds in strawberry crops with the use of an artificial vision system, comparing different methods for the detection of birds based on a database taken directly in the field of implementation. In addition, the pre-processing process for the bird detection algorithm based on image differences, and the training process for the YOLOV3-based machine vision system are explained; accordingly, the steps for the implementation of the selected detection process and its use to guide the drone to repel birds within the crop are mentioned.

Subsequently, the development process of the algorithm capable of detecting and guiding the drone to the landing station is explained. And, the results obtained after the implementation of the above mentioned are evidenced in the previously selected test field; where it is determined that the system can execute the whole process without human intervention within an autonomy of approximately 2 hours.

Keywords: artificial vision, digital image processing.

1. GENERALIDADES

1.1. PLANTEAMIENTO DEL PROBLEMA

1.1.1. Antecedentes del problema

Luego de establecer contacto con diferentes agricultores y proceder a sus respectivas fincas, se observan pérdidas en diferentes cultivos de fresas a razón de diversos animales. Al indagar en la finca de un conocido de los autores del presente documento, se determina que las aves son la plaga principal y más dañina que tiene su finca. De igual forma, se sigue el mismo patrón en la finca de otros agricultores colombianos; esto hace que se amplíe el campo de observación de la problemática globalmente, y se descubra que la plaga más relevante en los cultivos de fresas son las aves.

1.1.2. Descripción del problema

Se identifica que una finca productora de fresas, ubicada en el municipio de Guasca en el altiplano cundiboyacense, ve reducida, en seis millones de pesos, su producción anual por los daños ocasionados por las aves, según lo manifiesta el ingeniero agrónomo y dueño de la finca; Efraín Velásquez.

Además, las aves, a menudo, dañan las plantas y los sembrados, lo que causa una inversión de esfuerzo constante por parte de los trabajadores para mitigar pérdidas en la producción.

A su vez, se reconoce que se pueden propagar enfermedades a través de las aves en los cultivos y, por consiguiente, en los consumidores finales.

El problema radica en que las soluciones actuales no son viables, ya que pueden llegar a costar más de lo que vale la finca en sí.

1.1.3. Formulación del problema

Con base en lo anterior, se formula la siguiente pregunta de investigación:

¿Cómo implementar un sistema que autónomamente detecte y repele aves en un cultivo de fresas por medio de visión artificial y el uso de un dron comercial?

1.1.4. Línea de investigación del programa

La línea de investigación a la cual se acoge este proyecto es:

- Agricultura de precisión

Esto, debido a que el objetivo principal del proyecto es la implementación de un algoritmo que detecte aves en un cultivo de fresas y posteriormente las asuste con el uso de un dron comercial.

1.2. JUSTIFICACIÓN

Al año 2016, según las cifras más recientes de la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), se encuentran 400.889 hectáreas predispuestas para el cultivo de fresas en el mundo, de las cuales se producen 9.059.557 de toneladas de fresas, que dejan un valor de producción bruto de 17.738,14 millones de dólares a nivel global (1).

A nivel de Colombia, según el Ministerio de Agricultura y Desarrollo Rural (MinAgricultura), se encuentran 1625,68 hectáreas predispuestas para el cultivo de fresas distribuidos en 12 departamentos, de las cuales se producen 61.468,1 toneladas de fresas, que dejan un valor de producción bruto de 109,646173 millones de dólares en Colombia.

Según diferentes estudios realizados por la Universidad de Cornell en Estados Unidos de América, se determina que existe una pérdida del 30% de producción debido exclusivamente a las aves (2); lo que permite entender que en el mundo se están perdiendo 5.321,442 millones de dólares, y en Colombia se están perdiendo 32,8938519 millones de dólares.

Se analiza el terreno propuesto, correspondiente a 28.800 metros cuadrados predispuestos al cultivo de fresas, divididos en 16 lotes que contienen 10.000 matas cada uno; con lo cual, según datos otorgados por el dueño del terreno, se producen 160 toneladas al año, que dan un valor de producción bruto de 320 millones de pesos, y que, a causa de las aves dejan unas pérdidas de seis millones de pesos en el año.

Esto hace que sea necesario desarrollar un sistema funcional que tenga un costo menor al de las pérdidas; de esta forma, se plantea el presente proyecto.

1.3. OBJETIVOS

1.3.1. Objetivo general

Desarrollar e implementar un sistema autónomo de detección y repulsión de aves en un cultivo de fresas mediante un dron comercial.

1.3.2. Objetivos específicos

- Implementar una red sensórica a lo largo del terreno para detectar aves en el cultivo.
- Diseñar un algoritmo que determine la trayectoria del dron, con base en la posición de las aves.
- Programar y adecuar un dron comercial para ahuyentar las aves en el cultivo de manera autónoma.
- Desarrollar una estación de aterrizaje para el reposo del dron en el cultivo.
- Validar la utilidad del sistema propuesto luego de su implementación en el terreno de pruebas.

1.4. DELIMITACIÓN DEL PROYECTO

1.4.1. Alcances y limitaciones

1.4.1.1. Alcances

El proyecto tiene como alcance implementar un algoritmo de detección de aves en un cultivo de fresas ubicado en las afueras de Guasca, Cundinamarca, y posterior desplazamiento de un dron comercial con base en la ubicación sectorial donde se encuentre el ave.

El terreno donde se ha identificado la problemática y corresponde al lugar donde se implementa el sistema se encuentra a 1.35 Km de Guasca, Cundinamarca, cuenta con tres hectáreas divididas en quince lotes, todos cultivados con fresa de forma elevada, nivelada a un metro del suelo, con un sistema de riego por goteo y sin invernadero. Debido a la gran extensión del terreno se decide limitar la superficie de cubrimiento del sistema a menos de cien metros cuadrados para lo correspondiente a demostrar la funcionalidad del proyecto.

1.4.1.2. Limitaciones

La autonomía y calidad del sistema están limitadas a la autonomía y características propias del dron comercial utilizado y las especificaciones de su batería.

El sistema está basado en el uso de un dron comercial, por esta razón el análisis correspondiente a la aerodinámica, resistencia de los materiales, al diseño mecánico y sistema de control propio de este no son contemplados en el desarrollo del proyecto.

La eficiencia del algoritmo de detección de las aves está sujeto a las capacidades y condiciones propias de la instrumentación utilizada.

1.5. MARCO REFERENCIAL

1.5.1. Estado del arte

Actualmente, algunos agricultores acuden a métodos cada vez más drásticos, que pueden influir en la salud de los consumidores finales, como son venenos o insecticidas, otros utilizan métodos físicos que no interfieren en la salud de la planta, sin embargo, las aves se acostumbran a ellos y con el tiempo los ignoran (3).

Los compuestos químicos se han probado reiterativamente, obteniendo resultados dispares, sin embargo, se ha comprobado que algunos cuentan un positivo efecto repelente (4); no necesariamente debe tratarse de un químico que mate a las aves, en Uruguay se realizaron pruebas utilizando colorantes de alimento como repelentes para cotorras, sin embargo, la sociedad busca consumir productos ambientalmente amigables, esto genera dificultades para presentar el producto al campo, y la falta de productos químicos registrados para ese uso limita la aplicación de esta técnica (5).

En Venezuela, los Patos Silbadores se alimentan de cultivos de arroz, el ministerio de ambiente para solucionar el problema autorizó la cacería de estos con fines de control; sin embargo, se refutó esta propuesta, pues los cazadores hacen más daño ingresando a los cultivos que las propias aves (6).

Entre los métodos físicos más frecuentes para espantar aves se encuentran los acústicos, que consisten en emitir un ruido molesto para las aves cuando se detectan cerca, usualmente por sensores de movimiento (7), también se encuentran métodos bioacústicos, que consisten en estudiar la comunicación de los animales a través de señales sonoras, con el fin de determinar cómo atraerlos o espantarlos, este método puede ser usado tanto para atraer aves hacia zonas seguras, como para repelerlas de zonas donde se consideran molestas o peligrosas, como en los cultivos (8).

En Lima, Perú, se desarrolló un sistema de audio que emite sonidos específicos para espantar aves en una planta industrial, se basa en la teoría del instinto de supervivencia del ave. El sistema emula sonidos que el ave interpreta como peligro de amenaza, en este caso se usó el ladrido de los perros como agente atemorizante y se determinó que el sistema las espanta repetidas veces hasta que finalmente las aves abandonaban el lugar (9).

Los agricultores se ven obligados a innovar para evitar la presencia de aves en los cultivos y esto ha generado métodos como los de control, los cuales buscan manipular aquellos elementos que promueven la presencia de las aves, ya que estas suelen concentrarse en determinados lugares según la disponibilidad de alimento y agua, o en espacios donde pueden posar, socializar y anidar. Existen también métodos de señuelo, que consisten en cultivar plantas con frutos comestibles por las aves con el fin de desviar su atención de los campos de cultivo que se desean proteger (10).

Existen, por otro lado, los métodos de barrera, que consisten en cerrar físicamente el lugar donde se encuentran los cultivos que se desean proteger, este método suele resultar definitivo, pero requiere una inversión muy alta en la estructura (11).

El siguiente paso en las estrategias de disuasión física de aves es incluir drones, los drones como espantapájaros se utilizan actualmente en aeropuertos, estos emiten sonidos a lo largo de una ruta determinada para repeler a las plagas (12).

Los drones son una tecnología reciente que se ha convertido en un gran apoyo para el cumplimiento de actividades que anteriormente requerían de gran esfuerzo humano, estas aeronaves pueden cumplir con varios servicios, incluidos algunos relacionados con la agricultura (13).

En los últimos años ha emergido una nueva generación de drones simples de pilotar y de bajo costo, la cual tiene el potencial de ofrecer beneficios en una amplia diversidad de aplicaciones en el campo de la agricultura, arqueología, ayuda humanitaria, monitoreo del tráfico, minería, detección de fallas en gaseoductos, construcción, protección del medio ambiente, entre otros (14).

En los aeropuertos los choques con pájaros causan millones en daños, actualmente en Holanda se desarrolla una innovadora tecnología para ahuyentarlos: un robot que imita el aspecto y comportamiento de un halcón, que es casi imposible de distinguir del real; denominado *Robird* (15).

Un proyecto europeo de investigación desarrolla una flota de tractores autónomos para control de malas hierbas, donde un grupo de drones coordinan la información, sobrevuelan la parcela y toman fotografías para determinar la ubicación de las malas hierbas y enviar estos tractores a realizar el tratamiento correspondiente (16).

Los drones cuentan con varios usos ligados al procesamiento de imágenes, uno de ellos es que mejoran la planificación de las actividades agrícolas realizando tomas aéreas de terrenos, con esto se predicen daños y se toman decisiones adecuadas ante situaciones que puedan afectar el desarrollo normal de un cultivo (17). Actualmente no es necesario recorrer a pie un terreno para determinar los problemas que se presentan, los drones facilitan la evaluación de los cultivos a través de cámaras de alta definición e información georreferenciada. Lo más destacable es que permiten observar de forma eficiente las enfermedades, plagas, malezas y efectos de daños climáticos (18).

Los drones actualmente se utilizan en múltiples campos de la agricultura, principalmente enfocados en la recolección de información, también pueden desarrollar otras actividades como arrojar productos químicos, con fines de fumigación o de riego, también pueden dispersar a los animales como espantapájaros improvisados. Los drones tienen garantizado un uso en la agricultura (19).

1.5.2. Marco normativo

La UAEAC (Unidad Administrativa Especial de Aeronáutica Civil) o Aerocivil, es el organismo estatal colombiano encargado del control y regulación de la aviación civil en Colombia, entidad que emite la Circular Reglamentaria No. 002, la cual tiene como propósito: "Ampliar la información e impartir instrucciones de cumplimiento en referencia a los requisitos de Aeronavegabilidad y Operaciones necesarios para solicitar permiso de acuerdo con lo establecido en los Reglamentos Aeronáuticos de Colombia (RAC), en lo relacionado con la realización de operaciones de Sistemas de Aeronaves Pilotadas a Distancia - RPAS en Colombia" (20).

El 05 de febrero de 2019 la UAEAC publica mediante la resolución #04201 un apéndice a la norma RAC 91: "Por la cual incorporan a la norma RAC 91 de los Reglamentos Aeronáuticos de Colombia unas disposiciones sobre operación de sistemas de aeronaves no tripuladas LIAS y se numeran como Apéndice 13, y se adoptan otras disposiciones" (21).

Este apéndice establece las reglas aplicables que rigen:

- Los requisitos de operación de sistemas de aeronaves no tripuladas (UAS) con peso (masa) máximo al despegue (MTOW) superiora 250 gr, usados en actividades civiles, cualquiera que sea el propósito de utilización, que se efectúen dentro del espacio aéreo en el cual tenga jurisdicción el Estado colombiano, sin perjuicio de lo estipulado en convenios internacionales en los que Colombia sea parte.
- A toda persona natural o jurídica, según corresponda, que:
    - Explote y/u opere UAS con MTOW superior a 250 gr, con cualquier finalidad.
    - Certifique capacitación y entrenamiento en la operación de UAS (21).

Pero, estas reglas no son aplicables para los siguientes casos:

- Operaciones con aeromodelos. Nota. - La reglamentación sobre operación con aeromodelos se encuentra en la norma RAC 4, numeral 4.25.8., o la que en el futuro la modifique o sustituya.
- Operación con UAS con MTOW de hasta 250 gr (21).

Por lo tanto, los drones que pesan menos de 250 gr se encuentran exentos de requerir una autorización para operar por parte de la UAEAC (Unidad Administrativa Especial de Aeronáutica Civil). De esta forma, el dron comercial utilizado en el proyecto debe cumplir con esta característica.

2. IMPLEMENTACIÓN DE RED SENSÓRICA

2.1. SELECCIÓN DE SENSORES

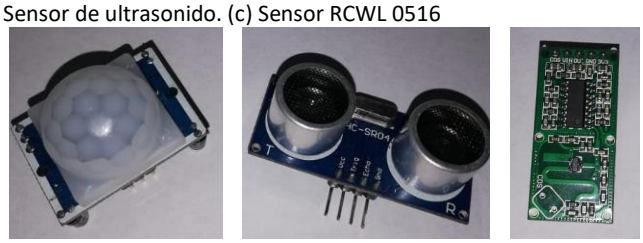
Es necesario implementar una red sensórica que indique cuándo el ave ingresa al cultivo y cuándo se detiene en algún punto, para esto, es necesario mencionar que el área a tratar es de cien metros cuadrados, distribuidos en un terreno cuadrado de diez metros por diez metros. Y, con esto en cuenta, se procede a comparar diferentes sensores a continuación.

PIR: los sensores *PIR* por sus siglas en inglés (*Passive InfraRed*), detectan cambios en la radiación infrarroja que emite un cuerpo. El sensor se visualiza en la *Figura 1 (a).*

Ultrasonido: detector de proximidad que emite un sonido y mide el tiempo que toma en regresar. Se expone en la *Figura 1 (b).*

RCWL 0516: este sensor detecta movimiento utilizando la técnica de radar Doppler. El sensor se puede observar en la *Figura 1 (c).*

Figura 1. (a) Sensor PIR. (b) Sensor de ultrasonido. (c) Sensor RCWL 0516



(a) (b) (c)

Fuente: Elaboración propia

Para destacar las características de cada sensor se realiza el cuadro comparativo: *Cuadro 1*, que considera la escala de 1 a 4, donde 4 es lo mejor para el proyecto; hecho de acuerdo con la experiencia de uso y la hoja de especificaciones de cada uno de los sensores.

Cuadro 1. Comparación entre sensores

Característica	PIR	Ultrasonido	RCWL0516	Cámara
Precio por unidad	4	3	2	1
Cantidad necesaria	2	1	3	4
Cableado	2	1	3	4
Confiabilidad	3	2	1	4
Requerimiento técnico	2	3	4	1
Área de cobertura por sensor	2	1	3	4

Fuente: Elaboración propia

De acuerdo con la ponderación realizada en el *Cuadro 1* el sensor que obtiene el mejor puntaje es la cámara, debido a que se requiere menos cableado, permite cubrir mayor área por cada sensor, y se necesitan menos unidades que en el caso de los otros sensores. Por lo tanto, se escoge la cámara como sensor para el uso e implementación en el proyecto.

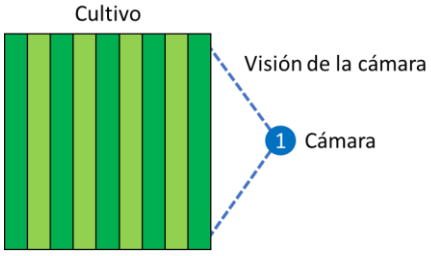
2.2. UBICACIÓN DE SENSORES

Con base en la posición de las cámaras se determina el algoritmo que debe usarse para detectar las aves en el cultivo. Se proponen: el sistema de cámara lateral, el de cámara aérea y el sistema de cámaras laterales, los cuales se describen a continuación.

2.2.1. Sistema de cámara lateral

Este sistema consiste en ubicar una cámara a uno de los laterales del cultivo, de tal manera que se observen los surcos de manera horizontal. La siguiente figura muestra la ubicación de la cámara:

Figura 2. Diagrama del sistema de cámara lateral



Fuente: Elaboración propia.

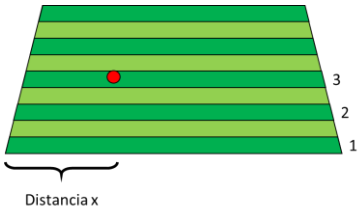
Al ubicar la cámara en esta posición, a una altura determinada, se espera que la imagen se visualice de la siguiente manera:

Figura 3. Vista del cultivo desde la cámara lateral

Fuente: Elaboración propia.

Al observar el cultivo de manera horizontal, se pueden diferenciar los surcos, al detectarse un ave en la imagen el algoritmo mide su coordenada en *x* y cuenta los surcos a los que se encuentra, a través de haber nombrado previamente cada surco, para determinar la coordenada *y*, de esta manera se obtiene la ubicación del ave para posteriormente enviar el dron. En la *Figura 4* se muestra un ejemplo de la detección de un ave, correspondiente al punto rojo, que aterriza en una coordenada *x* y con una distancia de 3 surcos para la coordenada *y.*

Figura 4. Ejemplo de ubicación del ave en sistema de cámara lateral

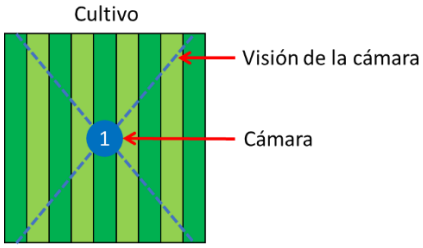


Fuente: Elaboración propia.

2.2.2. Sistema de cámara aérea

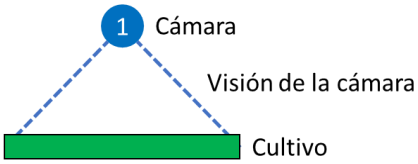
Este sistema consiste en ubicar una cámara a una altura determinada, de tal forma que permita observar el cultivo totalmente, desde arriba. Las *Figuras 5 y 6* muestran la ubicación de la cámara y la vista del cultivo desde esta posición:

Figura 5. Diagrama del sistema de cámara aérea vista superior



Fuente: Elaboración propia.

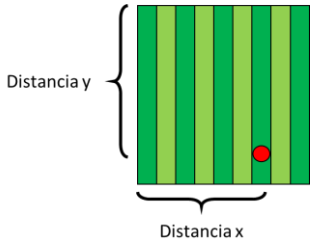
Figura 6. Diagrama del sistema de cámara aérea vista lateral



Fuente: Elaboración propia.

Observar el sistema desde arriba permite saber con exactitud el lugar en el que se encuentra un ave, ya que se observa la coordenada en *x* y la coordenada en *y* directamente desde la vista de la cámara, de esta manera se obtiene la ubicación del ave para posteriormente enviar el dron. En la *Figura 7* se muestra un ejemplo de la detección de un ave, siendo ésta el punto rojo:

Figura 7. Ejemplo de ubicación del ave en sistema de cámara aérea



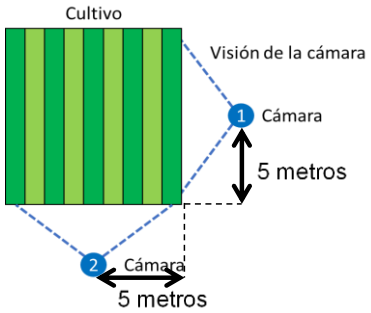
Fuente: Elaboración propia.

Para ubicar la cámara en una altura que se vea todo el cultivo se requiere de una estructura que genera costos adicionales y una intervención física dentro del terreno. Esto no es funcional, debido a que se busca que el sistema no genere invasión permanente, por lo tanto, el sistema de cámara aérea se descarta de una vez.

2.2.3. Sistema de cámaras laterales

Este sistema consiste en ubicar una cámara a un lado del cultivo, observando los surcos de manera horizontal, y otra observando los surcos de manera vertical. La *Figura 8* muestra el diagrama de la ubicación de las cámaras:

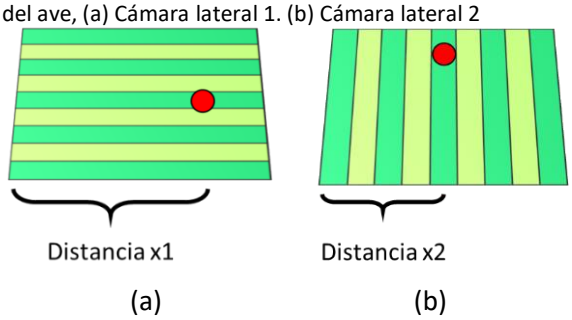
Figura 8. Diagrama del sistema de cámaras laterales



Fuente: Elaboración propia.

Al ubicar las cámaras en esta posición las imágenes permiten visualizar el cultivo como en la *Figura 9.* Al observar el sistema desde dos ángulos, se ubica con exactitud el lugar en el que se encuentra un ave, debido a que se observan las coordenadas en *x* de cada una de las cámaras para determinar mediante el cruce de las coordenadas el sector en el que se encuentra el ave. Un ejemplo de la detección de un ave en el cultivo se expone en la *Figura 9*.

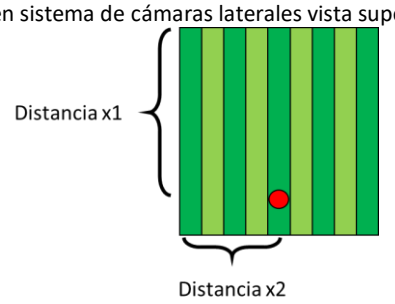
Figura 9. Ejemplo de ubicación del ave, (a) Cámara lateral 1. (b) Cámara lateral 2



Fuente: Elaboración propia.

Con ambas cámaras trabajando en conjunto, cada una sólo debe extraer la coordenada *x* del ave, para posteriormente unirlas y determinar las coordenadas en el plano del cultivo. En la *Figura 10* se muestra la representación de las coordenadas en un plano *2D* de la ubicación del ave, siendo ésta el punto rojo:

Figura 10. Ejemplo de ubicación del ave en sistema de cámaras laterales vista superior



Fuente: Elaboración propia.

2.2.4. Comparación de los sistemas

Para la selección del sistema a implementar, se destacan aspectos puntuales como, el cableado, la cantidad de cámaras y el tiempo de procesamiento, a continuación:

- El sistema de cámara lateral sólo requiere de una cámara, mientras que el sistema de cámaras laterales requiere de dos cámaras.
- Se estima un menor tiempo de procesamiento para el sistema de cámara lateral.
- El sistema de cámara lateral requiere menor cableado que el sistema de cámaras laterales.
- El sistema de cámaras laterales asegura la ubicación del ave, mientras que el sistema de cámara lateral puede no detectar el ave según ciertas posiciones.

Con base en lo anterior, se determina que el mejor sistema para la implementación en el proyecto es el sistema de cámaras laterales.

2.3. SELECCIÓN DE LAS CÁMARAS

Para seleccionar las cámaras, se realiza el siguiente cuadro comparativo entre las tres principales, y más económicas encontradas en el mercado nacional colombiano:

Cuadro 2. Comparación entre cámaras

Nombre	Resolución de video	FPS	Campo de visión	Precio
Genius FaceCam 1000X	720p	30	60°	$60.000
Genius Qcam 6000	1080p	30	90°	$115.000
Logitech C170	720p	30	58°	$55.000

Fuente: Elaboración propia.

De acuerdo con el *Cuadro 2*, se selecciona la Logitech C170 por la relación entre la calidad y el precio, ya que se considera que una resolución de 720p es suficiente para trabajar a la distancia planteada, correspondiente a diez metros en línea recta para cada cámara, y el campo de visión es óptimo para los fines del proyecto.

3. BASE DE DATOS

La toma de una base de datos es fundamental para todo sistema que implemente procesamiento digital de imágenes, debido a que es la parte base para la caracterización del problema en el contexto real, y es lo que permite generar un entrenamiento o procesamiento directo para algoritmos de visión artificial.

En esta sección se explica cómo se realiza el proceso de adquisición de datos en formato de vídeos, la conversión de estos a imágenes, el proceso de filtrado manual con base en el criterio de detección de aves visualmente, y el posterior etiquetamiento de las imágenes seleccionadas para el directo procesamiento con diferentes métodos mencionados en el siguiente capítulo.

3.1. TOMA PRINCIPAL DE DATOS

Lo primero que se debe realizar es llevar a cabo una toma de datos general, para esto, se toman vídeos en las condiciones reales de acuerdo con la ejecución del proyecto, para lo cual se usa la red de cámaras implementadas y seleccionadas en el capítulo 2.

Se toman vídeos que cumplen con las condiciones mencionadas a cinco horas diferentes del día; esto con el fin de contemplar cambios de iluminación, variación en la velocidad del viento, e incremento o disminución en la cantidad de aves, de acuerdo con los alcances definidos para el proyecto. Se adquiere un total de cinco vídeos con una duración total de 92 minutos y 9 segundos de datos.

Para la visualización en el presente documento se escogen aleatoriamente 4 diferentes tomas de los diferentes videos tomados y se reúnen en la *Figura 11*.

Figura 11. Capturas del video de la base de datos

Fuente: Elaboración propia.

Las características de la toma de datos que se realiza en los cinco vídeos se pueden evidenciar en el siguiente cuadro, correspondiente al *Cuadro 3*:

Cuadro 3. Duración de los videos de la base de datos

Vídeo	Duración (MM:SS)	Hora de grabación
1	23:41	09:00 AM
2	13:48	10:00 AM
3	14:07	11:00 AM
4	32:25	12:00 PM
5	08:08	01:00 PM

Fuente: Elaboración propia.

Como segundo paso, se convierten los vídeos obtenidos a imágenes, de tal forma que se obtienen las características relacionadas en el siguiente cuadro:

Cuadro 4. Imágenes resultantes por video de la base de datos

Vídeo	Cantidad de imágenes resultantes
1	1422
2	829
3	848
4	1946
5	489
TOTAL	5534

Fuente: Elaboración propia.

3.2. SELECCIÓN DE IMÁGENES

Es necesario seleccionar las imágenes que contengan el objeto de análisis (entendiendo como objeto de análisis a las aves que causan daños en el cultivo). Por lo tanto, luego de reunir todas las imágenes resultantes de la sección 3.1 en una carpeta, se seleccionan, observando imagen por imagen, las imágenes que contenga el objeto de análisis; obteniendo un total de 253 imágenes.

Pero, a su vez, con el objetivo de adquirir una base de datos completa, se seleccionan aleatoriamente imágenes en donde no esté el objetivo de análisis con base en una relación entre las imágenes que contienen el objeto de análisis y las imágenes que no contienen el objeto de análisis de 1 a 1, por lo tanto, se obtienen un total de 253 imágenes que no contienen el objeto de análisis.

De esta forma, se dice que se obtiene una base de datos útil con 506 imágenes.

3.3. ETIQUETAMIENTO DE IMÁGENES

Posteriormente a la obtención de la base de datos útil, se requiere un análisis concreto para identificar puntualmente dónde está el objeto de análisis en cada imagen; para esto, se realiza un proceso conocido como etiquetamiento, el cual consiste en ubicar por medio de coordenadas cartesianas el objeto de análisis, indicar el tamaño del objeto relativamente con el tamaño de la imagen y asignarle un nombre con el que se diferencia de lo demás dentro de la imagen.

Para llevar a cabo los pasos que requiere el proceso de etiquetamiento, se utilizan dos herramientas desarrolladas en Python y publicadas en GitHub públicamente para su uso (22). Y, se siguen los pasos mencionados en los siguientes incisos.

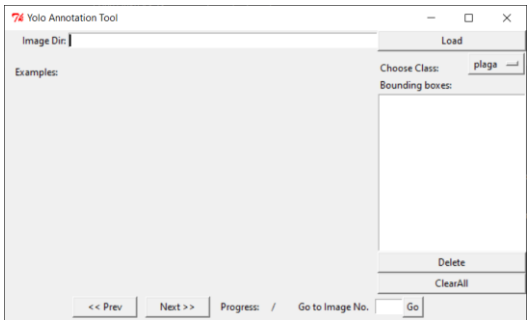
3.3.1. Selección de nombre para usar como etiqueta

Para diferenciar el objeto de análisis de lo demás, dentro de la imagen, es necesario asignarle un nombre particular al momento de realizar la etiqueta. Para esto, se decide llamar al objeto de análisis como *plaga*, de tal forma que cualquier ave es referenciada de esta manera, y se crea un archivo de texto con el nombre *classes*, que dentro de él sólo se encuentra la palabra de etiqueta, es decir *plaga*.

3.3.2. Yolo Annotation Tool

Luego de determinar la ubicación de las imágenes y crear el archivo *classes.txt*, se procede a ejecutar la herramienta de anotación principal, *Yolo Annotation Tool*. Se debe observar una ventana como la expuesta en la *Figura 12*.

Figura 12. Ventana de la herramienta de anotación



Fuente: Elaboración propia.

Ahora, se debe crear una carpeta llamada *Images* que contenga todas las imágenes seleccionadas en la sección 3.2. Consecuentemente, se debe dar clic en el botón *Load* en la ventana *Yolo Annotation Tool*; una vez hecho esto, se evidencia lo siguiente:

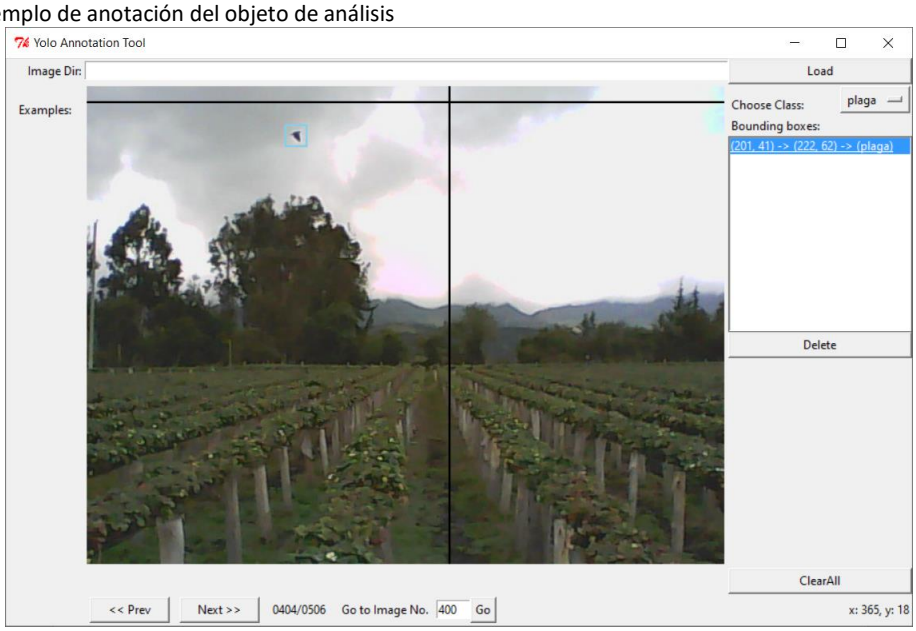
Figura 13. Herramienta de anotación con una imagen cargada

Fuente: Elaboración propia.

En seguida, se selecciona para cada imagen la ubicación del objeto de análisis, encerrándolo en un cuadro, y dando clic en el botón *Next*; en caso de no encontrarse el objeto de análisis, se debe dar clic en el botón *Next* sin seleccionar.

La *Figura 14* muestra cómo queda el proceso de selección luego de encerrar el ave en un cuadro:

Figura 14. Ejemplo de anotación del objeto de análisis



Fuente: Elaboración propia.

Este proceso se debe repetir para las 506 imágenes que se encuentran en la carpeta *Images*.

Una vez realizado el proceso, se deben crear los archivos *train* y *test*, en formato *txt*, los cuales contienen la ruta a la ubicación de cada imagen, donde se encuentran un 80 por ciento de las imágenes en el archivo *train.txt* y el 20 por ciento restante dentro del archivo *test.txt*; con los cuales se entrena y valida el entrenamiento, respectivamente.

Para crear estos archivos se ejecuta el código llamado *process.py*; luego de ejecutarlo se pueden evidenciar los archivos *train.txt* y *test.txt* dentro de la carpeta raíz.

Con esto se termina el proceso de etiquetamiento de imágenes de la base de datos útiles, y se obtienen los archivos necesarios para poder realizar el entrenamiento o procesamiento directo para la detección de las aves con algoritmos de visión artificial.

4. ALGORITMOS PARA LA DETECCIÓN DE LAS AVES

El presente apartado expone el desarrollo llevado a cabo para la detección de las aves encontradas en el cultivo de fresas, destacando y comparando aspectos relevantes de los diferentes métodos usados. A su vez, es necesario destacar que todos los algoritmos usados en este capítulo son desarrollados únicamente en el lenguaje interpretado Python, y principalmente con el uso de la librería *OpenCV*, correspondiente a una librería, de uso libre, enfocada al procesamiento digital de imágenes.

Además, es necesario mencionar que todos los procedimientos llevados a cabo en este capítulo son realizados con una computadora que se compone de: un procesador *Intel Core I7-7700HQ*, una tarjeta gráfica *NVIDIA GeForce GTX 1050 Ti*, una tarjeta *RAM* DDR4 de *16 GB*, y un disco de estado sólido de *256GB*.

4.1. IMPLEMENTACIÓN EN AMBIENTES CONTROLADOS

En esta sección se comparan aspectos que se destacan al momento del procesamiento de una prueba de implementación en ambientes controlados con diferentes métodos de visión artificial, con el uso de algoritmos diseñados para detectar caras, personas o, en su defecto, movimiento, para al final determinar los más apropiados para su implementación en el contexto real. Se parte de la suposición de que, si la detección de lo mencionado no se da en un ambiente controlado, no se puede llevar a cabo en la implementación del proyecto, donde no se tiene control de la situación, es decir, en la detección de las aves.

Para el proceso de detección, siguiendo el procedimiento expuesto en el capítulo 3, sin realizar el procedimiento de etiquetamiento de imágenes y solamente seleccionando las imágenes con personas**,** se desarrolla una base de datos con un vídeo, de elaboración propia, de una persona en movimiento; durante el desarrollo del presente capítulo se llama base de datos experimental. Para la visualización en el presente documento se escogen 4 fotogramas y se reúnen en la *Figura 15*.

Figura 15. Fotogramas de la base de datos experimental

Fuente: Elaboración propia.

4.1.1. Algoritmo por diferencia de imágenes

El algoritmo por diferencia de imágenes permite observar los cambios que se registran entre dos imágenes, en este caso aplicado al movimiento de una persona de acuerdo con la base de datos experimental.

El proceso general se expone en formato de pseudocódigo en el *Cuadro 5*:

Cuadro 5. Pseudocódigo del algoritmo por diferencia de imágenes

Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Cargar el par de fotogramas
4	Restar cada píxel entre ambos fotogramas
5	Binarizar la imagen resultante
6	Detección visual del cambio

Fuente: Elaboración propia

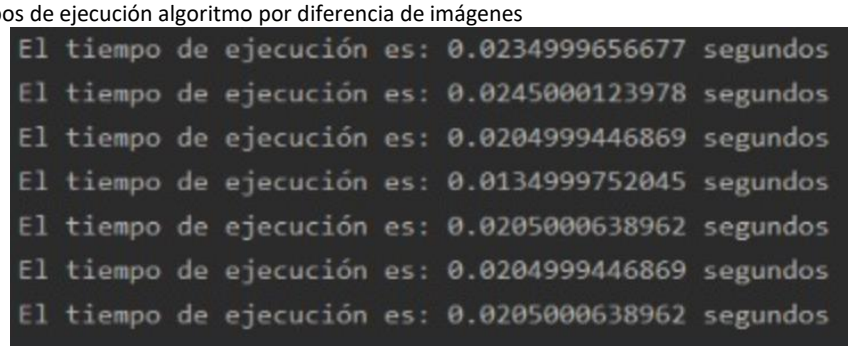
Ahora, se ejecuta el código para cada imagen de la base de datos experimental, obteniendo lo que se observa en la *Figura 16*, para tres pares de fotogramas:

Figura 16. Resultados del algoritmo por diferencia de imágenes

Fuente: Elaboración propia.

Según se puede evidenciar en la *Figura 16*, el algoritmo por diferencia de imágenes detecta correctamente el movimiento en todas las imágenes según el movimiento particular y lo diferencia del fondo estático apropiadamente. Además, lo realiza en un tiempo de ejecución promedio de 0.0204 segundos para cada par de fotogramas, según se expone en la *Figura 17*.

Figura 17. Tiempos de ejecución algoritmo por diferencia de imágenes



Fuente: Elaboración propia.

4.1.2. Clasificador HAAR Cascade

El método clasificador *HAAR Cascade* es uno de los más comunes cuando se habla de procesamiento digital de imágenes, esto se debe a que es simple de usar, rápido y permite detectar cualquier patrón correctamente entrenado.

Para evaluar su implementación se diseña un algoritmo de detección de caras utilizando el modelo pre-entrenado para la detección de rostros, de uso libre, *haarcascade_frontalface_alt.*xml.

El pseudocódigo del algoritmo utilizado se expresa en el *Cuadro 6* a continuación:

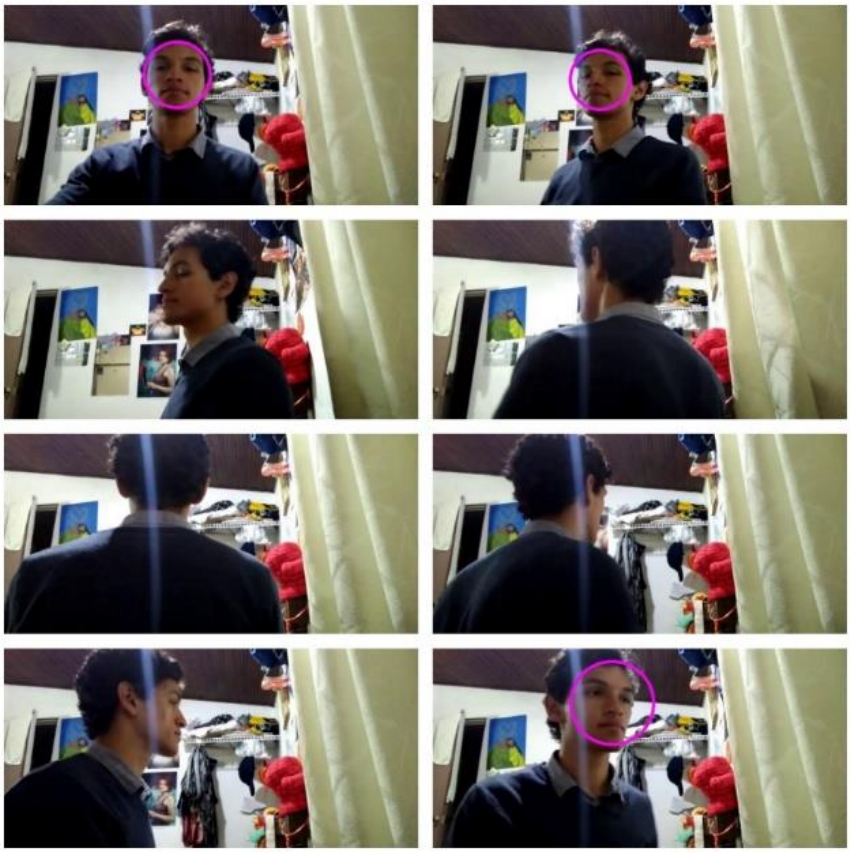
Cuadro 6. Pseudocódigo clasificador HAAR Cascade

Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Cargar el clasificador
4	Cargar la imagen
5	Convertir imagen a escala de grises
6	Ecualizar el histograma de la imagen en escala de grises
7	Marcar un área rectangular como ventana de detección
8	Calcular la diferencia de la suma de intensidades en cada píxel de la ventana de detección
9	Ubicar posibles posiciones de las caras para la ventana de detección
10	Validar la identificación de las caras en la ventana de detección con clasificadores débiles
11	Repetir desde la línea 7 con cada área rectangular marcada
12	Ubicar los resultados positivos
13	Detección visual de las caras

Fuente: Elaboración propia

Las líneas 7 a 12 expuestas en el *Cuadro 6* corresponden al proceso general interno del método *Clasificador HAAR Cascade* (23). Ahora, se ejecuta el código para cada imagen de la base de datos experimental, obteniendo lo que se observa en la *Figura 18:*

Figura 18. Resultados del método *Clasificador HAAR Cascade*



Fuente: Elaboración propia.

Como se evidencia en la *Figura 18***,** el método *Clasificador HAAR Cascade*, sólo detecta las caras cuando estas están prácticamente de frente a la cámara; esto se debe a que este método busca dos ojos, una nariz, una boca, esto distribuido en forma de T.

Con lo anterior, se determina que se requiere un patrón especifico en el proceso de detección, por lo tanto, debido a que el objeto de análisis del proyecto no permite contar con esta característica, no se analiza el tiempo de ejecución de la detección de la cara y se descarta el método de una vez.

4.1.3. TensorFlow

Es un ecosistema virtual para aplicaciones en aprendizaje de máquina, desarrollado por Google, focalizado en la implementación en máquinas con *GPU,* y con orientación principalmente al desarrollo de algoritmos de inteligencia artificial y máquinas de visión artificial.

Debido a la complejidad de desarrollo de los métodos para *TensorFlow*, se utiliza el algoritmo de detección de personas y objetos desarrollado por Google, y se edita para la realización de pruebas con el uso de la base de datos experimental (24). Adicionalmente, se implementa el algoritmo en la *GPU* de la computadora.

El proceso general corresponde al expuesto en formato de pseudocódigo en el *Cuadro 7*:

Cuadro 7. Pseudocódigo detección por TensorFlow

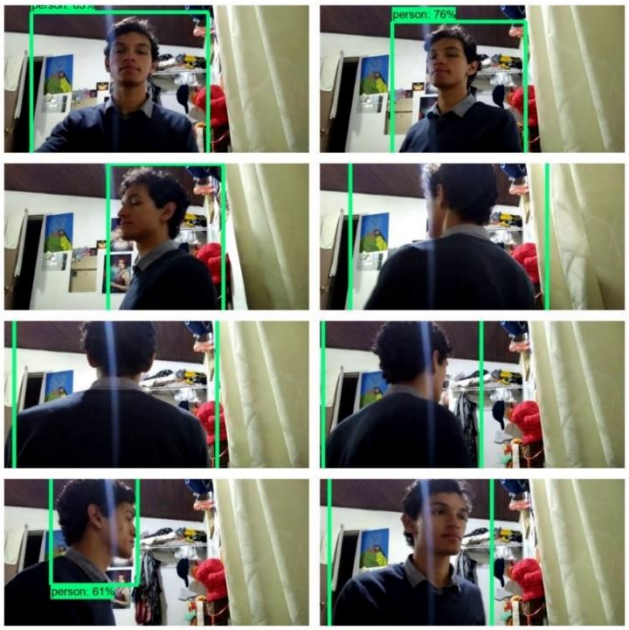
Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Cargar el modelo de detección de objetos
4	Cargar imagen
5	Analizar la imagen por secciones
6	Detectar el patrón neuronal en cada sección
7	Marcar un área rectangular en cada patrón
8	Integrar las áreas resultantes dentro de la imagen
9	Predecir los desplazamientos de las áreas resultantes integradas
10	Ubicar las predicciones finales
11	Enviar secciones al módulo de servicio TensorFlow
12	Recibir respuesta de análisis
13	Detección visual de las personas

Fuente: Elaboración propia

Las líneas 5 a 12 expuestas en el *Cuadro 7* corresponden al proceso general interno de *TensorFlow* con el uso del modelo *SSD* (25) para la detección de las personas.

Ahora, se ejecuta el código para cada imagen de la base de datos experimental, obteniendo lo que se observa en la *Figura 19* mostrada a continuación:

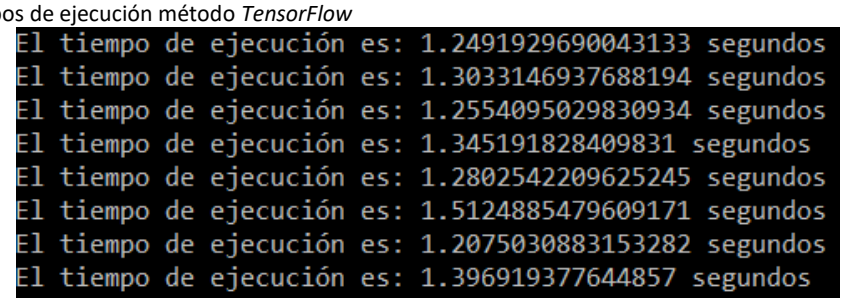
Figura 19. Resultados del método *TensorFlow*



Fuente: Elaboración propia.

Como se evidencia en la *Figura 19, TensorFlow* detecta a la persona que está en cada fotograma independientemente de la ubicación de ésta y lo realiza en un tiempo de ejecución promedio de 1.3137 segundos, según se puede ver en la *Figura 20***.**

Figura 20. Tiempos de ejecución método *TensorFlow*



Fuente: Elaboración propia.

4.1.4. CVLIB como método de detección

CVLIB es un método de visión artificial para Python, simple, de alto nivel, fácil de usar, y de uso libre (26). Está basada en la librería de aprendizaje profundo *Keras*, por tanto, su uso es complementario a *TensorFlow*, pero se menciona en una sección aparte, debido a que su implementación es más simple y rápida que *TensorFlow*.

Para evaluar su implementación se diseña un algoritmo de detección de caras, el cual está en la capacidad de detectar los rostros de la base de datos experimental siempre que una persona físicamente sea capaz de validarlo, con base en el pseudocódigo expresado a continuación en el *Cuadro 8*:

Cuadro 8. Pseudocódigo de CVLIB como método de detección

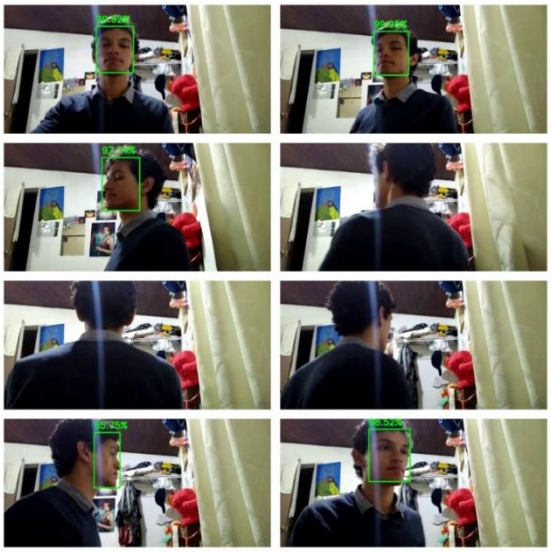
Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Cargar la imagen
4	Definir umbral de detección de la cara
5	Cargar archivo de pesos pre-entrenados del modelo
6	Restar el centro de RGB del modelo para cada píxel de la imagen
7	Computar el *blob* final de la imagen
8	Enviar resultados al módulo de servicio TensorFlow
9	Recibir respuesta de análisis
10	Detección visual de las caras

Fuente: Elaboración propia

Las líneas 4 a 9 expuestas en el *Cuadro 8* corresponden al proceso general interno del método *CVLIB* (27) para la detección de objetos.

Ahora, se ejecuta el código para cada imagen de la base de datos experimental, obteniendo lo que se observa en la *Figura 21* mostrada a continuación:

Figura 21. Resultados del método *CVLIB*



Fuente: Elaboración propia.

Como se evidencia en la *Figura 21***,** el método *CVLIB* detecta la cara de la persona que está en cada fotograma cuando una persona físicamente también la observa, y lo realiza en un tiempo de ejecución promedio de 0.8912 segundos, según se puede ver en la *Figura 22.*

Figura 22. Tiempos de ejecución método *CVLIB*

```
El tiempo de ejecución es: 1.275606632232666 segundos
El tiempo de ejecución es: 1.196540117263794 segundos
El tiempo de ejecución es: 0.6664605140686035 segundos
El tiempo de ejecución es: 0.6544144153594971 segundos
El tiempo de ejecución es: 0.2542734146118164 segundos
El tiempo de ejecución es: 1.181847333908081 segundos
El tiempo de ejecución es: 1.2870218753814697 segundos
El tiempo de ejecución es: 0.6572389602661133 segundos
```

Fuente: Elaboración propia.

4.1.5. YOLO

El método *Tú Sólo Miras Una Vez*, *YOLO* por sus siglas en inglés, es lo último desarrollado en sistemas de detección de objetos en tiempo real (28). Su nombre proviene del concepto de su funcionamiento, el cual está basado en mirar la imagen una vez y realizar predicciones con una sola red neuronal con un modelo previamente entrenado, las predicciones se analizan por secciones de píxeles que se denominan *cajas*, haciéndolo más rápido que los otros algoritmos de detección de objetos.

Actualmente se ha desarrollado un modelo más eficiente, conocido como *YOLOV3*, que es, como se puede deducir del nombre, la versión 3 del sistema *YOLO* original; es capaz de realizar predicciones más rápidamente debido a la capacidad de multi-escala y un clasificador incorporado. Adicionalmente, los creadores del sistema han desarrollado un sistema más rápido llamado *YOLOV3-TINY*.

El pseudocódigo del algoritmo general, tanto para *YOLOV3* como para *YOLOV3-TINY*, utilizado se expresa a continuación en el *Cuadro 9*:

Cuadro 9. Pseudocódigo de CVLIB como método de detección

Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Cargar archivo de configuración
4	Cargar archivo de pesos
5	Cargar archivo de nombres
6	Cargar la imagen
7	Cargar la arquitectura de la red
8	Extraer características por la cantidad de capas convolucionales
9	Predecir clase de objeto por las características
10	Marcar un área de caja para la clase de objeto
11	Predecir cajas de objeto por secciones de pixeles
12	Marcar un mapa de probabilidades de clase
13	Definir umbral de intersección sobre la unión
14	Definir umbral de supresión no máxima
15	Detección visual de las personas

Fuente: Elaboración propia

Las líneas 8 a 12 expuestas en el *Cuadro 9* corresponden al proceso general interno del método *YOLO* (29) para la detección de objetos.

Ahora, se expone la implementación de ambos métodos en los siguientes incisos usando el modelo pre-entrenado para la detección de múltiples objetos y el marco de redes neuronales *Darknet* desarrollado por los mismos creadores de *YOLO*.

4.1.5.1. YOLOV3

YOLOV3 es el método por excelencia para la detección de objetos, corresponde a la versión grande, robusta y de alta eficiencia. Está desarrollado principalmente para su uso con *GPU*, de hecho, es 500 veces más rápido que con *CPU* (30); por tanto, la prueba de implementación del sistema se realiza con *GPU*.

Se ejecuta el código para cada imagen de la base de datos experimental, cargando los archivos de configuración y de pesos de la versión *YOLOV3*, obteniendo los resultados mostrados en la *Figura 23* expuesta a continuación:

Figura 23. Resultados del método *YOLOV3*

Fuente: Elaboración propia.

Como se evidencia en la *Figura 23***,** el método *YOLOV3* detecta correctamente a la persona que está en cada fotograma independientemente de la ubicación de ésta, además detecta a la persona en la pared y algunos objetos, y lo realiza en un tiempo de ejecución promedio de 0.0517 segundos, según se observa en la *Figura 24.*

Figura 24. Tiempos de ejecución del método *YOLOV3*

```
El tiempo de ejecución es: 0.053856849670410156 segundos
El tiempo de ejecución es: 0.05688977241516113 segundos
El tiempo de ejecución es: 0.048899173736572266 segundos
El tiempo de ejecución es: 0.050891876220703125 segundos
El tiempo de ejecución es: 0.05184221267700195 segundos
El tiempo de ejecución es: 0.05485200881958008 segundos
El tiempo de ejecución es: 0.0508272647857666 segundos
```

Fuente: Elaboración propia.

4.1.5.2. YOLOV3-TINY

YOLOV3-TINY es la versión reducida de *YOLOV3*, orientada a ambientes controlados y sistemas embebidos (31), es más rápida que *YOLOV3,* pero sacrifica la precisión de las predicciones, por lo que lo hace un algoritmo no tan eficiente como *YOLOV3*.

Se ejecuta el código para cada imagen de la base de datos experimental, cargando los archivos de configuración y de pesos de la versión *YOLOV3-TINY*, obteniendo los resultados mostrados en la figura expuesta a continuación:

Figura 25. Resultados del método *YOLOV3-TINY*

Fuente: Elaboración propia.

Como se evidencia en la *Figura 25,* el método *YOLOV3-TINY* detecta correctamente a la persona que está en cada fotograma independientemente de la ubicación de ésta, pero a diferencia del método *YOLOV3* no detecta a la persona en la pared y realiza la detección con menor probabilidad de acierto, pero lo realiza en un tiempo de ejecución promedio de 0.01885 segundos, según se observa en la *Figura 26.*

Figura 26. Tiempos de ejecución del método *YOLOV3-TINY*

```
El tiempo de ejecución es: 0.018949508666992188 segundos
El tiempo de ejecución es: 0.020942211151123047 segundos
El tiempo de ejecución es: 0.02094411849975586 segundos
El tiempo de ejecución es: 0.016954660415649414 segundos
El tiempo de ejecución es: 0.02094435691833496 segundos
El tiempo de ejecución es: 0.019946813583374023 segundos
El tiempo de ejecución es: 0.019946575164794922 segundos
```

Fuente: Elaboración propia.

4.1.6. Comparación de los métodos de implementación

Se pueden resaltar características comparativas entre los diferentes sistemas o métodos implementados, para esto, se realiza el cuadro comparativo, correspondiente al *Cuadro 10*, con el objetivo de filtrar los sistemas y seleccionar aquellos que sean los más apropiados para implementar en la detección de aves en el campo.

Cuadro 10. Primera comparación de los métodos de detección

Método de implementación	Tiempo de ejecución promedio (segundos)	Fotogramas por segundo	Probabilidad de detección más alta
Diferencia de imágenes	0.0204	49.01	No aplica
TensorFlow	1.3137	0.76	0.76
CVLIB	0.8912	1.12	0.99
YOLOV3	0.0517	19.34	1
YOLOV3-TINY	0.0188	53.19	0.95

Fuente: Elaboración propia.

De acuerdo con la información registrada en el *Cuadro 10*, se descartan los métodos que tienen una ejecución menor a 15 fotogramas por segundo ya que no permiten la implementación en el contexto real, y de los métodos que quedan, se descartan los que tienen una *probabilidad de detección más alta* menor a 0.99, ya que se busca la mayor probabilidad de detección debido a que el objeto de análisis es pequeño y se puede confundir con otros objetos; por lo tanto, para realizar el proceso de implementación real, con la base de datos adquirida en el capítulo 3, se puede utilizar el sistema de detección de objetos *YOLOV3.* Por otra parte, el algoritmo por diferencia de imágenes también se puede utilizar, debido a que detecta correctamente el movimiento en cada prueba realizada, según se observa en la sección 4.1.1 y, además, lo realiza con un radio de procesamiento de 49.01 fotogramas por segundo.

4.2. IMPLEMENTACIÓN CON BASE DE DATOS DE LOS MÉTODOS FITLRADOS

En la presente sección, se expone la implementación del método *YOLOV3* y el algoritmo por diferencia de imágenes para la implementación en el campo, por lo tanto, se realiza todo el procedimiento que cada uno de estos requiere en el contexto real. Además, se realiza una comparación entre ambos métodos y se selecciona el más adecuado para el proyecto.

4.2.1. YOLOV3

El proceso de implementación requiere de un proceso de entrenamiento, el cual es necesario para que el sistema sepa qué detectar en el contexto real, y posteriormente un proceso de implementación con los archivos resultantes en el proceso de entrenamiento.

4.2.1.1. Entrenamiento

Para llevar a cabo el proceso de entrenamiento correctamente, y así evitar la pérdida de eficiencia del sistema por tiempo de ejecución y falsas detecciones, se debe realizar lo siguiente:

Primero, se debe seleccionar un modelo pre-entrenado para clasificación de redes de imágenes según la velocidad de procesamiento y el porcentaje de exactitud. Por lo tanto, según la recomendación de *Darknet*, se selecciona el modelo *Darknet53* (30).

Segundo, se descarga el archivo de pesos y el de configuración del modelo seleccionado.

Tercero, se crea el archivo *plaga.names* que contiene únicamente los nombres de las clases a entrenar, y por tanto a detectar; el archivo tiene únicamente la palabra *plaga*, la cual corresponde al objeto de análisis según se menciona en la sección 3.3.1.

Cuarto, se crea el archivo *plaga.data* que contiene la cantidad de clases, la ubicación de los archivos *train.txt* y *test.txt* discutidos en la sección 3.3.2, la ubicación del archivo *plaga.names*, y la ubicación de la copia de seguridad que se genera cada 100 iteraciones y el archivo resultante al terminar el entrenamiento; se evidencia el contenido en la *Figura 27*.

Figura 27. Archivo de datos para el entrenamiento con *YOLOV3*

```
classes= 1
train  = Yolo-Annotation-Tool-New/plaga-train.txt
valid  = Yolo-Annotation-Tool-New/plaga-test.txt
names = data/plaga-obj.names
backup = backup/
```

Fuente: Elaboración propia.

Quinto, se crea el archivo *plaga.cfg*, el cual contiene la información de la configuración del sistema; es decir, contiene la información de la cantidad de clases, la resolución de las imágenes a entrenar, la cantidad de capas, el número de iteraciones, y los detalles de cada capa convolucional. En la *Figura 28* se visualiza la parte esencial del contenido del archivo utilizado en el proyecto.

Figura 28. Archivo de configuración para el entrenamiento con *YOLOV3*

```
[net]
# Training
batch=64
subdivisions=64
width=416
height=416

max_batches = 2000
policy=steps
steps=1600,1800
```

Fuente: Elaboración propia.

Por último, se ejecuta el proceso de entrenamiento con el sistema *Darknet*, dejando como parámetros los archivos determinados previamente. El código necesario para arrancar el proceso se ejecuta en una consola de comandos dentro de la carpeta raíz de *Darknet*; ver *Figura 29.*

Figura 29. Código de ejecución del entrenamiento con *YOLOV3*

```
darknet.exe detector train data/plaga-obj.data plaga-yolov3.cfg darknet53.conv.74
```

Fuente: Elaboración propia.

Luego de la ejecución, el proceso de entrenamiento se demora aproximadamente 24 horas seguidas utilizando la *GPU* y la computadora mencionada al inicio del capítulo. Al terminar el proceso, se obtiene un archivo llamado *plaga-final.weights,* que es el archivo principal, que contiene los pesos de distribución probabilística para la detección del objeto de análisis.

4.2.1.2. Implementación

Se implementa el entrenamiento realizado utilizando un algoritmo de elaboración propia basado en el pseudocódigo que se expresa en el *Cuadro 11* a continuación:

Cuadro 11. Pseudocódigo implementación de *YOLOV3*

Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Cargar archivo de configuración: *plaga.cfg*
4	Cargar archivo de pesos: *plaga-final.weights*
5	Cargar archivo de nombres: *plaga.names*
6	Cargar la imagen
7	Cargar la arquitectura de la red
8	Extraer características por la cantidad de capas convolucionales
9	Predecir clase de objeto por las características
10	Marcar un área de caja para la clase de objeto
11	Predecir cajas de objeto por secciones de pixeles
12	Marcar un mapa de probabilidades de clase
13	Definir umbral de intersección sobre la unión: 0.5
14	Definir umbral de supresión no máxima: 0.5
15	Definir umbral de confianza de objeto
16	Detección visual de la plaga

Fuente: Elaboración propia

De esta forma, se cargan diferentes imágenes de la base de datos del capítulo 3, en las cuales en todas hay pájaros, y el sistema no es entrenado con estas; obteniendo los resultados mostrados para 16 imágenes en la *Figura 30*:

Figura 30. Resultados del método entrenado *YOLOV3*

Fuente: Elaboración propia.

En la *Figura 30* se evidencia que en 5 imágenes no se detectan pájaros, por lo tanto, se determina un margen de error del *31,25%*, con un tiempo de ejecución promedio de 0.110625 segundos, según se observa en la *Figura 31.*

Figura 31. Tiempos de ejecución del método entrenado *YOLOV3*

```
image 1/16 data\samples\plaga-36.jpg: El tiempo de ejecución es: 0.898876428604126 segundos
image 2/16 data\samples\plaga2-14.jpg: El tiempo de ejecución es: 0.06278562545776367 segundos
image 3/16 data\samples\plaga2-4.jpg: El tiempo de ejecución es: 0.06278467178344727 segundos
image 4/16 data\samples\plaga2-55.jpg: El tiempo de ejecución es: 0.05584907531738281 segundos
image 5/16 data\samples\plaga3-23.jpg: El tiempo de ejecución es: 0.06378769874572754 segundos
image 6/16 data\samples\plaga3-24.jpg: El tiempo de ejecución es: 0.06283164024353027 segundos
image 7/16 data\samples\plaga3-35.jpg: El tiempo de ejecución es: 0.06081700325012207 segundos
image 8/16 data\samples\plaga4-102.jpg: El tiempo de ejecución es: 0.059805870056152344 segundos
image 9/16 data\samples\plaga4-18.jpg: El tiempo de ejecución es: 0.06083512306213379 segundos
image 10/16 data\samples\plaga4-36.jpg: El tiempo de ejecución es: 0.060800790786743164 segundos
image 11/16 data\samples\plaga4-37.jpg: El tiempo de ejecución es: 0.06282925605773926 segundos
image 12/16 data\samples\plaga4-5.jpg: El tiempo de ejecución es: 0.06079721450805664 segundos
image 13/16 data\samples\plaga4-53.jpg: El tiempo de ejecución es: 0.05286097526550293 segundos
image 14/16 data\samples\plaga4-56.jpg: El tiempo de ejecución es: 0.05385589599609375 segundos
image 15/16 data\samples\plaga4-69.jpg: El tiempo de ejecución es: 0.06283187866210938 segundos
image 16/16 data\samples\plaga4-94.jpg: El tiempo de ejecución es: 0.0628046989440918 segundos
```

Fuente: Elaboración propia.

4.2.2. Algoritmo por diferencia de imágenes

Para aplicar la técnica de diferencia de imágenes en la detección de aves, es necesario seleccionar dos fotogramas no consecutivos que contienen un ave y cuyo cambio de posición es significativo, estas imágenes son tomadas manualmente de la base de datos propia expuesta en el capítulo 3.

En la *Figura 32* se exponen dos fotogramas que contienen un ave en movimiento separados por diez fotogramas de diferencia, donde en la *Figura 32 (a)* el ave está a la derecha con respecto a la *Figura 32(b)*. Para la visualización en el documento, se amplía cada fotograma y se señala la posición del ave en cada uno de los fotogramas ampliados.

Figura 32. (a) Fotograma 1 ampliado. (b) Fotograma 10 ampliado, señalando la diferencia

(a) (b)

Fuente: Elaboración propia.

Los fotogramas expuestos se cargan en el algoritmo desarrollado en la sección 4.1.1, y se obtiene el resultado que se observa en la *Figura 33,* en la cual se señala el movimiento del ave.

Figura 33. Resultado de aplicar el algoritmo por diferencia de imágenes

Fuente: Elaboración propia.

Como se observa en la *Figura 33*, se evidencia ruido en la imagen generado por el viento, lo que hace necesario realizar un preprocesamiento para resaltar los detalles y poder visualizar la diferencia de imágenes, eliminando la mayor cantidad del ruido posible.

Por lo tanto, se diseña un filtro, que corresponde a aplicar un incremento en el contraste con un *CLAHE*, el cual se determina al realizar pruebas con varios videos hasta encontrar los valores del incremento de contraste que generan mejores resultados. El algoritmo para aumentar el contraste se representa en el pseudocódigo expuesto en el *Cuadro 12*:

Cuadro 12. Pseudocódigo incremento de contraste *CLAHE*

Línea	Proceso
1	Importar librerías
2	Definir variables
3	Cargar la imagen
4	Cambiar escala de color de la imagen, de RGB a LAB
5	Aplicar el método *CLAHE* al valor L de la imagen
6	Cambiar escala de color de la imagen, de LAB a RGB
7	Detección visual del movimiento

Fuente: Elaboración propia

Ahora, se carga la *Figura 32* en el algoritmo desarrollado y se obtiene la figura siguiente:

Figura 34. Resultado de aplicar el pseudocódigo de incremento de contraste

Fuente: Elaboración propia.

Como se evidencia en la *Figura 34*, el movimiento del ave se resalta más que sin el contraste aumentado, sin embargo, se evidencia mayor ruido; por lo tanto, se aplica el filtro morfológico de erosión con un elemento estructurante cuadrado de *3x3* y se aplica una dilación para resaltar el objeto resultante, y se obtiene el resultado que se visualiza en la *Figura 35*:

Figura 35. Resultado del algoritmo por diferencia de imágenes completo

Fuente: Elaboración propia.

Para la implementación completa del algoritmo de diferencia de imágenes se representa en el pseudocódigo expuesto en el *Cuadro 13.*

Cuadro 13. Pseudocódigo implementación completa de algoritmo por diferencia de imágenes

Línea	Proceso
1	Importar librerías
2	Definir variables
3	Cargar dos fotogramas consecutivos.
4	Aplicar el incremento de contraste a ambos fotogramas
5	Restar cada píxel entre ambos fotogramas
6	Binarizar la imagen resultante
8	Aplicar filtro morfológico erosión a la imagen resultante
9	Aplicar filtro morfológico dilación a la imagen erosionada
10	Detección visual del movimiento

Fuente: Elaboración propia

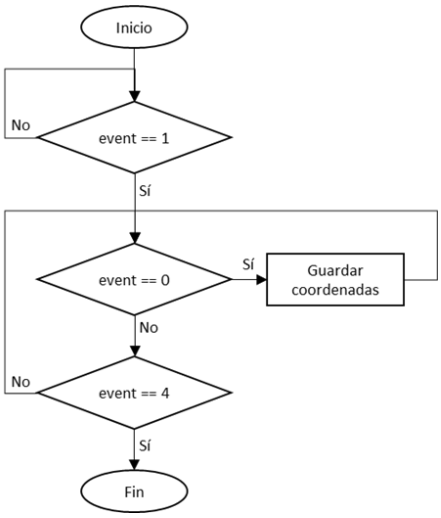
Con lo anterior, se valida el método para la obtención de la ubicación el ave a continuación.

4.2.2.1. Validación del algoritmo por diferencia de imágenes

Para validar la utilidad del algoritmo por diferencia de imágenes y del preprocesamiento requerido, es necesaria una validación completa del proceso; este proceso se debe realizar manualmente, y se debe efectuar porque esto permite conocer el porcentaje de error que se está presentando en el método de detección por diferencia de imágenes.

El proceso consiste en: trazar y graficar una línea manualmente sobre la trayectoria que tiene el ave en el video, repetir con una línea creada por el algoritmo por diferencia de imágenes para la trayectoria que este perciba del ave, comparar las dos líneas creadas y calcular el margen de error que mide la funcionalidad del método. Para esto, primero, se diseña un algoritmo para trazar la línea manualmente con base el diagrama de flujo expuesto en la *Figura 36*:

Figura 36. Diagrama de flujo trazar línea con el cursor



Fuente: Elaboración propia.

La *Figura 36* corresponde al método que permite guardar las coordenadas de cada punto de la línea trazada manualmente mientras se mantiene oprimido el botón izquierdo del ratón y se desplace el cursor sobre el video. Se almacena la coordenada *x* y la coordenada *y,* en una variable llamada *human_line***,** la cual se grafica por medio de las coordenadas, generando el resultado que se visualiza en la *Figura 37* y, posteriormente, se utiliza como línea de referencia del movimiento del ave en el cultivo.

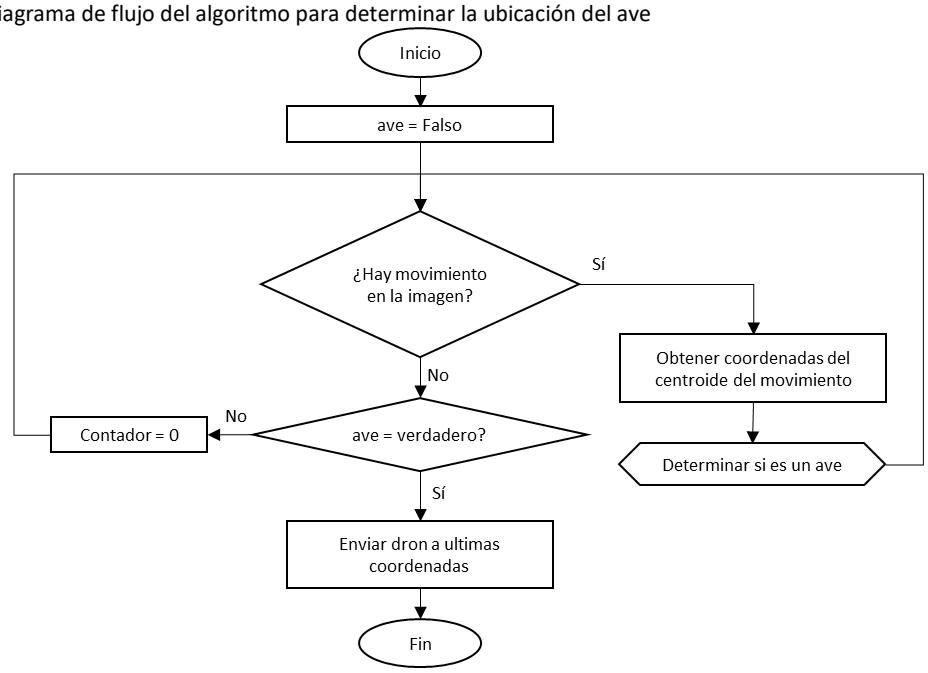
Figura 37. Ejemplo de línea hecha a mano por el humano

Fuente: Elaboración propia.

En seguida, se realiza el procedimiento para obtener la línea hecha por el ave usando el algoritmo por diferencia de imágenes; éste, identifica la coordenada *x* y la coordenada *y* del ave en cada fotograma del video, y las almacena en una variable llamada *bird_line*.

Para evitar que el ruido interfiera en la gráfica de la línea del ave, se diseña un algoritmo con base en el diagrama de flujo contenido en la *Figura 38*, con el cual se almacena las coordenadas de la posición del ave cuando éstas se encuentran a una distancia determinada del píxel anterior.

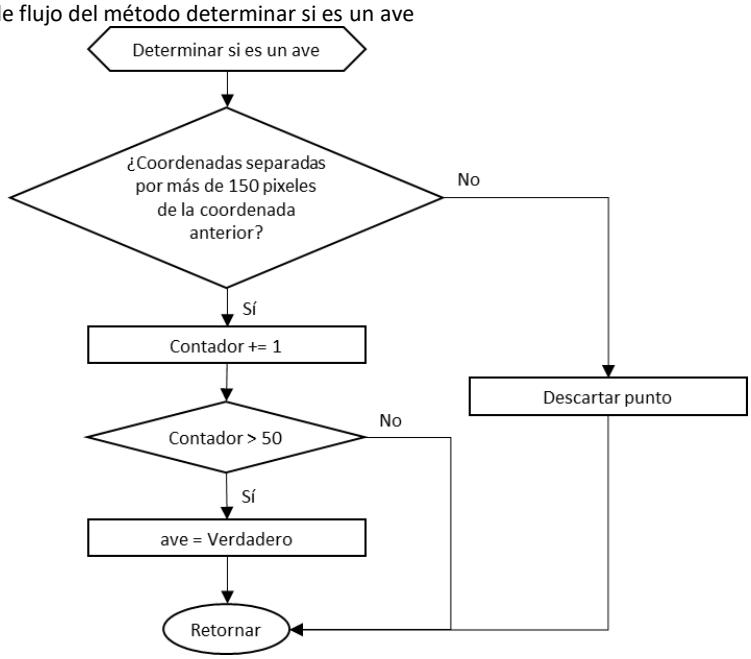
Figura 38. Diagrama de flujo del algoritmo para determinar la ubicación del ave



Fuente: Elaboración propia.

El algoritmo constantemente evalúa el movimiento en el cultivo, en caso de que se presente un movimiento que cumpla con las condiciones para ser considerado como un ave, el subprograma *Determinar si es un ave*, ver *Figura 39,* cambia el estado de la variable *ave* a verdadero, y al momento de que el movimiento se detenga se entiende que, de igual forma, el ave se ha detenido en el último píxel en el que se determina el movimiento.

Figura 39. Diagrama de flujo del método determinar si es un ave



Fuente: Elaboración propia.

Para la validación, en este caso, se considera que el punto detectado es un ave sólo si su distancia es menor en 150 pixeles, a la redonda, de la coordenada del píxel de movimiento anterior; de esta manera, el ruido en la imagen no genera una línea dispareja. Se observa la línea trazada por el algoritmo en la *Figura 40.*

Figura 40. Ejemplo de línea trazada automáticamente por el algoritmo

Fuente: Elaboración propia.

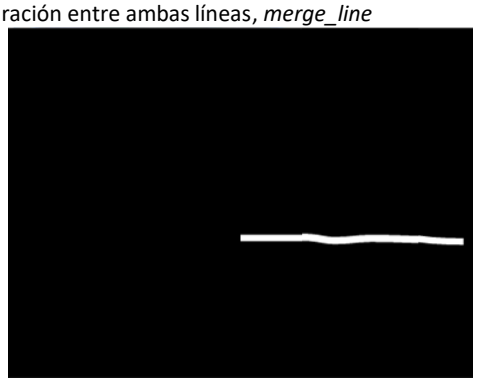
Ahora, se calcula el error entre las imágenes. Para calcular el error se convierten ambas imágenes, *human_line* y *bird_line*, a imágenes con fondo negro; se evidencia el resultado en la *Figura 41.*

Figura 41. Comparación entre la línea hecha a mano y la hecha por el algoritmo

Fuente: Elaboración propia.

Posteriormente, se compara cada coordenada de las imágenes entre sí, píxel a píxel, y cuando en ambas imágenes se encuentra un píxel blanco, significa que ambas líneas se conectan; estas coordenadas se almacenan en una nueva línea llamada *merge_line*, donde la imagen resultante son aquellos pixeles donde ambas imágenes se alinean, según se observa en la *Figura 42.*

Figura 42. Línea producto de comparación entre ambas líneas, *merge_line*



Fuente: Elaboración propia.

Se calcula el error comparando el número de píxeles en *merge_line* con el número de píxeles de la línea dibujada a mano, *human_line*; entre mayor el número de pixeles en *merge_line* significa que el algoritmo mejor traza la línea hecha por el ave.

Al momento de graficar la línea hecha por el ave se asigna un valor al grosor de los puntos y al grosor de la línea entre ellos, al aumentar este grosor la línea ocupa un mayor porcentaje dentro de la imagen. De igual forma sucede cuando la persona dibuja la línea en la imagen, ya que también tiene un grosor de pincel; cuando se aumenta este grosor, la línea ocupa un porcentaje mayor dentro de la imagen.

Siendo así, como al comparar ambas imágenes se busca que los píxeles que coinciden se mantengan en la *merge_line*, al aumentar el grosor de alguna de las dos líneas, aumenta la probabilidad de que los píxeles de esta línea coincidan con los de la otra. De esta manera, se puede variar la exactitud del algoritmo variando el grosor de la línea dibujada por el ave o de la línea hecha a mano.

Por lo tanto, se observa que, para garantizar una exactitud del cien por ciento, el grosor de la línea debe ser el mínimo posible, es decir, de 1 píxel. De esta manera, cada punto de una línea que coincida con la otra es debido a que sus coordenadas son exactamente las mismas; con base en la anterior observación, se determina una fórmula que indica la exactitud de la validación del algoritmo:

$$e = 100 - \frac{100 * gl}{h}$$

Donde: *e* es la exactitud de la validación del algoritmo, *gl* es el grosor de la línea en pixeles, y *h* es la altura de la imagen en pixeles.

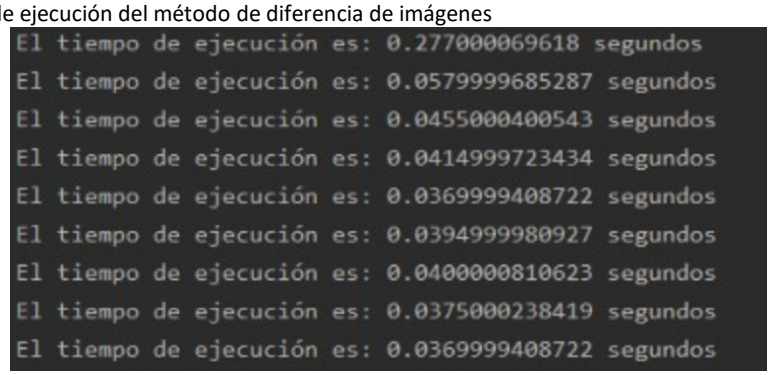
Para mantener reguladas la pruebas, estas se realizan con un grosor de línea de 5 pixeles y con las imágenes con la misma altura, correspondiente a 280 pixeles; lo que genera que el algoritmo cuente con una confiabilidad de validación, según la fórmula expuesta anteriormente, de 98.2%.

Se repite el procedimiento de validación en los múltiples videos de la base de datos expuesta en el *capítulo 3,* y se encuentra que el promedio del margen de error entre la línea humana y la línea trazada por el algoritmo es de 29,4%.

En desarrollo, se combina lo obtenido al principio de la sección 4.2.2, correspondiente al proceso de análisis de fotogramas obtenidos por las cámaras, el preprocesamiento que se le realiza a cada par

de fotogramas y la función que determina si es un ave desarrollada en esta sección. De esta forma, se obtiene el algoritmo de implementación completo, se ejecuta y se determina que el tiempo promedio de ejecución para diferentes pares de fotogramas, de acuerdo con la *Figura 43***,** es de 0.076625 segundos.

Figura 43. Tiempos de ejecución del método de diferencia de imágenes



Fuente: Elaboración propia.

4.2.3. Comparación de los métodos de implementación en el contexto real

Se pueden resaltar características comparativas entre los diferentes métodos implementados, para esto, se realiza el cuadro comparativo expuesto a continuación, con el objetivo de seleccionar el método que sea más apropiado para implementar en la detección de aves en el campo.

Cuadro 14. Comparación final de los métodos de detección

Método de implementación	Tiempo de ejecución promedio (segundos)	Fotogramas por segundo	Margen de error (porcentaje)	GPU requerida
Diferencia de imágenes	0,076625	13,05	29,4	No
YOLOV3	0,110625	9,03	31,25	Sí

Fuente: Elaboración propia.

De acuerdo con la información registrada en el *Cuadro 14,* se selecciona el método de implementación por diferencia de imágenes, debido a que este cuenta con mayor procesamiento de fotogramas por segundo, tiene un margen de error menor, y se puede implementar en computadoras sin tarjeta gráfica conservando la calidad de detección y el tiempo de ejecución, es capaz de detectar lo que se desea sin confundirlo con ruido, y es completamente mejorable y adaptable; esto, debido a que es un diseño propio elaborado por los autores del documento.

5. DESPLAZAMIENTO DEL DRON

5.1. SELECCIÓN DEL DRON

En esta sección se expone una comparación entre algunos drones comerciales para la selección del más adecuado para el proyecto. Inicialmente se realiza un filtro por costo, donde se buscan los drones más económicos del mercado de marcas reconocidas en el mundo, tales como DJI y Parrot; por lo tanto, se seleccionan los siguientes drones:

Primero, se selecciona el dron *Tello by DJI*, el cual se puede visualizar en la *Figura 44***.**

Figura 44. Fotografía de *Tello by DJI*

Fuente: DJI: https://store.dji.com.

Este cuadricóptero pequeño pesa aproximadamente 80 gramos, cuenta con una cámara sin grados de libertad y una duración promedio de la batería de 13 minutos. No cuenta con GPS por lo que su ubicación no se encuentra retroalimentada. Adicional a eso se encuentran proyectos y librerías para su control desde lenguajes de programación como Python o Go.

Segundo, se selecciona el dron *Spark by DJI*, el cual se puede visualizar en la *Figura 45.*

Figura 45. Fotografía de *Spark by DJI*

Fuente: DJI: https://store.dji.com.

Corresponde a un cuadricóptero de la marca DJI, que cuenta con un grado de libertad en la cámara para estabilizar imágenes, una distancia máxima de control de más de un kilómetro, su batería dura en promedio 16 minutos y cuenta con GPS. Adicionalmente, este dron no cuenta con una versión programable.

Segundo, se selecciona el dron Parrot Mambo, el cual se puede visualizar en la *Figura 46.*

Figura 46. Fotografía de Parrot Mambo

Fuente: Parrot: https://www.parrot.com/global.

Este dron pequeño es de la marca Parrot, el cual cuenta con una batería que dura aproximadamente 10 minutos, no cuenta con GPS ni tampoco con cámara por defecto, sin embargo, se puede conseguir como un accesorio adicional. Adicionalmente, este dron cuenta con la facilidad de ser programable.

Con base en lo mencionado anteriormente, se realiza un cuadro comparativo entre los diferentes drones a continuación:

Cuadro 15. Comparación de drones comerciales

Nombre	**Autonomía**	**Programable**	**Costo [COP]**	**GPS**	**Cámara**	**Alcance [m]**	**Peso (gramos)**
DJI Tello	13 minutos	Sí	400.000	No	Sí	100	80
DJI Spark	16 minutos	No	1.900.000	Sí	Sí	2000	300
Parrot Mambo	10 minutos	Sí	500.000	No	No	100	63

Fuente: Elaboración propia.

De acuerdo con el *Cuadro 15,* se selecciona el dron *Tello by DJI* por su buena autonomía, bajo costo, que puede ser programable, y que, a pesar de no contar con GPS se puede garantizar su ubicación en el cultivo con el diseño de un algoritmo de trayectoria y procesamiento de imágenes para este. Además, cuenta con un peso menor a 80 gramos, lo que permite que el dron sea utilizado en Colombia sin licencia, según la norma expuesta en la sección 1.5.2.

5.2. ESTACIÓN DE ATERRIZAJE

El dron escogido cuenta con una duración de 13 minutos de vuelo, y no posee de conexión para carga mientras se encuentra volando, esto hace que se torne necesario el desarrollo de un método para aumentar la autonomía del sistema. Por lo tanto, en este apartado se desarrolla una estación de aterrizaje para el reposo del dron, y se diseña un algoritmo que haga que este sea capaz de encontrar la estación.

5.2.1. Diseño inicial de la estación de aterrizaje

El dron *Tello by DJI* no cuenta con GPS, por tanto, se debe desarrollar un sistema para ubicar al dron en el cultivo y llevarlo hasta la estación mediante procesamiento de imágenes. De esta forma, se determina que la característica funcional que debe tener la estación de aterrizaje para ser detectada, debe ser un color fijo contenido en un objeto lo suficientemente grande para ser detectado por la cámara del dron; entonces, se utiliza un papel de color rosado.

La estación de aterrizaje debe contar con una plataforma amplia con el fin de que su despegue no se vea obstaculizado y que permita al dron aterrizar de manera rápida considerando un margen de error. La estación debe estar elevada para que no interfiera con las plantas que se encuentran en el suelo que pueden enredar al dron e impedir su correcto funcionamiento.

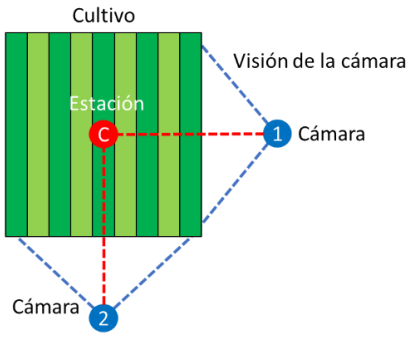
La computadora encargada de controlar el dron y de procesar las imágenes provenientes de las cámaras, se encuentra en la misma ubicación que la estación de aterrizaje; esto, para que todos los sistemas queden ubicados en un mismo sitio, lo que facilita su instalación. La computadora no cuenta con ninguna conexión física al dron sin embargo sí a las cámaras y este cableado es tomado en cuenta a la hora de ubicar la estación de aterrizaje. Con base en esto, se proceden a plantear diferentes ubicaciones para la estación de aterrizaje.

5.2.1.1. Ubicación de la estación de aterrizaje

Se debe tener en cuenta que se ubica la computadora junto a la estación de aterrizaje, ya que se facilita la tarea de determinar su posición debido a que se facilita la instalación del sistema, y, que las posibles ubicaciones de la estación de aterrizaje determinan el algoritmo de desplazamiento que debe usarse para enviar al dron a la ubicación de las aves en el cultivo.

Ahora, en primer lugar, se plantea la ubicación en el centro del área a proteger; este sistema consiste en ubicar la estación de aterrizaje en el centro del área de protección, dentro del cultivo, de manera que el dron inicie su recorrido a la menor distancia posible de cualquier zona del cultivo. En la *Figura 47* se muestra la ubicación de la estación de aterrizaje en el centro del área de protección.

Figura 47. Ubicación de la estación en el centro del área a proteger



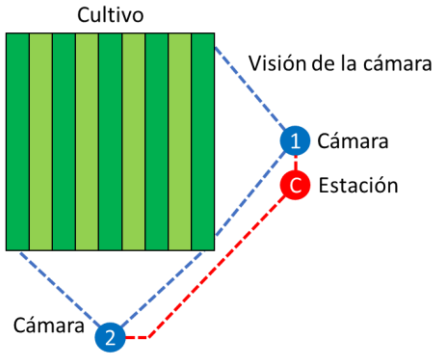
Fuente: Elaboración propia.

Con la estación de aterrizaje en el centro del cultivo la distancia a recorrer por parte del dron, para cualquier punto en el cultivo, es la mínima posible, sin embargo, incrementa el cableado necesario

para conectar las cámaras a la computadora, además de requerir intervención directa en el cultivo, lo cual no es bien visto por los agrónomos.

Por lo tanto, en segundo lugar, se plantea la ubicación junto a una de las cámaras; este sistema consiste en ubicar la estación de aterrizaje junto a una de las cámaras, de manera que el cableado a una de las cámaras es el menor posible, y así el dron inicie su recorrido desde uno de los laterales del área de protección. En la *Figura 48* se muestra la ubicación de la estación de aterrizaje junto a una de las cámaras.

Figura 48. Ubicación de la estación junto a una de las cámaras



Fuente: Elaboración propia.

Con base en lo anterior, se determina que la mejor posición para ubicar la estación de aterrizaje es en una de las cámaras; aunque al colocar la estación junto a alguna cámara el desplazamiento hasta el otro extremo es más lejano, se requiere menos cableado, la ubicación de los elementos del sistema facilita su instalación, y no se requiere intervención en el terreno, lo que permite una integración del sistema sin irrumpir los cultivos.

Consecuentemente, se procede a ubicar la estación de aterrizaje junto a una de las cámaras y realizar el cableado correspondiente.

5.2.1.2. Cableado de la estación de aterrizaje

Debido a que se seleccionan cámaras que permiten ser conectadas por cables USB, se utiliza una extensión USB de 10 metros, la cual permite cubrir la distancia entre la computadora y la cámara más lejana.

La conexión se realiza por los puertos USB de la computadora y se accede a las cámaras seleccionando el puerto al que se desea acceder.

Una vez se determina la ubicación de las cámaras y el cableado necesario, se procede a realizar la instalación. En la *Figura 49* se muestra el proceso de medición para la ubicación de las cámaras.

Figura 49. Medición de la distancia de las cámaras

Fuente: Elaboración propia.

Posteriormente a la medición, se ubica una estructura simple de madera a la distancia específica y se entierra; siendo esta distancia 5 metros tomados desde la esquina del cultivo. En seguida, se ubican las cámaras a 1.5 metros de altura, de tal forma que cada cámara abarque el área de implementación en el terreno. En la *Figura 50* se muestra una cámara instalada en el cultivo:

Figura 50. Cámara instalada en el cultivo

Fuente: Elaboración propia.

De esta misma forma, se realiza la instalación para la otra cámara y se ubica la estación de aterrizaje junto a esta, según se puede ver en la *Figura 51.*

Figura 51. Estación de aterrizaje

Fuente: Elaboración propia.

5.2.2. Desarrollo del algoritmo de detección de la estación de aterrizaje

El algoritmo de detección de la estación de aterrizaje se realiza mediante el uso de la librería de *OpenCV* en el lenguaje interpretado Python, las imágenes se obtienen mediante la cámara del dron haciendo uso de la librería *Tellopy*; librería que facilita la adquisición de video desde el dron hasta la computadora por medio del mismo punto de acceso *WiFi* generado por el dron.

La estación de aterrizaje, como se menciona a principio de la sección 5.1.1. cuenta con una bandera de color rosado para mayor detección por el color.

El algoritmo para detectar la estación de aterrizaje, una vez se inicia la cámara del dron, se diseña con base en el pseudocódigo siguiente.

Cuadro 16. Pseudocódigo algoritmo detección de estación de aterrizaje

Línea	Proceso
1	Guardar fotograma del video como imagen
2	Cambiar escala de color de la imagen, de RGB a HSV
3	Aplicar máscara de color a la imagen
4	Aplicar erosión a la imagen resultante
5	Aplicar dilación a la imagen erosionada
6	Detectar los contornos de la imagen dilatada
7	Determinar el contorno más grande en la imagen
8	Dibujar el contorno en la imagen resultante

Fuente: Elaboración propia

Con lo anterior, se detecta la estación de aterrizaje, ver *Figura 52,* y se procede a desarrollar el algoritmo de desplazamiento del dron a esta.

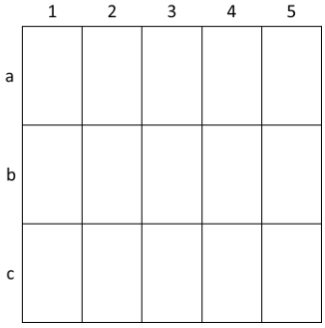
Figura 52. Detección de la estación de aterrizaje

Fuente: Elaboración propia

5.2.3. Desarrollo del algoritmo de desplazamiento a la estación de aterrizaje

Se procede a dividir la imagen resultante de la sección 5.2.2, en 3 cuadrantes horizontales y en 5 cuadrantes verticales que determinan en dónde se encuentra la estación de aterrizaje. En la siguiente figura se muestra la partición de la imagen en los cuadrantes mencionados:

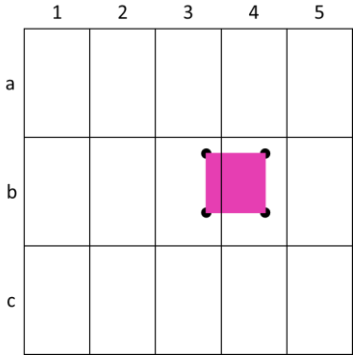
Figura 53. Cuadrantes horizontales y verticales en la imagen



Fuente: Elaboración propia.

Finalmente, se determina el movimiento del dron con base en la cantidad de esquinas que se encuentran en cada uno de los cuadrantes. Por ejemplo, al encontrarse todos los puntos en el cuadrante b2, b3 o b4, como se muestra en la *Figura 54*, significa que la estación se encuentra en el centro de la imagen y la orden que se le da al dron es avanzar en línea recta.

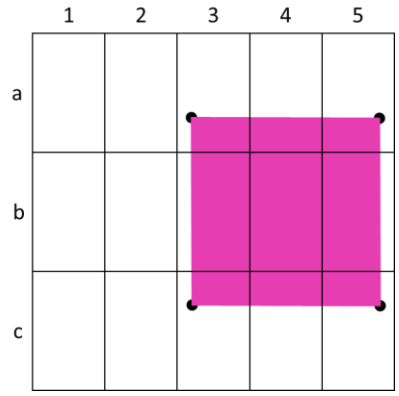
Figura 54. Todos los puntos en los cuadrantes b3 y b4



Fuente: Elaboración propia.

Una vez la estación se encuentra más cerca, la bandera ocupa una mayor porción de la imagen, como se muestra en la *Figura 55*, donde los puntos de las esquinas se reparten y queda cada punto perteneciendo a un cuadrante diferente, de tal forma que las instrucciones ahora son para centrar la imagen, desplazando el dron hacia los lados; en este caso, desplazando el dron hacia la derecha.

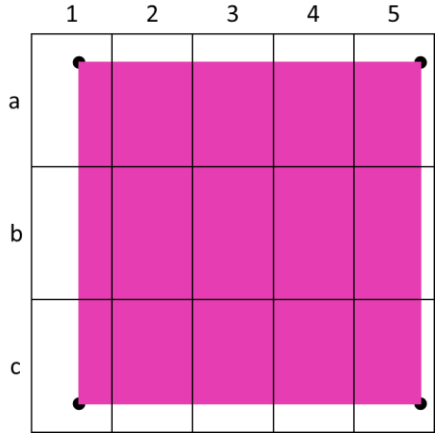
Figura 55. Puntos en cuadrantes a3, a5, c3 y c5



Fuente: Elaboración propia.

Una vez todos los puntos se encuentran en los cuadrantes a1, a5, c1 y c5, como se muestra en la *Figura 56*, se entiende que el dron ha llegado a la estación y se procede a aterrizar.

Figura 56. Puntos en cuadrantes a1, a5, c1 y c5. Se procede a aterrizar

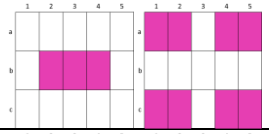
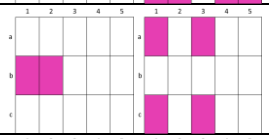
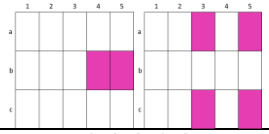
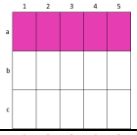
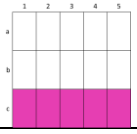


Fuente: Elaboración propia.

El algoritmo determina un total de 5 movimientos con base en la posición de las esquinas del cuadrado que genera la imagen de la bandera.

En el *Cuadro 17* se muestran algunas de las combinaciones que guían el movimiento del dron.

Cuadro 17. Algunas combinaciones que guían el movimiento del dron

Ubicación de las esquinas de la bandera	Acción del dron
	Desplazar hacia adelante
	Desplazar hacia la izquierda
	Desplazar hacia la derecha
	Desplazar hacia arriba
	Desplazar hacia abajo

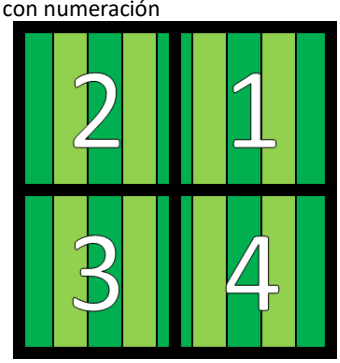
Fuente: Elaboración propia.

Con lo anterior, se concluye con el proceso del desarrollo del desplazamiento del dron a la estación de aterrizaje.

5.3. SECTORIZACIÓN DEL TERRENO

Para sectoriza el terreno, se realizan varias pruebas sobrevolando el dron manualmente hacia las aves que se observan en el cultivo, y se determina que el dron debe encontrarse a una distancia mínima de aproximadamente 1.5 metros de un ave para espantarla. Siendo así, el área cubierta por las cámaras, al ser de 100 metros cuadrados requiere ser dividida en cuatro sectores, como se muestra en la siguiente figura:

Figura 57. División del terreno en sectores con numeración



Fuente: Elaboración propia.

El algoritmo de detección de aves se adecua de tal manera que, tras determinar las coordenadas de la última ubicación del ave, se selecciona si se encuentra en la derecha o en la izquierda de la imagen. Esto se hace mediante el centroide del movimiento de las imágenes; el funcionamiento se visualiza en la *Figura 58*.

Figura 58. Ubicación del centroide del cambio entre dos imágenes

Fuente: Elaboración propia.

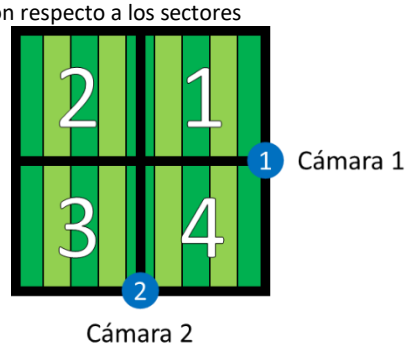
El centroide, al tratarse de un solo punto en la imagen cuyas coordenadas en el eje x se conocen, permite estipular que: si el valor de la coordenada x es mayor al de la mitad del ancho de la imagen, este movimiento se está llevando a cabo en el sector derecho de la imagen, y de lo contrario el movimiento se realiza en el sector izquierdo de la imagen. Para continuar con el ejemplo, se presentan los valores que permiten al algoritmo determinar que el movimiento se realiza a la izquierda de la imagen, como es el caso en la *Figura 59.*

Figura 59. Valores que permiten al algoritmo determinar el sector del movimiento

Fuente: Elaboración propia.

Como el método se implementa en ambas cámaras, se obtiene el sector en el que se encuentra específicamente el movimiento. En la *Figura 60* se muestra la ubicación de las cámaras con respecto a los sectores del cultivo.

Figura 60. Ubicación de las cámaras con respecto a los sectores



Fuente: Elaboración propia.

A modo de volver explicita la información, en el *Cuadro 18* se muestran los sectores resultantes de acuerdo con la posición del centroide del movimiento detectado en cada cámara.

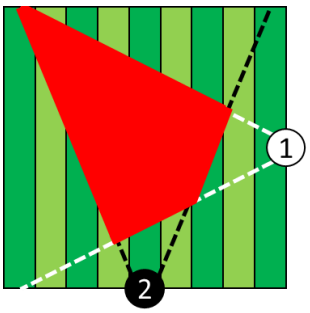
Cuadro 18. Sectorización por ubicación del ave en cada cámara

Cámara 1	Cámara 2	Sector
Derecha	Derecha	1
Derecha	Izquierda	2
Izquierda	Izquierda	3
Izquierda	Derecha	4

Fuente: Elaboración propia.

Adicionalmente a la sectorización, se menciona que las cámaras, al contar un ángulo de visión de 45° generan un área de punto ciego que permite únicamente abarcar un área de aproximadamente 20 metros cuadrados; el área visible se resalta en rojo en la *Figura 61*:

Figura 61. Área real de visión de las cámaras



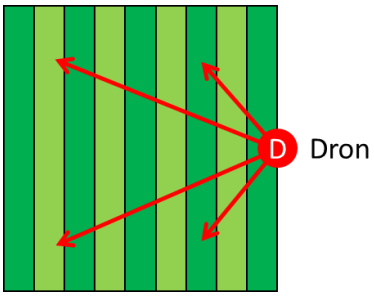
Fuente: Elaboración propia.

5.4. RUTINA DE DESPLAZAMIENTO

Debido a que el dron no cuenta con GPS, no tiene una retroalimentación de su ubicación, por lo tanto, la manera para que alcance el sector especificado es indicarle la dirección paso a paso del desplazamiento para llegar al objetivo, y el tiempo que debe durar desplazándose durante cada

paso. En la *Figura 62* se muestran las trayectorias del dron para que se dirija al centro de cada uno de los sectores:

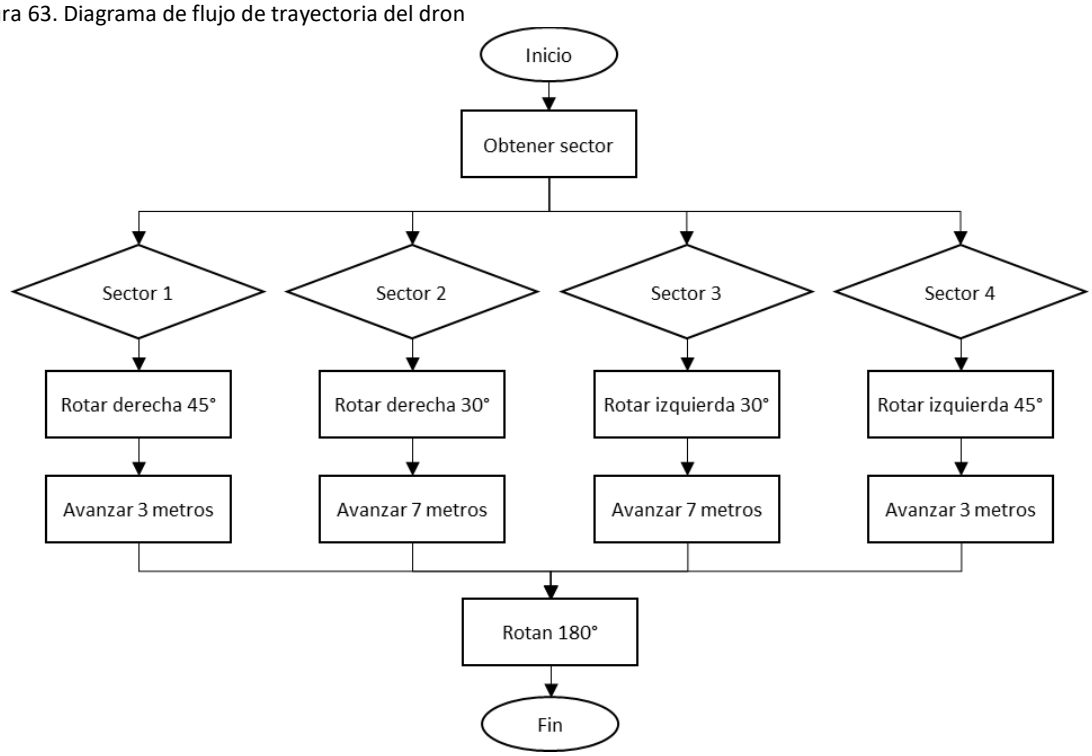
Figura 62. Trayectorias del dron a cada uno de los sectores



Fuente: Elaboración propia.

Consecuentemente, en la *Figura 63* se muestra la rutina que toma el dron dependiendo del sector en el que se encuentre el ave:

Figura 63. Diagrama de flujo de trayectoria del dron



Fuente: Elaboración propia.

Una vez el dron alcanza el objetivo, cumpliendo su rutina, realiza un giro sobre su propio eje de 180 grados antes de iniciar su cámara, de esta manera la estación se ubica en el área de visión del dron y se ejecuta el algoritmo de detección de la estación de aterrizaje.

6. SISTEMA DE ENCENDIDO REMOTO

El dron utilizado cuenta con una autonomía de 13 minutos, razón por la que los creadores del dron implementaron un sistema de apagado mientras el dron no esté en uso. Debido a esto, se hace necesaria la creación de un método capaz de encender el dron para su funcionamiento autónomo; siendo así, en el presente capítulo se realiza una breve comparación de diferentes métodos para llegar al diseño del sistema final, además se explican los pasos para la programación del módulo seleccionado, y se expone la intervención electrónica del dron para su implementación.

6.1. COMPARACIÓN DE MÉTODOS

Para decidir el método que mejor se adapta al proyecto, se toman como principio de ponderación las siguientes características fundamentales en el sistema:

Primero, el sistema debe tener la capacidad de ser activado en cualquier momento del día, es decir, con cualquier cantidad de luz solar, o en su defecto con la ausencia de ella.

Segundo, el sistema debe poder ser activado a distancias de hasta 2 metros en línea recta; según la distancia máxima de aterrizaje del dron adquirida en el capítulo 2.

Tercero, el sistema debe estar en la capacidad de ser activado en cualquier posición y ubicación del dron dentro la zona de aterrizaje.

Con base en los tres principios fundamentales con los que debe contar el sistema de encendido remoto se realiza el *Cuadro 19,* que considera la escala de 1 a 3, donde 3 es lo ideal, 2 es válido, y 1 no sirve en el proyecto, de acuerdo con la experiencia de uso y la hoja de especificaciones de cada uno de los módulos.

Cuadro 19. Comparación de módulos para el encendido remoto

Tipo de módulo	Fallo por cantidad de luz	Distancia de operación máxima	Ángulo de posición entre el receptor y el transmisor	Precio
Infrarrojo	1	3	1	3
De radiofrecuencia	3	3	2	2
WiFi	3	2	3	3
Bluetooth	3	1	3	2

Fuente: Elaboración propia.

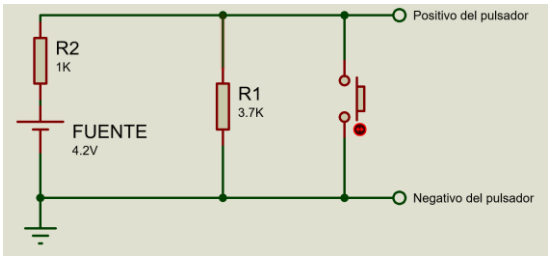
Según la ponderación realizada en el *Cuadro 19,* se descarta el módulo infrarrojo ya que no cumple con la primera característica fundamental, es decir, no tiene la capacidad de ser activado bajo cualquier cantidad de luz solar. Siguiente, se rechaza el módulo bluetooth ya que no cumple con la segunda característica fundamental, es decir, no tiene la capacidad de ser activado a distancias de hasta 2 metros, o, aunque puede ser capaz, el sistema no asegura su funcionamiento. Siguiente, aunque los módulos de radiofrecuencia y de WiFi cumplen con la tercera característica fundamental,

es decir, ambos son capaces de funcionar bajo diferentes ubicaciones entre el receptor y el emisor, debido a que el precio más bajo es el del módulo WiFi se determina que el método que mejor se adapta al proyecto corresponde al módulo WiFi, que por su tamaño y ser el módulo que se encuentra en Colombia se escoge el ESP8266.

6.2. ANÁLISIS ELECTRÓNICO Y PROGRAMACIÓN DEL MÓDULO SELECCIONADO

Para programar el módulo seleccionado primero se debe entender lo que se espera que haga, para esto se analiza el funcionamiento del encendido del dron y se realiza un diseño esquemático de la conexión electrónica del encendido del dron con base en el funcionamiento experimental del sistema electrónico; se puede observar el diseño esquemático en la *Figura 64.*

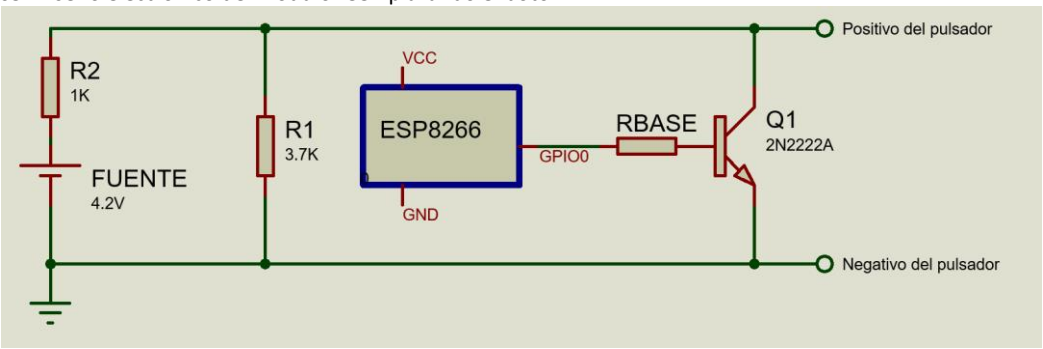
Figura 64. Diseñó esquemático del encendido del dron



Fuente: Elaboración propia.

Con base en lo anterior, se espera que el módulo WiFi ESP8266 funcione como el botón pulsador, para esto, a través de un transistor entre la conexión del pin positivo del botón del dron y la tierra de este, se plantea el diseño electrónico mostrado en la *Figura 65*.

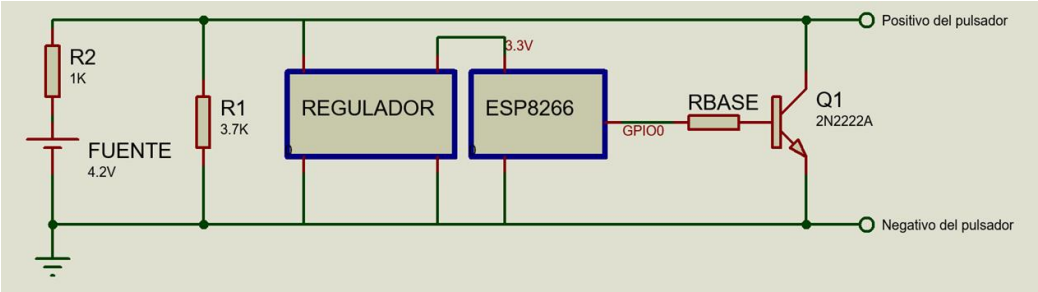
Figura 65. Diseño electrónico del módulo reemplazando el botón



Fuente: Elaboración propia.

Ahora, debido a que el módulo se alimenta con voltajes de entre 2.6V y 3.6V se acopla un regulador de voltaje de la batería del dron al módulo para garantizar que con el voltaje máximo de la batería, que corresponde a 4.2V, no lleguen más de 3.6V al módulo; por lo tanto, se acopla un regulador de voltaje de 5V a 3.3V, y se diseña nuevamente el sistema quedando como se observa en la *Figura 66.*

Figura 66. Diseño electrónico con el regulador de voltaje



Fuente: Elaboración propia.

Adicionalmente, se calcula el valor de la resistencia de base utilizando un transistor 2N2222A con beta de 220, un voltaje estable de 3.3V en la salida del módulo ESP8266, un voltaje base-emisor de 0.6V, y la corriente en el colector de 200mA; tomando en cuenta las fórmulas para los cálculos:

$$Ibase = \frac{Icolector}{beta} \qquad IBsaturación = Ibase * 2 \qquad Rbase = \frac{Ventrada - VBE}{IBsaturación}$$

Se dice que:

$$Ibase = \frac{200\ mA}{220} = 909.09\ \mu A$$

$$IBsaturación = 909.09\ \mu A * 2 = 1.8181\ mA$$

$$Rbase = \frac{3.3\ V - 0.6\ V}{1.8181\ mA} = 1.48\ K\Omega$$

Por lo tanto, a modo de aproximar el valor de la resistencia a un valor que permita mayor saturación, se dice que la resistencia de base es de $1.3\ K\Omega$.

Ahora, ya que se espera que el módulo seleccionado active una salida digital y la desactive igual que un botón pulsador, se desarrolla un código para generar un punto de acceso WiFi desde el módulo ESP8266 ubicado en el dron para la comunicación del módulo ESP8266 ubicado en la computadora, y recibe un dato por la conexión WiFi para que sea habilitada y deshabilitada la salida GPIO0.

En seguida, se desarrolla un código para el módulo ubicado en la computadora, el cual accede al punto de acceso generado por el módulo ubicado en el dron y envía un dato a través de la conexión WiFi; el pseudocódigo del algoritmo implementado es el siguiente:

Cuadro 20. Pseudocódigo encendido remoto

Línea	Proceso
1	Importar librerías
2	Declarar variables
3	Conectar al punto de acceso generado por el módulo ubicado en el dron
4	Si el módulo recibe un estado alto por el pin GPIO0
5	Enviar dato por medio de la conexión WiFi al módulo ubicado en el dron

Fuente: Elaboración propia

Se adapta el algoritmo de detección de aves para que cuando detecte un ave envíe el valor de alto al GPI0 del módulo ubicado en la computadora, y se adapta el código de llegada a la estación de aterrizaje para que cuando llegue también envíe el valor de alto al GPI0 de este módulo.

Posteriormente, se programa el módulo seleccionado siguiendo la estructura de programación expuesta en la hoja de datos del integrado ESP8266 (32). El proceso de programación para ambos módulos es el siguiente:

Primero, se descarga la plataforma de Arduino y las librerías del módulo ESP8266.

Segundo, utilizando un conversor USB a TTL, se conecta la alimentación de 3.3V y la tierra del conversor a los pines iguales del módulo.

Tercero, se conecta el pin *RX* del conversor al *TX* del módulo y se conecta el pin *TX* del conversor al *RX* del módulo.

Cuarto, se conecta el pin *GPIO0* del módulo a tierra.

Quinto, se conecta el pin *GPIO2* del módulo a 3.3V y se conecta el pin *RESET* del módulo a tierra.

Sexto, se deja al aire el pin *RESET* del módulo y se hace lo mismo con el pin *GPIO2*.

Por último, se da clic en el botón *subir* dentro de la plataforma de Arduino para que se programe el código dentro del módulo ESP8266.

Una vez terminado el procedimiento se completa con el proceso de programación del módulo, se debe realizar el mismo procedimiento para ambos módulos con los códigos correspondientes. Y con esto se procede a realizar la implementación e intervención física en el dron.

6.3. IMPLEMENTACIÓN EN EL DRON

El proceso de intervención electrónico del dron requiere de alta precisión ya que se trabaja con componentes de superficie y cualquier error puede ser fatal para el funcionamiento del dron, por lo tanto, algunas conexiones expuestas en la presente sección se realizan superficialmente. El proceso se realiza siguiendo los pasos a continuación:

Primero, se destapa el dron, según se observa en la *Figura 67.*

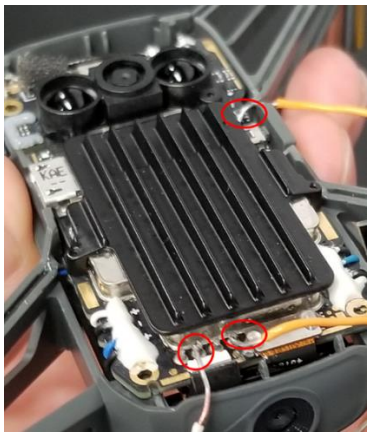
Figura 67. El dron destapado

Fuente: Elaboración propia.

Segundo, se realiza una limpieza al dron con pomada para limpiar componentes electrónicos.

Tercero, se aplica una gota de soldadura, utilizando un cautín, en los tres puntos de intervención en el dron; los puntos se visualizan en la *Figura 68.*

Figura 68. Dron destapado señalando puntos de intervención



Fuente: Elaboración propia.

Cuarto, se suelda un cable, previamente limpiado, en cada punto de intervención en el dron.

Quinto, se aplica silicona en los tres puntos para asegurar que no se suelten los cables.

Sexto, se sueldan todos los componentes en los puntos necesarios según el diseño esquemático de la *Figura 66.*

Séptimo, se aplica silicona en puntos de contacto con el dron para asegurar la estabilidad.

Los elementos se ubican de tal forma que: ningún elemento quede superior a las hélices, ningún componente quede en la sección inferior del dron ya que el disipador de calor deja de realizar su función, no se tape la cámara porque se requiere para detectar la estación de aterrizaje, y no sean necesarios acoplamientos estructurales, como un soporte o una extensión para ubicar una placa electrónica, porque le reducen la estabilidad al dron. Con esto, se completa la implementación e intervención del sistema en el dron, y queda el dron como se evidencia en la *Figura 69*.

Figura 69. El dron ya completo desde diferentes perspectivas

Fuente: Elaboración propia.

7. PRUEBAS DE FUNCIONAMIENTO

En el presente capítulo se detallan las pruebas realizadas en el contexto real del proyecto con el objetivo de determinar la viabilidad de cada parte por separado y, al integrar las partes, poder visualizar el funcionamiento teniendo en cuenta cada parte en sí misma.

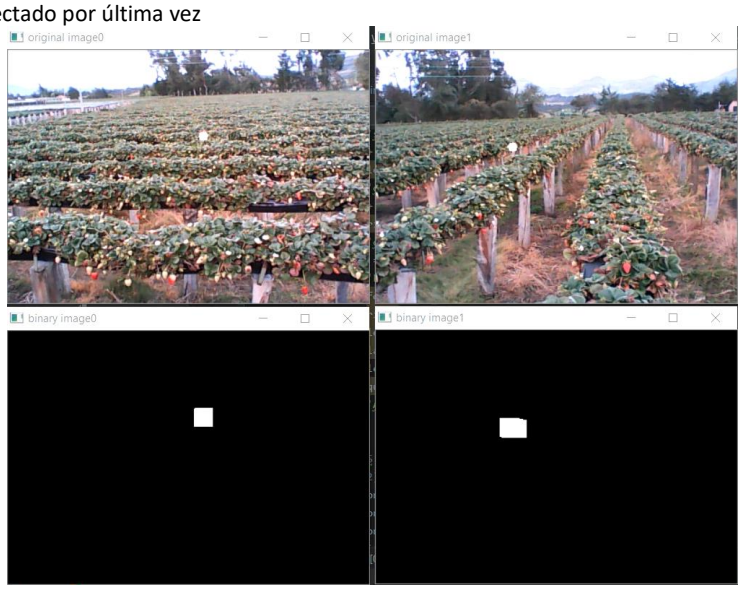
7.1. PRUEBA DEL ALGORITMO DE DETECCIÓN DE AVES

Se realizan múltiples pruebas para comprobar el funcionamiento del algoritmo de detección de aves, para esto: se inician las cámaras y se ajusta su posición, se lanza un objeto que simula ser un ave debido a que las aves no suelen acercarse al terreno cuando hay personas y, por último, se valida que el sector que detectan las cámaras corresponde con el sector donde cae el objeto. El proceso realizado se expone a continuación.

En primera instancia, se observa que las condiciones ambientales del terreno presentan cambios respecto a la visita en la que se realiza la base de datos, principalmente en la velocidad del viento, lo cual genera mayor movimiento en el área, lo que se traduce en ruido; para prevenir esto, se hace necesario aumentar el valor inferior de umbralización, lo que logra eliminar el ruido y los falsos positivos.

Ahora, se lanza el objeto de simulación, obteniendo el resultado expuesto en la *Figura 70*:

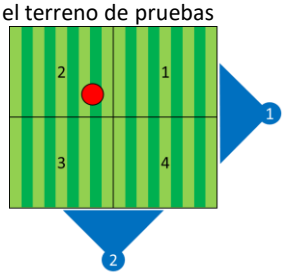
Figura 70. Objeto detectado por última vez



Fuente: Elaboración propia.

En la *Figura 70* se puede observar el momento en el que el objeto es detectado por última vez por ambas cámaras y, por lo tanto, se considera como la posición en la cual se encuentra el objeto y, a su vez, la posición a la que debe ir el dron; en este caso, al sector 2, pues la primera cámara detecta el objeto en su derecha y la segunda cámara en su derecha, como se muestra en la *Figura 71.*

Figura 71. Ubicación del objeto de análisis en el terreno de pruebas



Fuente: Elaboración propia.

Se repite el procedimiento 20 veces, y se determina que el algoritmo de detección de aves cuenta con una precisión del 90%, debido a que 18 veces los lanzamientos son identificados correctamente. Adicionalmente, se determina con un 100% de precisión el sector en el que se encuentra el ave.

7.2. PRUEBA DE LA DETECCIÓN DE LA ESTACIÓN DE ATERRIZAJE

Las pruebas de detección de la estación de aterrizaje se realizan con un papel de color rosado, cuyos valores de máscara son ajustados con respecto a los valores de pruebas previas realizado en un ambiente controlado.

En la *Figura 72* se muestra el momento en el que el dron abre la cámara e inicia el procesamiento de la estación de aterrizaje, en esta imagen el dron se encuentra en el punto más lejano a la estación de aterrizaje. Y, se observa que la estación es identificada sin ningún ruido o interferencia.

Figura 72. Detección de la bandera desde la máxima distancia

Fuente: Elaboración propia.

A medida que el dron avanza se observa la bandera desde un punto más cercano y se observa que la escala de color funciona sin ningún problema, en la *Figura 73* se observa la detección cuando el dron está en un punto más cercano de la bandera.

Figura 73. Detección de la bandera desde la distancia media

Fuente: Elaboración propia.

El algoritmo identifica sin ningún problema la estación de aterrizaje rosada, esto se valida de día, al atardecer, y con oscuridad. Se presentan falsos negativos en la imagen debido al cambio de luz natural, sin embargo, aunque el algoritmo presenta una interferencia leve no se generan fallos y el algoritmo permite al dron continuar con su proceso. En la *Figura 74* se observa el momento en el que el algoritmo identifica que la llegada del dron a la estación de aterrizaje, y, por tanto, el dron procede a aterrizar.

Figura 74. Bandera al detectar que ya se ha llegado a la estación

Fuente: Elaboración propia.

Una vez el algoritmo determina que el dron ha llegado a la estación de aterrizaje, procede a aterrizar sobre esta. Este proceso se repite 10 veces y se observa que el dron identifica y se ubica frente a la estación de aterrizaje 10 veces, lo que genera un 100% de precisión.

7.3. PRUEBA DEL DESPLAZAMIENTO DEL DRON

Se realizan múltiples pruebas para comprobar el correcto funcionamiento del algoritmo de desplazamiento del dron a cada sector, estas se realizan partiendo desde el reposo en la estación de aterrizaje.

Para comenzar, se ejecuta el desplazamiento del dron al sector 1. El dron se comporta correctamente, según se puede ver en la *Figura 75,* ya que queda dentro del área del primer sector.

Figura 75. Desplazamiento del dron al sector 1

Fuente: Elaboración propia.

Continuando, se ejecuta el desplazamiento del dron al sector 2. El dron se comporta correctamente, según se puede ver en la *Figura 76,* ya que queda dentro del área del segundo sector.

Figura 76. Desplazamiento del dron al sector 2

Fuente: Elaboración propia.

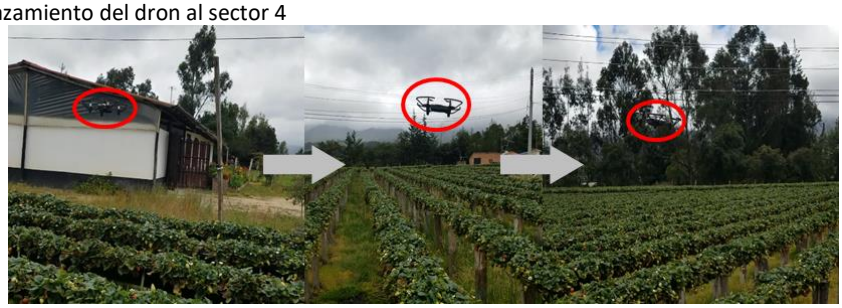
En seguida, se ejecuta el desplazamiento del dron al sector 3. El dron se comporta correctamente, según se puede ver en la *Figura 77,* ya que queda dentro del área del tercer sector.

Figura 77. Desplazamiento del dron al sector 3

Fuente: Elaboración propia.

Posteriormente, se ejecuta el desplazamiento del dron al sector 4. El dron se comporta correctamente, según se puede ver en la *Figura 78,* ya que queda dentro del área del cuarto sector.

Figura 78. Desplazamiento del dron al sector 4



Fuente: Elaboración propia.

Ahora, para validar el funcionamiento del desplazamiento del dron, se repiten las pruebas a cada sector múltiples veces. Al realizar esto, se encuentra que se genera un error constante cuando hay vientos grandes. Se revisa el registro de vuelo del dron al comunicarse con la computadora cada vez que se genera el error y, se encuentra que, cuando hay vientos grandes, se genera el error *Big wind*, que a su vez genera el estado *Unknown* en el dron; lo cual causa que, cuando se le da una orden al dron, él haga algo totalmente diferente o, en su defecto, se quede quieto.

Figura 79. Errores causados por *Big wind*

Fuente: Elaboración propia.

En las diferentes ubicaciones del dron mostradas en la *Figura 79,* se expone que el error *Big wind* causa que el dron llegue a lugares diferentes, se estrelle, o incluso lo haga perderse.

Es necesario saber cuándo el sistema falla para determinar cuándo es funcional el proyecto, por tanto, se utiliza un anemómetro para determinar la velocidad del viento cuando se genera el error, y según se evidencia en la *Figura 80,* la velocidad del viento máxima es de 11.4 km por hora.

Figura 80. Velocidad del viento con anemómetro

Fuente: Elaboración propia.

7.4. PRUEBA DEL ENCENDIDO REMOTO

Para comprobar la funcionalidad del encendido remoto en el campo se simula la detección del ave, de tal forma que cuando, manualmente, se dice que hay un pájaro, el dron debe encenderse, desplazarse y, posteriormente aterrizar y apagarse. Las pruebas cuentan con la siguiente ejecución:

Primero, en la *Figura 81* se presenta el dron apagado, se sabe ya que no alumbra el LED al lado de la cámara.

Figura 81. Dron apagado con sistema de encendido remoto implementado en campo

Fuente: Elaboración propia.

Ahora, al encenderse, como se visualiza en la *Figura 82,* alumbra el LED al lado de la cámara.

Figura 82. Dron encendido con sistema de encendido remoto implementado en campo

Fuente: Elaboración propia.

Posteriormente, el dron procede a elevarse y desplazarse, tal como se observa en la siguiente figura:

Figura 83. Proceso de ida del dron con sistema de encendido remoto

Fuente: Elaboración propia.

Consecuentemente, el dron regresa, aterriza y se apaga, según se visualiza en la *Figura 84.*

Figura 84. Proceso de regreso del dron con sistema de encendido remoto

Fuente: Elaboración propia.

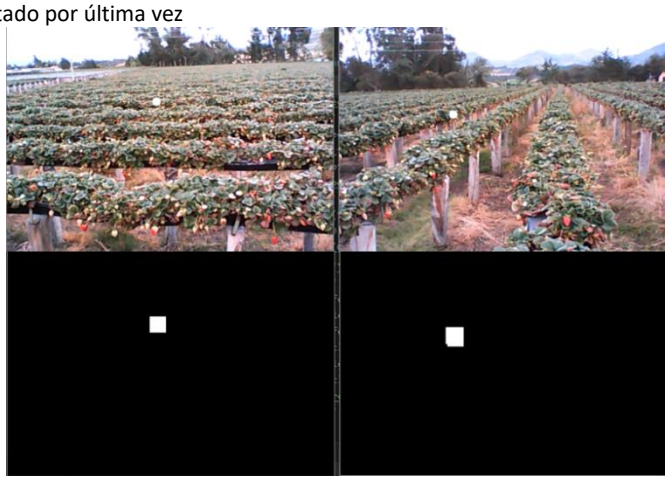
Con el fin de validar el funcionamiento del sistema de encendido remoto en el campo, se realiza múltiples veces la misma prueba a diferentes distancias entre el dron y la computadora que envía la señal de encendido y apagado; se realizan las pruebas a 1 metro, 5 metros, 10 metros, y 15 metros, donde en todos los casos se ejecuta correctamente. De tal forma, se determina que el sistema de encendido remoto al implementarlo en el campo es funcional y viable para su uso integrado con los otros algoritmos.

7.5. PRUEBA DEL SISTEMA INTEGRADO

Luego de haber evaluado cada parte del sistema por separado, se realizan pruebas con el sistema integrado. Para esto, se utiliza el código del sistema real a implementar en el campo, el cual se compone por cada parte de los algoritmos desarrollados.

Lo primero que se hace es arrancar el código, que comienza con el proceso del algoritmo de detección de aves. Se deja funcionando por aproximadamente 30 segundos para asegurarse de que no se confunda un ave con ruido, luego se procede a lanzar el objeto que simula ser un ave para hacer que el algoritmo se dirija hacia un cuadrante aleatorio. En la *Figura 85* se observa el último punto del objeto que simula ser un ave.

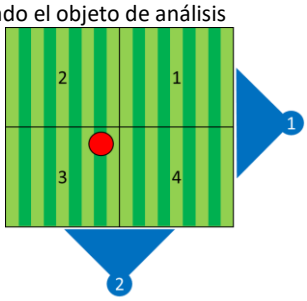
Figura 85. El objeto detectado por última vez



Fuente: Elaboración propia.

En este caso, ambas cámaras detectan el objeto de análisis, por última vez, a la izquierda de la imagen, lo que el algoritmo registra como que el ave ha aterrizado en el sector 3, según se muestra en la siguiente figura:

Figura 86. Cuadrante en el que se ha detectado el objeto de análisis



Fuente: Elaboración propia.

Consecuentemente, al detectar el objeto de análisis, se ejecuta el algoritmo de encendido remoto del dron, el cual procede a encender el dron. Una vez se encuentra encendido el dron y este genera un punto de acceso al cual se conecta la computadora, se ejecuta el algoritmo de desplazamiento del dron; por lo tanto, el dron procede a despegar, según se puede ver en la *Figura 87***,** y dirigirse al sector seleccionado, en este caso el sector 3.

Figura 87. Se observa al dron despegar desde la estación

Fuente: Elaboración propia.

Como se observa en la *Figura 88*, el dron se dirige al sector donde se encuentra el objeto de análisis, que en este caso es el sector 3.

Figura 88. Dron en el sector del objeto de análisis

Fuente: Elaboración propia.

Figura 89. Dron en el sector del objeto de análisis desde otra perspectiva

Fuente: Elaboración propia.

Posteriormente, según se observa en la *Figura 89*, el dron llega al sector 3 y se detiene cuando se encuentra a aproximadamente a *1.2 metros* del objetivo.

Consecuentemente, después de llegar al sector donde se encuentra el objeto de análisis, se ejecuta el algoritmo de detección de la estación de aterrizaje, que involucra el proceso de desplazamiento a ella. Por tanto, el dron primero procede a rotar 180 grados sobre su propio eje para dirigir su cámara hacia la estación de aterrizaje, siguiente inicia la cámara integrada en sí; este proceso tarda en promedio 24 segundos para comenzar a transmitir la imagen a la computadora y comenzar con el procesamiento. El momento en el que abre la cámara se visualiza en la siguiente figura:

Figura 90. Procesamiento de la estación de aterrizaje desde la cámara del dron

Fuente: Elaboración propia.

Como se observa en la *Figura 90*, la bandera de la estación de aterrizaje se detecta y, de esta forma, el dron procede a acercarse a la bandera hasta estar en la distancia adecuada para poder aterrizar.

Figura 91. Imagen al detectar que ya se llega a la estación

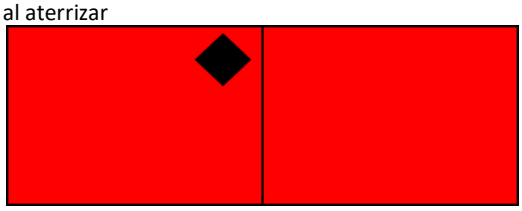
Fuente: Elaboración propia.

Una vez el dron llega a la estación de aterrizaje, la bandera se ve del tamaño de la *Figura 91*, se ejecuta la parte del algoritmo de aterrizaje, que está implícito en el algoritmo de detección de la estación de aterrizaje. Una vez el dron aterriza queda dentro de la estación de aterrizaje, según se puede ver en la *Figura 92* y en la *Figura 93*, donde la ubicación dentro de esta depende de las condiciones ambientales no controlables, tales como el viento.

Figura 92. Posición del dron al aterrizar en la estación

Fuente: Elaboración propia.

Figura 93. Diagrama de ubicación al aterrizar



Fuente: Elaboración propia.

Por último, una vez el dron aterriza por completo, se ejecuta el algoritmo de apagado remoto del dron y se procede a ejecutar el primer algoritmo nuevamente; es decir, se inician las cámaras en busca de un ave, según se puede ver en la *Figura 94*.

Figura 94. Imagen de cámaras encendidas después de aterrizar

Fuente: Elaboración propia.

Por razón de condiciones ambientales, tales como lluvia y viento, que no permiten el funcionamiento del sistema durante el resto del día al realizar las pruebas, se repite únicamente 2 veces el proceso completo, donde en las 2 veces el algoritmo completo implementado en el sector de pruebas real funciona correctamente, detectando el objeto de análisis, yendo hasta el sector donde se encuentra, regresando a la estación de aterrizaje y repitiendo el proceso.

Posteriormente, se realiza un análisis del tiempo de vuelo del dron en cada viaje para determinar su tiempo de funcionamiento antes de que sea necesario recargarlo; para esto, se registra el tiempo en cada video, donde se obtiene que el tiempo de funcionamiento en el primer video es de 157 segundos, en el segundo video es de 83 segundos y en el tercer vídeo es de 77 segundos. Por lo tanto, el tiempo promedio de funcionamiento del dron es de 106 segundos, correspondiendo al tiempo medido desde que el sistema detecta un ave y procede a encender el dron, hasta que el dron se apaga y las cámaras que vigilan el cultivo se vuelven a encender.

Debido a que el tiempo de vuelo de la batería del dron es de aproximadamente 13 minutos, el dron se encuentra en la capacidad de realizar todo el procedimiento continuamente un total de 7 veces antes de que sea necesario recargarlo.

Adicionalmente, al mismo tiempo que se realizan las pruebas de funcionamiento, se registran las aves por hora que llegan a un terreno lejano al de implementación durante un transcurso de 12 horas, de acuerdo con la siguiente tabla:

Cuadro 21. Registro de cantidad de aves por hora en un sector específico

Hora	1	2	3	4	5	6	7	8	9	10	11	12
No. de aves	3	4	3	2	6	2	5	3	2	4	2	5

Fuente: Elaboración propia.

Con la información registrada en el *Cuadro 21,* se comprende que en promedio llegan un total de 3,41 aves por hora a la sección especifica de análisis, por lo tanto, asumiendo el mismo comportamiento para la sección de implementación en funcionamiento discontinuo con un promedio de 3,41 viajes por hora, el sistema cuenta con un total de 2,15 horas de autonomía total.

8. RESULTADOS Y CONCLUSIONES

Luego de desarrollar lo expuesto en el presente documento, se generan unas conclusiones particulares y se encuentran ciertos detalles relevantes que se exponen en el presente apartado.

Según se expone en el **capítulo 2**, se implementó una red sensórica a lo largo del terreno de pruebas, que abarca un área menor a 100 metros cuadrados, que involucra la utilización de dos cámaras ubicadas, cada una, a 5 metros de una esquina del terreno de pruebas, de tal forma que se hizo posible la detección y la sectorización del objeto de análisis dentro del cultivo de fresas.

Como se presenta en el **capítulo 5**, se diseñó un algoritmo para determinar la trayectoria del dron, con base en la posición de las aves dentro del cultivo; para ello, en primer lugar, se estableció la ubicación de la estación de aterrizaje, posteriormente, se sectorizó el terreno para saber la ubicación de las aves y, finalmente, se determinó la rutina de desplazamiento del dron según cada sector en que se encuentra el ave.

Se adecuó el dron comercial *Tello by DJI* utilizado en el proyecto para que pueda ahuyentar las aves de manera autónoma; este adecuamiento consistió en la intervención del sistema electrónico del dron para que, a través del algoritmo expuesto en el **capítulo 6**, se encienda automáticamente al momento en que se detecte un ave con el uso de la red sensórica y el algoritmo de detección de aves expuestos en los **capítulos 2** y **5**, respectivamente.

Se desarrolló una estación de aterrizaje que permite el reposo del dron mientras no hay aves en el cultivo, además, se diseñó y ubicó una bandera en la posición de la estación de aterrizaje, para que con el algoritmo diseñado y expuesto dentro de la **sección 5.2**, el dron llegue a la estación de aterrizaje de manera autónoma.

Se validó la utilidad del sistema propuesto, a partir de las pruebas de funcionamiento realizadas en el **capítulo 7**, mediante las cuales se determinó que el sistema realiza de forma autónoma las acciones previstas, es decir, detecta las aves que se ubican dentro del terreno de pruebas, enciende el dron, lo dirige al sector donde se encuentra el ave, detecta la bandera de la estación de aterrizaje, regresa a la estación y aterriza. Además, el sistema es capaz de repetir este procedimiento hasta 7 veces, lo que en funcionamiento continuo representa 742 segundos, y, de acuerdo con el estimado de aves por hora en el terreno, obtenido en la **sección 7.5**, se concluye que el sistema cuenta con un total de 2,15 horas de autonomía total.

En relación con las pruebas de funcionamiento expuestas en el **capítulo 7**, se encontró lo siguiente:

- En primer lugar, según se puede ver en la **sección 7.1**, se presentó mayor ruido de lo esperado en la implementación en el campo debido a la presencia de viento en el terreno de pruebas, lo cual sólo pudo ser ajustado hasta que se implementó la prueba del algoritmo de detección de aves en el terreno. Este ajuste consistió en fijar el valor inferior de umbralización de 19 a 25 para eliminar el ruido generado por el viento.
- Según se expone en la **sección 7.2,** fue necesario ajustar el valor de la máscara para detectar el color rosado en el campo, puesto que el dron no lograba detectar la estación de aterrizaje. Estos valores fueron ajustados a: (150, 70, 70) y (180, 255, 255).
- Cuando se realizaron las pruebas del desplazamiento del dron se detectó un error por causa del viento, lo que no permitió ejecutar correctamente los algoritmos de su trayectoria en

algunos momentos; por lo tanto, se requirió del uso de un anemómetro para medir la velocidad del viento a la cual el dron generaba el error, y se determinó que a velocidades superiores a los 11,4 km por hora el dron no se desplaza apropiadamente y por lo tanto afecta la funcionalidad del proyecto. Esto se puede observar en la **sección 7.3**.

- La prueba de encendido remoto fue 100% exitosa, como se expone en la **sección 7.4**.

9. RECOMENDACIONES Y TRABAJOS FUTUROS

De acuerdo con el tiempo de funcionamiento del sistema, para mejorar la autonomía se propone, a futuro, usar el siguiente dron en la línea de DJI, correspondiente al Spark, cuyo funcionamiento es más preciso que el Tello y, además, permite la integración de un sistema de carga inalámbrica para garantizar que, mientras que el dron se encuentre en la estación de aterrizaje esté cargando.

En consideración al error causado por el viento, se propone, a futuro, utilizar un dron más robusto y resistente al viento, y un sistema hecho con base en el funcionamiento de los anemómetros para garantizar que el dron sólo despegue cuando la velocidad del viento sea inferior a la que genera fallos en el desplazamiento de este.

Para incrementar la zona de implementación dentro del cultivo, se recomienda utilizar cámaras con mayor ángulo focal y, a su vez, si se quiere mejorar la calidad de detección se recomienda utilizar cámaras con mayor resolución.

REFERENCIAS BIBLIOGRÁFICAS

(1) FAO, O. d. Organización de las Naciones Unidas para la Alimentación y la Agricultura. *Obtenido de sn: http://www. fao. org/home/es*, 1990.

(2) HEIDENREICH, Cathy. Bye Birdie–Bird Management Strategies for Small Fruit. *Cornell University. Récupéré de http://www. fruit. cornell. edu/berry/ipm/ipmpdfs/byebyebirdiesmallfruit. pdf*, 2007.

(3) CANAVELLI, S. Consideraciones de manejo para disminuir los daños por aves en girasol. *Información técnica de cultivos de verano: Campaña*, 2010, p. 175-190.

(4) PARODI, Patricio. Dos compuestos químicos como repelentes de pájaros en 'trigo maduro. 1971.

(5) CANAVELLI, Sonia B.; ARAMBURÚ, Rosana; ZACCAGNINI, María Elena. Aspectos a considerar para disminuir los conflictos originados por los daños de la cotorra (Myiopsitta monachus) en cultivos agrícolas. *El hornero*, 2012, vol. 27, p. 89-101.

(6) CASLER, C.; RIVERO, A.; LIRA, J. Los patos Dendrocygna como causantes de daños en los cultivos de arroz en Venezuela (Aves: Anatidae). *Mem. Soc. Cienc. Nat. La Salle*, 1981, vol. 41, p. 105-115.

(7) GUERRERO ARENAS, Felipe Tomás Guillermo; RAMÍREZ KOU, Jesús Armando. Sistema emisor de audio controlado orientado a espantar aves intrusas. 2016.

(8) TUBARO, Pablo Luis. Bioacústica aplicada a la sistemática, conservación y manejo de poblaciones naturales de aves. *Etología*, 1999, vol. 7, no 3, p. 19-32.

(9) GUERRERO ARENAS, Felipe Tomás Guillermo; RAMÍREZ KOU, Jesús Armando. Sistema emisor de audio controlado orientado a espantar aves intrusas. 2016.

(10) RODRÍGUEZ AYALA, Juan Francisco. Diseño y construcción de un sistema electrónico de ahuyentamiento de aves por medio de recursos sonoros y visuales para la protección de campos de cultivo. 2009.

(11) CORDERO, Aníbal Terrones; TORRES, Yolanda Sánchez. Análisis de la rentabilidad económica de la producción de jitomate bajo invernadero En acaxochitlán, Hidalgo. *Revista Mexicana de Agronegocios*, 2011, vol. 29, p. 752-761.

(12) GRIMM, Brian A., et al. Autonomous unmanned aerial vehicle system for controlling pest bird population in vineyards. En *ASME 2012 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers Digital Collection, 2012. p. 499-505.

(13) PERALTA PINTO, Juan Camilo, et al. Vinculación de la Aviación no tripulada a la Aviación convencional.

(14) ZAVALAGA, C. Índices para el inicio y cierre de las campañas de extracción de guano en la RNSIIPG (Especial atención a los aspectos reproductivos de las tres especies de aves guaneras y considerando como caso de estudio a la Isla Guañape Sur). *Informe técnico Proyecto GEF Humboldt–UNDP, Lima*, 2015.

(15) FOLKERTSMA, Gerrit Adriaan, et al. Robird: a robotic bird of prey. *IEEE robotics & automation magazine*, 2017, vol. 24, no 3, p. 22-29.

(16) CALVO GARCÍA, Javier, et al. *Smart Agro Visualization Tool*. 2017. Tesis de Licenciatura.

(17) MENESES, Viviana Alexandra Berrío; TÉLLEZ, Jemay Mosquera; VELASQUEZ, Diego Fernando Alzate. Uso de drones para el análisis de imágenes multiespectrales en agricultura de precisión. *@ limentech, Ciencia y Tecnología Alimentaria*, 2015, vol. 13, no 1.

(18) GONZÁLEZ, Adrián, et al. Drones aplicados a la agricultura de precisión. *Publicaciones e Investigación*, 2016, vol. 10, p. 23-37.

(19) ELORZA, Pilar Barreiro; UBIERNA, Constantino Valero. Drones en la agricultura. *Tierras de Castilla y León: Agricultura*, 2014, no 220, p. 36-42.

(20) COLOMBIANA, A. Aerocivil regula el uso de drones comerciales en Colombia.

(21) COLOMBIANA, Aerocivil. Requisitos generales de aeronavegabilidad y operaciones para rpas en Colombia. *Circular Reglamentaria*, 2015, no 002, p. 8.

(22) MANIVANNAN. Yolo Annotation Tool-New. *Obtenido de sn: https://medium.com/@manivannan_data/yolo-annotation-tool-new-18c7847a2186*, 2019.

(23) VIOLA, Paul, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 2001, vol. 1, no 511-518, p. 3.

(24) HOWAL, Sadanand, et al. Object Detection for Autonomous Vehicle Using TensorFlow. En *International Conference on Intelligent Computing, Information and Control Systems*. Springer, Cham, 2019. p. 86-93.

(25) LIU, Wei, et al. Ssd: Single shot multibox detector. En *European conference on computer vision*. Springer, Cham, 2016. p. 21-37.

(26) STAVENS, David. The OpenCV library: computing optical flow. 2007.

(27) ARUNPONNUSAMY. cvlib. *Obtenido de sn: https://github.com/arunponnusamy/cvlib/tree/master/cvlib*, 2019.

(28) REDMON, Joseph, et al. You only look once: Unified, real-time object detection. En *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 779-788.

(29) REDMON, Joseph; FARHADI, Ali. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

(30) REDMON, Joseph. Darknet: Open source neural networks in c. 2013.

(31) BARRY, Daniel, et al. xYOLO: A Model For Real-Time Object Detection In Humanoid Soccer On Low-End Hardware. *arXiv preprint arXiv:1910.03159*, 2019.

(32) DATASHEET, ESP8266. ESP8266EX Datasheet. *Espr. Syst. Datasheet*, 2015, p. 1-31.

(33) CHEUNG, Lucía; MEDINA, Carlos. Implementación y análisis de un detector de manos basado en visión artificial. 2013.

(34) BARBA GUAMÁN, Luis Rodrigo. *Utilización de métodos de visión artificial para pc como apoyo en la automoción*. 2015. Tesis Doctoral. ETSI_Sistemas_Infor.

(35) SOO, Sander. Object detection using Haar-cascade Classifier. *Institute of Computer Science, University of Tartu*, 2014, p. 1-12.

(36) REDMON, Joseph; FARHADI, Ali. YOLO9000: better, faster, stronger. En *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. p. 7263-7271.

(37) BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.