

# **ESCUELA POLITÉCNICA NACIONAL**

## **FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**

### **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA DETECCIÓN Y SEGUIMIENTO DE UNA PERSONA A TRAVÉS DE UN CUADRICÓPTERO DE TAMAÑO REDUCIDO EMPLEANDO VISIÓN ARTIFICIAL**

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE  
INGENIERO EN ELECTRÓNICA Y CONTROL**

**CASTELO PICHUCHO BRYAN GABRIEL**

**MOSQUERA LÓPEZ BRYAN GEOVANNY**

**DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD.**

**Quito, Enero 2022**

## **AVAL**

Certifico que el presente trabajo fue desarrollado por Bryan Gabriel Castelo Pichucho y Bryan Geovanny Mosquera López, bajo mi supervisión.

---

**Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.**  
**DIRECTOR DEL TRABAJO DE TITULACIÓN**

## **DECLARACIÓN DE AUTORÍA**

Nosotros, Bryan Gabriel Castelo Pichucho y Bryan Geovanny Mosquera López, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejamos constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

---

BRYAN GABRIEL CASTELO PICHUCHO

---

BRYAN GEOVANNY MOSQUERA LÓPEZ

## **DEDICATORIA**

Dedico este trabajo de titulación junto con todo el esfuerzo que ha conllevado para realizarlo principalmente a mis padres Galo y Marizol junto con mis hermanos Marcos, María Paula y Giovanni a quienes quiero mucho y siempre les tengo presentes en mi mente y corazón.

*Bryan Mosquera*

Dedico este trabajo a Dios y a mi familia que son el pilar más importante en mi vida, en especial a mis padres Ana y Patricio que sin su amor, esfuerzo, apoyo y paciencia no hubiese sido posible llegar a esta meta.

A mis hermanos menores Alan y Josué que junto con mis padres fueron un apoyo constante e incondicional durante mi etapa en la Poli.

*Bryan Castelo*

## **AGRADECIMIENTO**

Primero quiero agradecer al de arriba, pero al de más arriba, al todo poderoso que me ha dado la salud, la fuerza, la perseverancia y el entendimiento necesarios para seguir adelante y poder cumplir con este objetivo que me había planteado.

Quiero agradecer a mi familia por la paciencia y el cariño brindados desde mis primeros años de vida. El amor y apoyo con que me han acompañado a lo largo de todas las decisiones que he tomado y que me han llevado a ser la persona que soy ahora.

A mi abuelita Lilián, quien siempre ha estado pendiente de mi bienestar y el de mis hermanos, superando la distancia con rezos y enviándonos su amor para que cumplamos lo que nos proponíamos. Agradezco también a mis tíos, tías, primos, abuelita Martha y bisabuelita Luz María, quienes me han escuchado y acompañado durante mis pesares en la carrera, los que no han sido pocos; deseando siempre que llegue a culminar con éxito esta etapa.

A Bryan y Dio, quienes se han convertido en mis consejeros y compañeros de aventuras, risas, juegos, amores y tristezas desde que nos conocemos.

A mis amigos de la carrera, a quienes lamento no haber conocido antes, ya que habrían hecho mis días universitarios mucho más alegres y llevaderos. A mis amigos del colegio, que después de tanto tiempo, la distancia y ocupaciones que nos separan; han seguido a mi lado siempre pendientes, reuniéndose cuando es posible y alegrándose por mis victorias. Al resto de mis amigos, que pese a no pertenecer a mi carrera, fueron parte importante de mi vida.

A mi amiga y novia Náthaly Cristina por entenderme cuando estoy molesto y quererme con todos mis defectos y virtudes.

Al Doc. Patricio Cruz y al Ing. Henry Lema por su tiempo, sus enseñanzas y paciencia durante el trayecto para el desarrollo de este trabajo. A mi compañero de tesis, junto con quien conseguimos completar con éxito este proyecto que nos dejó muchas enseñanzas.

Al cantante de música urbana conocido como J Balvin y al género reggaetón, que a través de sus canciones me han hecho compañía durante las largas noches de desvelo, ya sea estudiando o festejando. Finalmente agradezco al héroe de mi infancia, quien me enseñó que un gran poder conlleva una gran responsabilidad.

*Bryan Mosquera*

En primer lugar agradezco a Dios por ser la guía durante toda mi carrera y darme la fortaleza necesaria para llegar hasta aquí, él me permitió cumplir esta meta que parecía tan lejana cuando di mis primeros pasos en la universidad, no sabía el gran reto que me esperaba.

Agradezco a toda mi familia por estar siempre a mi lado en los momentos buenos y no tan buenos pero en especial doy gracias a mis padres Ana y Patricio quienes me han brindado su amor incondicional y me han apoyado en los ratos difíciles, a pesar de no entender casi nada de mi carrera, nunca faltaron sus consejos y palabras de ánimo que me devolvían las fuerzas para continuar cuando estaba a punto de tirar la toalla, gracias por su esfuerzo y paciencia, sin ustedes nada de esto hubiese sido posible. También agradezco a mis hermanos Alan y Josué que más de una vez me han sacado una cana verde pero con su compañía y apoyo hicieron que las tareas y el estudio no se vuelva algo monótono en el hogar.

A mis amigos de universidad que siempre me han brindado su apoyo y han compartido su conocimiento conmigo haciendo que hasta las materias más difíciles se vuelvan agradables y llevaderas, con ellos compartimos muchas experiencias que harán de esta etapa algo inolvidable.

A mi amigo y compañero de tesis con el que decidimos emprender en este trabajo desde cero sin tener mucho conocimiento acerca del tema, muchas de sus ideas hicieron que este proyecto salga a flote.

A nuestro director de tesis, el Dr. Patricio Cruz quien junto con el Ing. Henry Lema han sido una guía y apoyo muy importante en esta última etapa, gracias por darnos el empujoncito necesario cuando nos encontrábamos con algún apuro y no sabíamos que hacer ni como avanzar.

Finalmente quiero agradecer a la Escuela Politécnica Nacional o Poli como nosotros le decimos de cariño, por regalarme muchas alegrías y tristezas, por las experiencias vividas dentro y fuera de sus instalaciones, por las personas que he conocido y los amigos que he hecho durante este tiempo, fue una etapa que sin duda no olvidaré y que voy a extrañar mucho.

*Bryan Castelo*

# ÍNDICE DE CONTENIDO

<b>AVAL</b> .....	<b>I</b>
<b>DECLARACIÓN DE AUTORÍA</b> .....	<b>II</b>
<b>DEDICATORIA</b> .....	<b>III</b>
<b>AGRADECIMIENTO</b> .....	<b>IV</b>
<b>ÍNDICE DE CONTENIDO</b> .....	<b>VI</b>
<b>RESUMEN</b> .....	<b>X</b>
<b>ABSTRACT</b> .....	<b>XI</b>
<b>1 INTRODUCCIÓN</b> .....	<b>1</b>
1.1 OBJETIVOS .....	3
1.2 ALCANCE .....	3
1.3 MARCO TEÓRICO .....	4
1.3.1 VEHICULOS AÉREOS NO TRIPULADOS (UAVs) .....	4
1.3.1.1 Clasificación de los UAVs .....	6
1.3.1.2 Cuadricópteros .....	8
1.3.1.3 Autopilotos.....	10
1.3.2 AVANCES EN LA SEGURIDAD BASADA EN UAVS .....	10
1.3.3 DJI RYZE TELLO .....	13
1.3.3.1 Principales ventajas.....	14
1.3.3.2 Principales desventajas.....	15
1.3.3.3 Comparación del DJI Ryze Tello con otros UAVs similares .....	15
1.3.4 VISIÓN POR COMPUTADOR .....	16
1.3.4.1 Procesamiento de Imágenes .....	17
1.3.5 ALGORITMOS DE DETECCIÓN DE OBJETOS.....	18
1.3.5.1 Haar Cascade.....	19
1.3.5.2 Redes neuronales.....	20
1.3.6 ALGORITMOS DE SEGUIMIENTO O TRACKING .....	22
1.3.6.1 TLD (Tracking – Learning – Detection) .....	22
1.3.6.2 KCF (Kernelized Correlation Filter) .....	22
1.3.7 SINGLE BOARD COMPUTER (SBC).....	23
<b>2 METODOLOGÍA</b> .....	<b>26</b>
2.1 ALGORITMO HÍBRIDO DE VISIÓN POR COMPUTADOR .....	26
2.1.1 LIBRERÍAS PRINCIPALES .....	27
2.1.1.1 Librería OpenCV.....	27

2.1.1.2	Librería djitellopy.....	28
2.1.2	ADQUISICIÓN DEL VIDEO .....	28
2.1.3	PROCESAMIENTO DEL VIDEO .....	29
2.1.4	IDENTIFICACIÓN DE OBJETOS .....	31
2.1.4.1	Discriminación de clases .....	35
2.1.4.2	Delimitación del objeto.....	36
2.1.5	FILTRO DE COLOR .....	37
2.1.6	SEGUIMIENTO DE OBJETOS .....	41
2.1.6.1	Funcionamiento del algoritmo de seguimiento .....	43
2.1.6.2	Pérdida del objeto en el algoritmo de seguimiento.....	44
2.1.7	DESARROLLO DEL ALGORITMO HÍBRIDO .....	45
2.1.7.1	Funcionamiento del algoritmo híbrido .....	45
2.1.7.2	Consideraciones durante el desarrollo del algoritmo.....	46
2.1.7.3	Descripción completa del algoritmo híbrido.....	50
2.2	ALGORITMO DE CONTROL.....	51
2.2.1	EJES DE CONTROL .....	52
2.2.2	INTERACCIÓN DEL ALGORITMO DE VISIÓN CON EL DE CONTROL....	53
2.2.3	CONSIDERACIONES EN EL CONTROL .....	54
2.3	CONEXIÓN ENTRE LA RASPBERRY PI 4 Y EL DRON .....	55
2.3.1	DESARROLLO DE LA INTERFAZ.....	56
2.3.1.1	Hardware.....	56
2.3.1.2	Software Tkinter.....	58
<b>3</b>	<b>PRUEBAS, RESULTADOS Y DISCUSIÓN.....</b>	<b>60</b>
3.1	PRUEBAS DEL ALGORITMO HÍBRIDO DE VISIÓN POR COMPUTADOR.....	60
3.1.1	ETAPA DE DETECCIÓN.....	61
3.1.1.1	Comparación entre los algoritmos de detección.....	61
3.1.1.2	Selección del algoritmo de detección.....	61
3.1.1.3	Filtro de color.....	63
3.1.2	ETAPA DE SEGUIMIENTO .....	65
3.1.2.1	Comparación entre algoritmos de seguimiento .....	65
3.1.2.2	Selección del algoritmo de seguimiento .....	67
3.2	PRUEBAS DE IMPLEMENTACIÓN DEL ALGORITMO HIBRIDO DE VISIÓN EN LA RASPBERRY PI 4 JUNTO CON EL DRON .....	67
3.2.1	VUELO DEL DRON .....	68
3.2.1.1	Pruebas de control manual .....	68
3.2.1.2	Pruebas de vuelo predefinidos.....	69
3.2.2	TRANSMISIÓN DE VIDEO.....	72



3.2.3	PRUEBAS ALGORITMO HÍBRIDO IMPLEMENTADO .....	73
3.2.3.1	Pruebas de temperatura .....	73
3.2.3.2	Pruebas de carga en RAM y CPU.....	74
3.2.3.3	Pruebas de precisión .....	75
3.2.4	ALGORITMO DE CONTROL.....	77
3.2.4.1	Calibración de controladores .....	77
3.2.4.2	Pruebas de precisión .....	80
3.3	CONDICIONES DE OPERACIÓN DEL PROTOTIPO.....	89
<b>4</b>	<b>CONCLUSIONES Y RECOMENDACIONES.....</b>	<b>91</b>
4.1	CONCLUSIONES.....	91
4.2	RECOMENDACIONES.....	93
<b>5</b>	<b>REFERENCIAS BIBLIOGRÁFICAS.....</b>	<b>94</b>
<b>ANEXOS</b>	<b>.....</b>	<b>98</b>
ANEXO A	.....	99
INDICACIONES DE USUARIO	.....	99
- Chaleco	.....	99
- Distancias	.....	99
INDICACIONES DEL PROTOTIPO	.....	100
- Alimentación	.....	100
- Dron	.....	100
ANEXO B	.....	105
PRUEBA 1	.....	105
- Ensayo 1	.....	105
- Ensayo 2	.....	106
- Ensayo 3	.....	106
- Ensayo 4	.....	107
- Ensayo 5	.....	108
PRUEBA 2	.....	108
- Ensayo 1	.....	109
- Ensayo 2	.....	109
- Ensayo 3	.....	110
- Ensayo 4	.....	111
- Ensayo 5	.....	111
- Ensayo 6	.....	112
PRUEBA 3	.....	113
- Ensayo 1	.....	113
- Ensayo 2	.....	114

- Ensayo 3 .....	114
- Ensayo 4 .....	115
- Ensayo 5 .....	116
- Ensayo 6 .....	116

## RESUMEN

Gran parte de la investigación y desarrollo de los UAVs se ha enfocado mayormente en el campo profesional, llegando a implementar vehículos muy sofisticados, versátiles y costosos que tienen la habilidad de seguir objetos, evadir obstáculos, o transportar carga, dejando a los drones pequeños relevados al campo recreativo. En este trabajo se propone el uso de un dron de tamaño reducido (DJI Ryze Tello) como herramienta principal para monitoreo; para lo cual, se ha desarrollado un sistema de bajo costo que permita detectar y seguir a una persona la cual usa un chaleco de color naranja en un ambiente no controlado, es decir en exteriores. El trabajo consiste en la implementación de un algoritmo capaz de combinar la versatilidad de una red neuronal para la detección de objetos (SSD MobileNet V2) con la rapidez de un algoritmo de tracking o seguimiento (KCF), obteniendo así un híbrido entre ambos que elija de forma automática cuál de los dos algoritmos se ejecuta para que el dron no pierda de vista a la persona que se está siguiendo. Se realizan pruebas de los algoritmos utilizándose en una SBC Raspberry Pi 4 de 8 GB para posteriormente comandar al dron. Como resultado se cuenta con un sistema portátil de bajo costo capaz de procesar el video proveniente desde la cámara del dron para elegir de manera automática a una persona que porta el chaleco distintivo y realizar el seguimiento de la misma por parte de la aeronave.

**PALABRAS CLAVE:** Visión Artificial, SSD MobileNet, KCF, DJI Ryze Tello, Detección y Seguimiento de Personas.

## **ABSTRACT**

Mainly the research and development for UAVs have focused on its professional side, reaching the implementation of very sophisticated, versatile and expensive vehicles that have the ability to follow objects, avoid obstacles, or transport cargo, leaving small drones relieved to the recreational field. In this project, the use of a small size drone (DJI Ryze Tello) is proposed as the main tool for monitoring purposes. Consequently, a low-cost system has been developed for detecting and following a person who wears a distinctive colored vest in an uncontrolled outdoor environment. The work consists of implementing an algorithm capable of combining the versatility of a neural network for object detection (SSD MobileNet V2) with the speed of a tracking algorithm (KCF). Thus, this combination proposes a hybrid between them that is able to choose automatically which algorithm is executed such that the drone does not lose the person being followed. Tests of the proposed algorithm are carried out in an 8 GB Raspberry Pi 4 SBC which then sends commands to the drone. As result, this project has implemented a low-cost portable system capable of processing the video coming from the drone to automatically select a person wearing the distinctive vest and then tracking him/her by the aircraft.

**KEYWORDS:** Artificial Vision, SSD MobileNet, KCF, DJI Ryze Tello, Person Detection and Tracking.

# 1 INTRODUCCIÓN

Durante los últimos años, los avances tecnológicos han sido cada vez más rápidos y considerables, lo que ha provocado que se abran nuevas líneas de investigación. Por ejemplo, dentro del campo de la robótica, se está trabajando activamente en el campo de los vehículos aéreos no tripulados (UAVs), debido principalmente a su gran potencial para realizar tareas que por tierra resultarían costosas, complicadas o peligrosas. Dentro de estas tareas están: el monitoreo de áreas extensas, la prevención y control de incendios, mapeo de terrenos, vigilancia y seguridad, entre otras [1].

Dentro de la clasificación de los UAVs o drones, los cuadricópteros han ganado amplio terreno gracias a su alta maniobrabilidad, relativo bajo costo y su capacidad de permanecer suspendidos en el aire [2]. Estas aeronaves se han empleado en aplicaciones aún más específicas como grabación y/o transmisión de video, principalmente por la habilidad de realizar despegue y aterrizaje vertical; maniobras que no pueden ser ejecutadas por un dron de ala fija. Estas características han hecho de los cuadricópteros la opción número uno cuando se requiere realizar el mapeo de terrenos irregulares, seguimiento de objetos desde el aire, vigilancia en zonas de difícil acceso, transporte de objetos, etc.

En el campo de la vigilancia y monitoreo, se abarcan tareas de seguimiento y detección de objetivos, mismas que tradicionalmente se las realiza con equipos y sistemas fijos, cuya principal desventaja es el rango de visión limitado. Para solucionar este problema, una posibilidad es aumentar el número de cámaras y distribuir las de una manera específica de forma que se logre cubrir un área mayor; sin embargo, esto aumentaría significativamente los costos y requeriría de algoritmos más complejos para manejar un mayor número de equipos [3].

El presente proyecto se enfoca en la necesidad de monitorear y vigilar un objeto móvil, en este caso a una persona, lo cual se puede lograr empleando un UAV. Se debe tener en cuenta que algunos modelos de drones comerciales (como el Parrot ArDrone o el DJI Mavic Air) cuentan con la función de seguimiento o tracking de objetos. Para los cuales, mediante la aplicación correspondiente, se selecciona un área en la pantalla dibujando un rectángulo sobre el objeto que se desea realizar la acción de seguimiento y el controlador de vuelo del UAV toma acciones para que el objeto no salga de su campo de visión [4]. Sin embargo, en caso de que, a pesar de estas acciones, el objetivo seleccionado sale del campo de visión, surge el problema de que el sistema no tiene la capacidad de buscarlo y reconocerlo nuevamente de manera automática para poder retomar el seguimiento. Esto se debe principalmente a que el controlador a bordo no tiene la suficiente capacidad computacional

para realizar este proceso automático de reconocimiento y así tomar una acción de manera oportuna. Una posible solución es propuesta e implementada en el presente proyecto, la cual consiste en realizar la fusión de un algoritmo de seguimiento con uno que permita la identificación (detección) de objetos para lograr que el dron pueda recuperarse cuando el objetivo desaparece de su campo de visión, sea esto por oclusión que es la pérdida del objeto debido a que otro se interpone entre este y la cámara, o por salir del área de escena.

Los algoritmos de detección, por ejemplo, basados en redes neuronales, son capaces de reconocer uno o varios objetos específicos de entre otros procesando la imagen adquirida desde una cámara y analizando su forma, textura, color, tamaño, etc. Esto se lo realiza mediante cálculos matemáticos por lo que generalmente requieren de una gran capacidad computacional. Los algoritmos de seguimiento, en cambio, se basan en el reconocimiento de píxeles, para ello el usuario debe encerrar el objeto a seguir dentro de un recuadro; entonces el algoritmo buscará esos mismos píxeles en toda la secuencia de imágenes y los encerrará dentro del recuadro. Por lo que, comparados con los algoritmos de detección, la capacidad computacional necesaria para realizarlo no es muy elevada [5].

Uno de los requerimientos adicionales de este proyecto es que el procesamiento de las imágenes debe ser realizado por un sistema que sea portátil para que en un futuro pueda ser colocado a bordo de un dron. Para esto se deberán implementar los algoritmos correspondientes en una SBC (single board computer, por sus siglas en inglés) o computador de placa reducida que disponga de su respectiva batería. Esto implica que los recursos computacionales son limitados, por lo que se ve la necesidad de realizar una mejora utilizando estos dos tipos de algoritmos, detección y seguimiento, para no sobrecargar a la SBC y que la imagen pueda ser procesada rápidamente.

Por esto, el presente proyecto se enfoca en la implementación de un sistema que permita al cuadricóptero DJI Ryze Tello [6] realizar la detección y seguimiento de una persona en exteriores. La captura de imágenes se realizará a través de la cámara con la que viene equipada este dron y el procesamiento se realizará en una Raspberry Pi 4 que no estará montada en la aeronave debido a la baja capacidad de carga que tiene este cuadricóptero. Este trabajo busca implementar un algoritmo que permita fusionar la capacidad de identificación de objetos de una red neuronal y la rapidez de un algoritmo de seguimiento, para de esta forma aprovechar los recursos que tiene una SBC, que al ser sistemas embebidos pequeños y de prestaciones restringidas, son limitados para el procesamiento de imágenes.

## 1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es:

Diseñar e implementar un sistema para la detección y seguimiento de una persona a través de un cuadricóptero de tamaño reducido empleando visión artificial.

Los objetivos específicos del Proyecto Técnico son:

- Realizar la revisión bibliográfica correspondiente a visión por computador, algoritmos de detección y seguimiento, haciendo énfasis en los empleados en aplicaciones móviles; así como de información sobre las características técnicas del dron DJI Rize Tello y la SBC Raspberry Pi 4.
- Seleccionar un algoritmo de seguimiento y uno de detección para fusionarlos entre sí y crear un algoritmo híbrido de visión que pueda ser implementado en una Raspberry Pi 4.
- Diseñar un algoritmo de control que permita enviar comandos de vuelo al autopiloto del DJI Rize Tello en base a la referencia generada por el algoritmo de visión para el seguimiento de una persona.
- Integrar la Raspberry, su batería y demás periféricos en un chaleco que será portado por la persona a la que el dron tendrá que seguir.
- Realizar pruebas que permitan evaluar el desempeño tanto de los algoritmos como de todo el sistema en conjunto.

## 1.2 ALCANCE

- Se realiza la revisión bibliográfica de al menos dos algoritmos de seguimiento incluidos en la librería de visión artificial OpenCV, y de por lo menos dos métodos de reconocimiento de objetos para seleccionar los mejores candidatos de cada tipo que puedan ser combinados y ejecutados en una SBC con la finalidad de detectar y seguir a una persona en movimiento.
- Se buscan y analizan las características técnicas de la tarjeta SBC Raspberry Pi 4 incluyendo el procesador, capacidad en memoria, tarjeta gráfica, puertos, conexiones, integración con Python, entre otras, para conocer sus ventajas y limitaciones.

- Se realiza una recopilación sobre las características técnicas correspondientes al dron DJI Rize Tello para constatar sus ventajas, posibles aplicaciones, conexión con Python y sus limitaciones.
- Se desarrolla un algoritmo de visión que sea capaz de aprovechar las fortalezas de un algoritmo de detección y uno de tracking enfocado a identificar y seguir a una persona que lleve puesto un chaleco que tendrá un color distintivo.
- Se programa un algoritmo de control cuyas acciones serán generadas en base al algoritmo de visión y de esta forma conseguir que el dron pueda seguir a la persona en movimiento.
- Los algoritmos de visión y control son desarrollados en Python y se utiliza OpenCV para la implementación de los algoritmos correspondientes a visión por computador.
- Se realizan pruebas de vuelo a diferentes alturas para verificar el funcionamiento en conjunto del algoritmo de visión con el de control ejecutándose en una laptop, luego se migra el código a la Raspberry Pi 4 donde nuevamente se realizan las pruebas y los ajustes necesarios para que el dron pueda seguir a su objetivo.
- Se diseña una interfaz que permita al usuario interactuar con el sistema (Raspberry y dron) de manera sencilla.
- Se realizan pruebas a través de las cuales se pueda evaluar la correcta operación de todos los componentes del prototipo final.

## **1.3 MARCO TEÓRICO**

En este capítulo se aborda la parte teórica y los conceptos principales de los elementos y herramientas usados en este proyecto. Se hace una breve introducción del área principal asociada a este proyecto, se revisan las características y definiciones de los UAVs y de las herramientas computacionales usadas para el desarrollo del algoritmo planteado, profundizando en el campo de la visión artificial, así como en distintos algoritmos de seguimiento y detección disponibles y propuestos en las referencias bibliográficas relacionadas. También se describirán y analizarán las características del hardware sobre el cual se desarrolla este proyecto.

### **1.3.1 VEHICULOS AÉREOS NO TRIPULADOS (UAVS)**

Las plataformas UAVs (Unmanned Aerial Vehicles), o más comúnmente conocidas como drones, son aeronaves que no requieren un piloto humano a bordo. Estos vehículos aéreos



pueden ser de diferentes tipos (globos, aviones, helicópteros, cuadricópteros, etc.) y pueden ser guiados inalámbricamente desde tierra por una o varias personas que se encargan de enviar las instrucciones necesarias al autopiloto de forma: manual (radio control), semiautomática (rutas predefinidas) o automática (inteligencia artificial), dependiendo del propósito o tipo de aplicación en el que se lo empleará [7].

Un UAV por lo general está construido con materiales muy ligeros, lo que permite tener una gran maniobrabilidad y capacidad de carga a grandes alturas sin que se vea afectado demasiado su rendimiento energético. La mayoría de estas plataformas cuentan con equipos de gran tecnología a bordo como cámaras de alta definición, GPS y sensores de proximidad para evitar choques [8].

Entre sus principales ventajas se encuentran:

- Se los puede usar en aplicaciones que impliquen riesgo para las personas o explorar zonas de difícil acceso gracias a su transmisión de video en tiempo real.
- Permiten transportar carga útil de manera ágil y segura, a un costo muy reducido.
- Existe gran variedad de precios, tamaños, capacidad de carga, autonomía, tecnología a bordo, etc., dependiendo del uso para el que se lo requiera.
- Pueden volar cualquier día sin importar la hora porque, al ser controlados de forma remota por un operario o un computador, no están atados a la condición física de un piloto humano que necesita descansar tras largos periodos de vuelo antes de poder trabajar de nuevo.
- En la actualidad, existen drones que permiten al usuario programar su controlador, eso abre una infinidad de aplicaciones que podrían tener, desde seguir a un objeto de forma individual hasta hacer grandes formaciones en conjunto (como se pudo ver en la inauguración de los juegos olímpicos de Tokio 2020 [9]) [7].

Por otro lado, entre sus principales desventajas se encuentran:

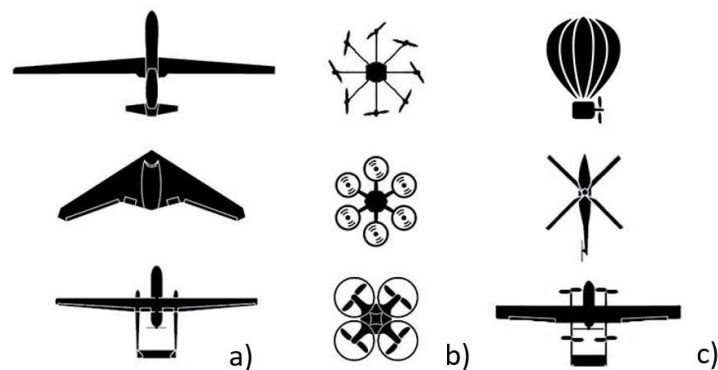
- El enlace de comunicación es muy frágil, siendo esto un inconveniente si la aplicación requiere cubrir grandes distancias de vuelo.
- Al ser vehículos aéreos pequeños, la influencia del clima (viento, lluvia, sol, sombra) es muy importante, pudiendo llegar incluso a impedir su vuelo.
- La fuente de energía, tamaño, alcance del sistema de navegación y hasta la forma de su estructura externa limitan la capacidad de vuelo. Un claro ejemplo se tiene

entre los drones de ala fija y hélices rotatorias, los primeros pueden volar largas distancias por mucho más tiempo debido a que son más aerodinámicos y usan menos recursos para funcionar, lo que significa menos consumo de energía [8].

- Al ser dispositivos tan versátiles, muchas veces son adquiridos por personas poco éticas que pueden hacer mal uso de estos como invadir espacio aéreo restringido, destrucción de propiedad pública o privada, hostigamiento a peatones, etc. [7].

### 1.3.1.1 Clasificación de los UAVs

Estas aeronaves se pueden clasificar de diferentes maneras: dependiendo de la forma de sus alas (ala fija, hélices rotatorias, híbridos), su conectividad (wifi, RF, bluetooth, etc.), su tipo de aplicación (civil o militar), entre otras. Como se puede observar en la Figura 1.1 la primera forma de clasificación mencionada es la que generalmente abarca a las demás. A continuación, se mencionan las principales diferencias entre los tipos de drones según la forma y estructura de sus alas y las aplicaciones más comunes en las que se los emplea [10].



**Figura 1.1** Tipos de drones por sus alas: a) Ala fija, b) Hélices rotatorias, c) Híbridos [11].

#### 1.3.1.1.1 Ala fija:

Este tipo de aeronave se caracteriza por tener las alas fijadas a su cuerpo, generalmente tienen forma de avión y trabajan bajo el mismo principio: sustentación por flujo de aire bajo las alas. Estos vehículos son capaces de volar grandes distancias a altas velocidades y a gran altura, sin embargo, son incapaces de realizar vuelos lentos y despegar en lugares donde la pista sea inexistente [7].

#### 1.3.1.1.2 Hélices rotatorias:

Contrario a los de ala fija, este tipo de aeronave cuenta con hélices rotatorias en su estructura, el principio de funcionamiento es igual al de un helicóptero: el empuje del aire

hacia abajo usando rotores. Esto les brinda la capacidad de poder despegar y aterrizar de forma vertical y hacer vuelos estacionarios. Comúnmente se los puede encontrar con 4, 6 y 8 rotores, lo que significa un mayor consumo de energía, bajas velocidades y alturas de vuelo [7].

En la Tabla 1.1 se realiza una breve comparación entre los aspectos más importantes que se deben tener en cuenta al momento de seleccionar el tipo de dron más adecuado para cada aplicación.

**Tabla 1.1** Comparación entre drones de ala fija y hélices rotatorias [7].

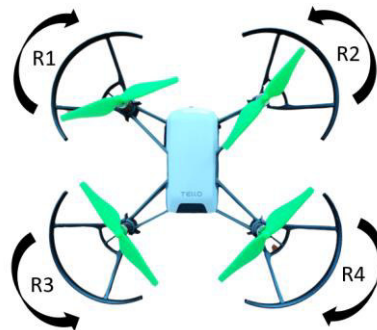
	<b>Drones de Ala Fija</b>	<b>Drones de Hélices Rotatorias</b>
<b>Forma de las alas</b>	Alas fijadas al cuerpo del dron	Hélices rotatorias
<b>Despegue/aterrizaje</b>	Necesita pista	Vertical
<b>Vuelo estacionario</b>	No	Si
<b>Velocidad máxima de vuelo</b>	Drones militares: 1000km/h Drones civiles: 80km/h	Menor a los de ala fija
<b>Altura de vuelo</b>	Alta	Baja
<b>Distancia de vuelo</b>	Larga	Corta
<b>Estabilidad</b>	Estable en comparación con el de hélices rotatorias	Cabeceo y alabeo involuntario
<b>Capacidad de adquirir datos</b>	Puede tomar fotografías de amplias zonas	Capaz de capturar imágenes de un objetivo específico a múltiples alturas y direcciones
<b>Ejemplos</b>	Sky Walker, Boeing J-UCAV x-45 A, RQ-11 Raven	DJI Tello, Parrot AR.Drone 2.0, Mavic 2 Mini

#### 1.3.1.1.3 *Híbridos:*

Los drones híbridos presentan las características de los dos tipos explicados anteriormente. Cuentan con la estructura y mecanismos necesarios para despegar y aterrizar de forma vertical sin la necesidad de una pista, como los de hélices rotatorias, y luego son capaces de propulsarse de forma horizontal logrando así cubrir grandes distancias de vuelo, como los de ala fija; sin embargo, su estructura física no necesariamente es similar. Dentro de esta clasificación se encuentran drones en forma de globo aerostático, helicópteros, aviones, etc., como se observa en la Figura 1.1.

### 1.3.1.2 Cuadricópteros

Este tipo de plataformas, también conocidas como helicópteros quadrotor, son aeronaves propulsadas por 4 motores ubicados en forma de cruz para evitar el desequilibrio del aparato. Su vuelo depende de 4 perfiles aerodinámicos (hélices) acopladas a los motores que giran a diferente velocidad y sentido para controlar el desplazamiento, elevación, rotación y torque (dos pares idénticos, uno gira en sentido horario y el otro en sentido antihorario), como se puede ver en la Figura 1.2. Esto es para contrarrestar las fuerzas de empuje y arrastre angulares generadas por la rotación de las hélices. Por lo que, si los dos pares giran a la misma velocidad, el sistema permanece en perfecto equilibrio angular, caso contrario se producen desplazamientos o rotaciones [12].

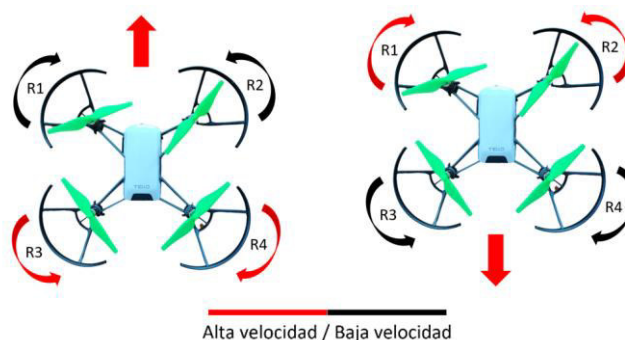


**Figura 1.2** Sentido de giro de las hélices en un cuadricóptero

#### 1.3.1.2.1 Modo de funcionamiento

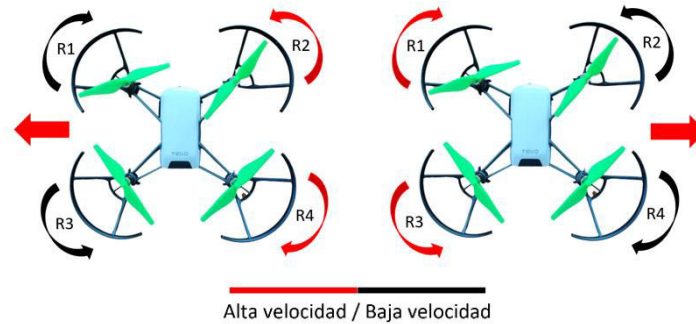
Para ciertas aplicaciones, el operario de un dron se verá en la necesidad de programar un controlador que se ajuste a su necesidad, pero antes de hacerlo, se debe tener claro los cuatro tipos de movimientos que el dron realiza para desplazarse:

*Pitching (cabeceo)*: corresponde a la inclinación del dron sobre su eje y. Permite que el UAV se desplace hacia adelante o hacia atrás aumentando la velocidad de los rotores, como se observa en la Figura 1.3 [13].



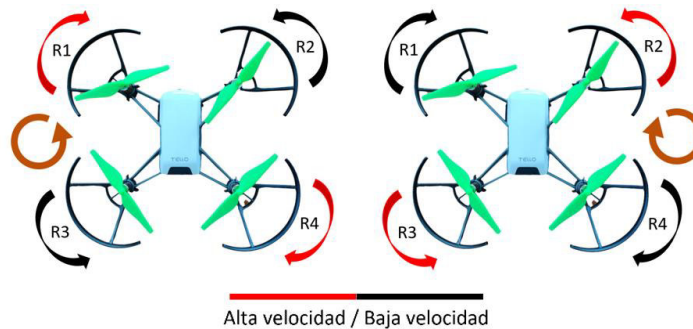
**Figura 1.3** Comportamiento de los rotores para la maniobra de pitch (cabeceo)

*Rolling (alabeo)*: corresponde a la inclinación del dron sobre su eje x. Permite que el UAV se desplace lateralmente aumentando la velocidad de los rotores, como se observa en la Figura 1.4 [13].



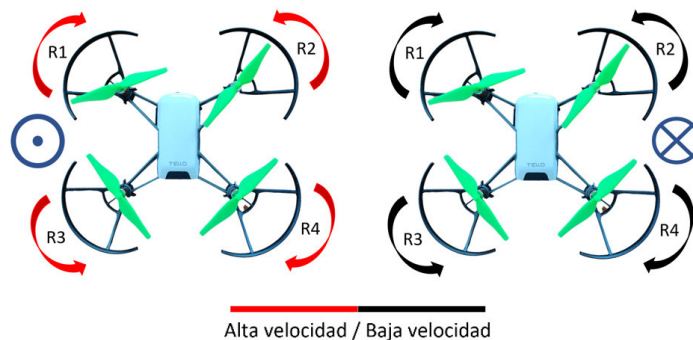
**Figura 1.4** Comportamiento de los rotores en la maniobra de roll (alabeo)

*Yaw (Guiño)*: corresponde a la rotación sobre su eje z. Permite que el UAV rote sobre su propio eje aumentando la velocidad de los rotores en pares, como se observa en la Figura 1.5 [13].



**Figura 1.5** Comportamiento de los rotores para el movimiento de yaw (guiño)

*Movimientos de ascenso y descenso*: estos se realizan subiendo o bajando la velocidad de todos los rotores al mismo tiempo como se observa en la Figura 1.6.



**Figura 1.6** Comportamiento para el ascenso (izquierda) y descenso (derecha)

### **1.3.1.3 Autopilotos**

El autopiloto o controlador de vuelo en un dron es el cerebro que se encarga de procesar los datos adquiridos desde los sistemas y sensores embebidos, con el fin de determinar cuáles son las acciones de control más adecuadas para el vehículo según el tipo de aplicación [14]. Actualmente existe una gran variedad de controladores desarrollados por empresas que muchas veces guardan “bajo llave” el acceso a su programación (sistemas cerrados). Esto pasa generalmente en drones que son dirigidos a un área recreativa (DJI, Parrot, Holy Stone, etc.), en donde el usuario es quien controla la aeronave a través de un control remoto. Por otro lado, existen controladores de código abierto (Pixhawk, Intel, ArduPilot) que permiten al usuario programar diversos tipos de plataformas aéreas a conveniencia [14] aumentando o quitando partes de su estructura (sensores, actuadores, etc.) según lo que se requiera y lograr que todo trabaje en conjunto.

### **1.3.2 AVANCES EN LA SEGURIDAD BASADA EN UAVS**

En la actualidad, el avance tecnológico en el campo de la seguridad crece muy rápidamente. El aumento de empresas militares y de seguridad privada ha permitido que la industria electrónica cuente con una fuerte inyección de capital enfocado a realizar nuevas investigaciones y desarrollar equipos que garanticen innovación y soluciones tecnológicas en estos campos. Según la multinacional Accenture, se calcula que alrededor del 70% de estas empresas realizarán fuertes inversiones en tecnología en los próximos 5 años, esto incluye a la seguridad [15]. A continuación, se describen algunas áreas de desarrollo enfocadas a la seguridad con drones:

#### *USO DE DRONES EN LA POLICÍA*

En la actualidad, las instituciones de control y seguridad pública han empezado a utilizar los drones como herramientas para distintos fines, dentro de los cuales se puede encontrar:

- Control y vigilancia de grandes aglomeraciones.

Los UAVs dedicados para este fin están dotados con sensores que son capaces de detectar anomalías en las aglomeraciones de personas, esto con el propósito de poder reaccionar ante incidentes que pueden terminar en disturbios. Se los puede ver en eventos deportivos, marchas o protestas civiles, conciertos [16].

- Investigaciones forenses o inspecciones visuales.

Para incidentes terrestres o marítimos en donde se requiere una rápida inspección del área, sin alterar la escena del suceso, antes de enviar los cuerpos de apoyo se están empleando a los drones [16].

- Operaciones especiales.

Se usan UAVs cuando el tipo de operación es de gran riesgo y así evita poner en peligro la vida de los militares o policías encargados de combatir, por ejemplo, altercados, permitiendo planear mejores estrategias de abordaje en situaciones de secuestros, operaciones terroristas, tráfico de drogas, etc. [16].

- Búsqueda y rescate.

En estos casos los drones son de mucha ayuda y más aún cuando poseen cámaras térmicas o de gran zoom, linternas, altavoces o micrófonos. Esto con el fin de poder localizar a las personas extraviadas y poder comunicarse con ellas antes de que lleguen los organismos de rescate a la escena [16].

- Disuasión de actividades ilícitas.

En el país, actualmente existe un plan piloto que plantea usar estas aeronaves como método para combatir robos a transeúntes, vehículos mal estacionados, microtráfico, entre otros delitos efectuando sobrevuelos en las zonas donde más se producen estos actos y de esta manera identificar a los posibles sospechosos y proceder con su captura [17].

#### *USO DE DRONES EN LA SEGURIDAD PRIVADA*

En el pasado era muy común encontrarse con lugares que contaban con sistemas cerrados de video en donde un vigilante se sentaba varias horas frente a un monitor a observar si había intrusos en las instalaciones. Con el tiempo esto se fue modernizando a cámaras con detección de movimiento que podían alertar al usuario cuando algún objeto se movía dentro del campo de visión, En la actualidad, con la aparición y desarrollo de los drones, el campo de la seguridad y vigilancia privada ha cambiado significativamente, ahora los sistemas de vigilancia son más seguros. La maniobrabilidad y versatilidad de los drones permite monitorear grandes áreas sin tener los problemáticos puntos ciegos. A continuación, se describen algunos usos que se dan a los UAVs dentro de este campo.

- Vigilancia automatizada de perímetros.

De la misma forma que un sistema de videovigilancia, un sistema de drones que cuenten con cámaras a bordo puede vigilar el mismo perímetro o incluso mayor que un sistema de

videocámaras fijas usando menos equipos gracias a su libertad de movimiento, por ende los puntos ciegos son casi inexistentes [16].

- Vigilancia a distancia.

Los drones son los ojos a distancia de guardias de seguridad, les permiten vigilar grandes zonas o revisar algún incidente lejos de su posición actual, sin exponer su integridad de manera innecesaria [16].

- Drones escoltas.

Permiten acompañar desde el aire el transporte de cargas sensibles o de gran valor, así como de personas que soliciten su vigilancia, esto con el fin de tener una visión más amplia de la zona y con ello poder detectar y evitar cualquier problema que pueda presentarse en el camino de manera anticipada [16].

#### *USO DE DRONES PARA LA LUCHA EN CONTRA DEL COVID-19*

Desde inicios del 2020 hasta la actualidad, el país y el mundo entero ha venido luchando en contra de un virus inédito que está acabando con la vida de una gran cantidad de personas sin importar la etnia, estatus social o ubicación geográfica. Esto ha encendido las alarmas en todo el mundo causando una inyección de grandes cantidades de dinero que es destinado a la investigación y desarrollo de técnicas que permitan frenar la propagación de esta pandemia y sus fatales consecuencias. Dentro de estas investigaciones está el uso de drones, estos pequeños y ágiles vehículos se han convertido en una herramienta extremadamente útil en esta batalla, gracias a su versatilidad y tecnología en continuo desarrollo, además de ser equipos de fácil adquisición por los organismos de control en todo el mundo. A continuación, se mencionarán algunas de las tareas que cumplen los drones en el marco de la pandemia:

- Detectar personas enfermas.

Mediante el uso de sensores infrarrojos, cámaras térmicas montadas en los drones y algoritmos especializados, se busca distinguir a las personas enfermas de entre las sanas interpretando sus acciones como toser, estornudar, su temperatura corporal, etc. para controlar el aislamiento y evitar la cantidad creciente de contagios innecesarios por descuido [18]. Hay que tener en cuenta que este no es un método a prueba de fallas para saber si una persona está contagiada, aunque sirve como base para identificar a quienes se les debe aplicar una prueba con la brevedad del caso para confirmar o descartar el contagio.



- Delivery de alimentos, medicina y material médico.

La cuarentena mundial que se impuso por la rápida propagación del virus ha provocado que varias empresas y organismos sanitarios se adapten y busquen nuevas formas para llegar a las personas que no pueden salir de sus domicilios. En países del primer mundo se ha empezado a utilizar drones para el transporte de muestras médicas, entrega de suministros de primera necesidad y alimentos a personas que, debido a la emergencia sanitaria, no pueden salir de sus viviendas ya que los UAVs son sistemas de transporte seguro y que sus tiempos de entrega son bastante rápidos [18].

- Distanciamiento social y control de masas

Debido al descuido, desconocimiento e inconciencia de muchas personas hacia las normas impuestas por los gobiernos para frenar la propagación del coronavirus, los organismos sanitarios y de control en varios países, incluido Ecuador, se encuentran implementando un sistema poco visto anteriormente, drones equipados con tecnología y medios de comunicación como altavoces y alarmas que permiten monitorear el distanciamiento social y el uso de mascarilla en lugares públicos [18]. En el país se implementó esta medida en las playas más concurridas para evitar la propagación del Covid-19 [19].

Gracias a las aplicaciones antes mencionadas se plantea el uso de un dron de bajo costo como parte de un sistema que permita ejecutar aplicaciones de monitoreo y seguridad de manera automática. El candidato ideal para esta aplicación por sus prestaciones y precio es el DJI Ryze Tello.

### **1.3.3 DJI RYZE TELLO**

Esta pequeña aeronave elaborada por la empresa china Ryze Robotics en conjunto con la prestigiosa marca DJI y los fabricantes de procesadores Intel, garantizan una experiencia de usuario impecable y un respaldo tecnológico excelente, siendo de las mejores alternativas en el rango de precio en el que se encuentra. Su diseño bien estilizado, véase la Figura 1.7, es notablemente similar a drones pequeños de otras marcas; sin embargo, se caracteriza por su excelente material de construcción y las ventajas que posee en desarrollo, control y experiencia de usuario que vienen de la mano de las empresas que lo respaldan. El uso de este cuadricóptero no tiene restricción de edad, permitiendo a niños y adultos disfrutar de esta aeronave, de manera especial a quienes quieren adentrarse en el mundo de los drones, teniendo la posibilidad de jugar y aprender gracias a su programación en bloques [6].



**Figura 1.7** DJI Rize Tello

Sus aplicaciones de control son bien desarrolladas e intuitivas y la cantidad de herramientas que existen para programarlo lo convierten en el aliado perfecto de novatos, entusiastas y estudiantes interesados en realizar investigación, brindando la posibilidad de desarrollar nuevas técnicas de control para este tipo de vehículos. Cuenta con un sistema de control de altura óptico bastante preciso. Su manejo y maniobrabilidad es fluida y responde de manera adecuada a los comandos enviados desde una estación en tierra. Su diseño pequeño, liviano y resistente en combinación con las protecciones de software y hardware le permiten volar con confianza, seguridad y mantener su integridad pese a caídas, golpes o choques [6].

#### **1.3.3.1 Principales ventajas.**

- Cuenta con una aplicación propia perteneciente al fabricante para su manejo, se encuentra disponible en la app store de IOS y la Play Store de Android bajo el nombre de Tello.
- Los controles en la aplicación móvil son bastante intuitivos, por lo que cualquier persona podría volarlo sin problema.
- Al ser un dron tan pequeño y ligero, es ideal para aplicaciones donde el espacio es reducido.
- Sus antenas Wi-Fi de 2.4GHz le permite conectarse con cualquier dispositivo que tenga la aplicación instalada sin la necesidad de cables o equipos de conexión extra.
- Su estructura resistente combinada con algoritmos de seguridad lo hacen ideal para pilotos novatos que están en proceso de aprendizaje.
- La calidad de imagen es bastante buena y su estabilización por software es la adecuada en comparación con drones del mismo rango de precios.

- Viene con distintos modos de vuelo preinstalados permitiendo seleccionar el control para principiantes o pilotos con experiencia y de esta manera mejorar la experiencia de usuario.
- Brinda al usuario la posibilidad de desarrollar aplicaciones mediante el uso de software, permitiendo la programación por bloques o usando el paquete de Tello SDK disponible de manera libre al público en general.




### **1.3.3.2 Principales desventajas.**

- Su principal desventaja es el tamaño reducido, al ser tan pequeño y liviano las ráfagas de viento lo desestabilizan o impiden su vuelo.
- Posee motores DC a diferencia de drones de gamas más altas que cuentan con motores tipo brushless.
- Al utilizar un sistema óptico para el control de altura y posición, las aplicaciones para este dron se limitan a zonas con buena iluminación.
- El tiempo de autonomía de vuelo es corto (12-13 minutos), por lo que no es recomendable alejarlo demasiado de la base en tierra porque al reducirse la carga de la batería se limita su operación y rendimiento.
- Al utilizar un sistema de conexión Wi-Fi de 2.4GHz para el control y transmisión de video, es muy susceptible a interferencias electromagnéticas produciendo lag en la comunicación.
- Su capacidad de carga es nula.
- Al cambiar de modo de vuelo a piloto experto (sport) existe una pérdida total de la estabilización de la imagen por software.

### **1.3.3.3 Comparación del DJI Ryze Tello con otros UAVs similares**

Actualmente el uso de drones está en pleno auge y el número de empresas desarrolladoras cada vez es mayor, debido a esto en el mercado existes varias marcas y modelos de UAVs con características muy similares entre sí. En la Tabla 1.2 se muestran y comparan los aspectos técnicos más importantes del DJI Rize Tello, Parrot AR.Drone 2.0 y el DJI Spark, drones con características técnicas y físicas similares, pero con costos diferentes.

**Tabla 1.2** Comparación entre drones similares disponibles en el mercado [6] [8] [20].

	<b>DJI Spark</b>	<b>Parrot AR.Drone 2.0</b>	<b>DJI Rize Tello</b>
<b>Modelo real</b>			
<b>Cámara</b>	12 megapíxeles 1080p@30fps	5 megapíxeles 720p@30fps	5 megapíxeles 720p@30fps
<b>Conectividad</b>	Wi-Fi 2.4 / 5.8 GHz	Wi-Fi 2.4GHz	Wi-Fi 2.4GHz
<b>Autonomía</b>	16 minutos	12 minutos	13 minutos
<b>Distancia de vuelo respecto a la base</b>	100m	120m	100m
<b>Velocidad Máxima</b>	14m/s	5m/s	8m/s
<b>Peso</b>	300gr	420gr	80gr
<b>Acceso al controlador</b>	Cerrado	Abierto	Abierto
<b>Precio</b>	445\$	660\$	180\$

Como se muestra en la Tabla 1.2, el DJI Rize Tello es una excelente opción de bajo costo para proyectos como el desarrollado en este trabajo de titulación.

Tomando como antecedente lo expuesto anteriormente, se plantea el uso del dron DJI Ryze Tello como una alternativa de bajo costo que podría ser comandado y guiado por un algoritmo de visión por computador para cumplir aplicaciones de monitoreo de una persona de interés.

#### **1.3.4 VISIÓN POR COMPUTADOR**

De forma general, cuando se habla de visión por computador se está haciendo referencia a una parte de la visión artificial que se encarga de procesar y analizar imágenes de forma automática tratando de emular la visión humana [21]. Para cumplir esta meta existen numerosos algoritmos que trabajan con diferentes técnicas que extraen información del mundo real a partir de imágenes o videos en línea para luego procesarlas y realizar las acciones para las cuales se lo desarrolló [22]. Esta acción puede ser: identificar formas, rostros, personas, objetos, colores, seguir objetos, realizar comparaciones y presentar resultados.

### 1.3.4.1 Procesamiento de Imágenes

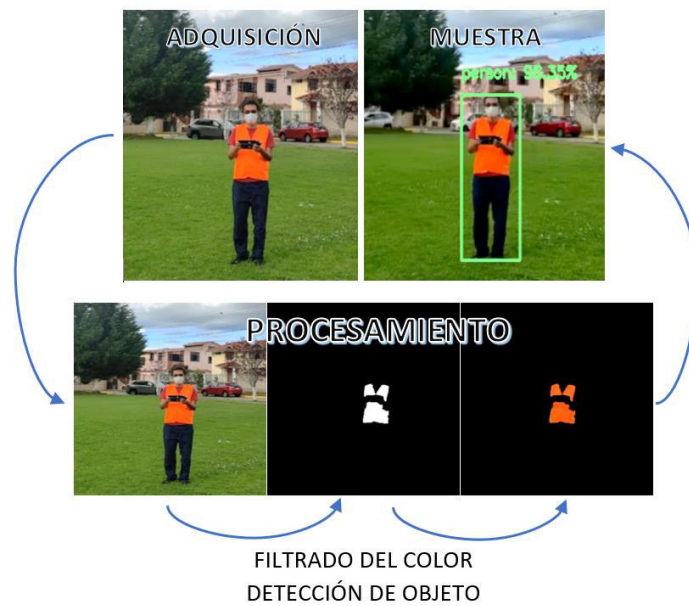
Las imágenes que se perciben son representaciones de la reflexión de luz sobre los objetos, esta acción se realiza en el ser humano por medio del sentido de la vista; mientras que, en términos tecnológicos la adquisición de imágenes se la realiza por medio de una cámara. Para la formación de imágenes digitales se tiene una interacción de los siguientes componentes: la fuente de luz, los objetos y la cámara. Esta última formada por: el sistema óptico, el sensor y el digitalizador [23]. Al utilizar una cámara digital se recolecta información en forma de imagen, esta se convierte en un número entero de píxeles con un rango limitado de valores, conformando una matriz, donde cada elemento es considerado para la conformación de los colores correspondientes a los píxeles con sus propias características, para formar la imagen en conjunto. Mediante el procesamiento de estos números es posible cargar la imagen al software o programa convirtiéndola en una matriz y de esta manera se tiene la posibilidad de manipular los datos o valores internos de este elemento para poder alterar el archivo.

Los principales problemas del procesamiento de imágenes son:

- Las imágenes pueden ser afectadas por la perspectiva pues mediante esta es posible realizar cambios en la apariencia de la imagen y por tanto puede volverse ambigua.
- Las imágenes pueden ser afectadas por factores ambientales como lo son el clima, los movimientos, reflejos, la iluminación entre otros.
- Los objetos que se encuentran en la imagen pueden estar ocluidos por otros, haciendo que sean difíciles de clasificar.

Debido a estos problemas, los sistemas encargados del procesamiento deben ser robustos o tener la capacidad de lidiar con este tipo de cambios, sean ambientales, de perspectiva o por la aparición de otros objetos en la imagen.

Para realizar el procesamiento de imágenes se debe seguir los pasos mostrados en la Figura 1.8 en donde la etapa adquirir la imagen hace referencia a las funciones necesarias para que se pueda leerla desde la fuente que se desee, sea esta una cámara, una fuente de video continuo, un disco o la red. La parte correspondiente al procesamiento propiamente dicho es la aplicación de las técnicas necesarias para conseguir lo que se requiera, por ejemplo, detectar los bordes de los objetos, determinar los colores o identificar objetos en específico.



**Figura 1.8** Ejemplo del procesamiento de imágenes

En este punto existen diferentes niveles de procesamiento que van de acuerdo con la complejidad que tenga la aplicación que se desee realizar. El procesamiento de una imagen va desde realizar un cambio de tamaño (resize), separar o unir partes de esta, realizar transposiciones, extraer colores específicos, entre otros. Como paso final es necesario mostrar el resultado de la imagen procesada, por ejemplo, con una etiqueta de identificación, con un recuadro que señale lo que se desea o resaltando de cierta manera el resultado generado [24].

En la Figura 1.8 se encuentra una imagen inicial u original en el lado superior izquierdo, misma que ha sido obtenida, por ejemplo, del disco duro y enviada al respectivo algoritmo para ser procesada. Los pasos correspondientes al procesamiento ocurren en las fotografías inferiores donde se la transforma primero en escala de grises, para posteriormente extraer los bordes, y finalmente el algoritmo analiza los datos obtenidos anteriormente para dar un resultado, obteniendo de esta manera la imagen superior derecha. En este ejemplo, se presenta la figura original dibujando rectángulos que señalan los objetos y añadiendo texto que describe la clase de los objetos que se encuentran en su interior.

### 1.3.5 ALGORITMOS DE DETECCIÓN DE OBJETOS

Este tipo de algoritmos tienen como meta la detección de diferentes objetos pertenecientes a clases conocidas como lo son: personas, autos, animales, rostros, entre otras, dentro de

una imagen. Dichos algoritmos son capaces de identificar distintos objetos en diversas posiciones o escalas en la imagen que se analiza [25].

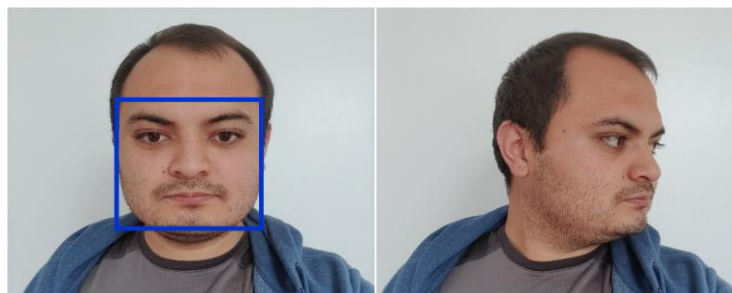
Los algoritmos de detección de objetos son modelos que se construyen a base del entrenamiento mediante ejemplos para así poder identificar características propias de cada clase, generalmente utilizando cientos a miles de imágenes para poder realizar esta acción [25]. Entre los principales algoritmos de detección de objetos tenemos a las Haar Cascade y a las redes neuronales.

### 1.3.5.1 Haar Cascade

Los clasificadores conocidos como Haar Cascade son algoritmos encargados de analizar el contraste entre regiones cercanas de la imagen para determinar si una fotografía o una secuencia de fotos contiene en su interior concordancias, igualdades conocidas como “match”, con un tipo específico. De esta forma se da la posibilidad de obtener identificaciones de los objetos que estén programadas para reconocer [26].

Las Haar Cascade se consideran inestables e imprecisas en muchos casos debido a las variaciones de luz, cambios en el ángulo de visión, la distancia, la falta de estabilización del video y la interferencia. Cuando se presentan los casos descritos anteriormente, al extraer los detalles de la imagen procesada se obtienen características o “features” erradas, dando resultados no satisfactorios que evitan la identificación del objeto pese a que el algoritmo haya sido entrenado para hacerlo [26].

Un ejemplo de esto sería una Haar Cascade que está diseñada para solo identificar rostros de frente como se muestra en la Figura 1.9. Al presentarse una pequeña variación, que sería el giro de la cabeza, la persona ya no sería identificada puesto que no cumple con las “features” que debe tener para considerarse un rostro visto desde el frente.



**Figura 1.9** Ejemplo de Haar Cascade que detecta solo rostros viendo de frente

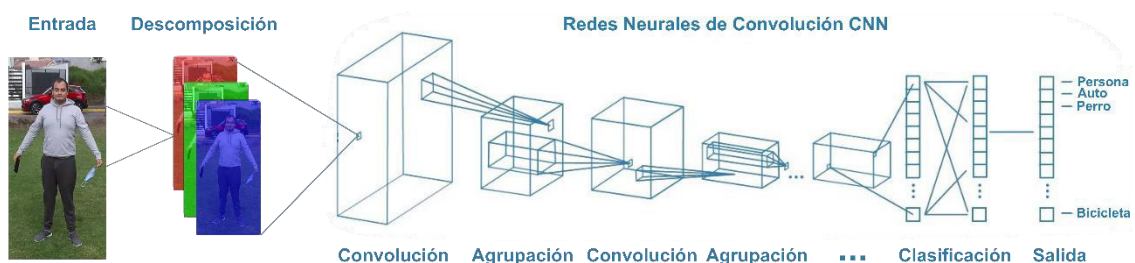
Estas features son propias de los clasificadores en cascada propuestos por Paul Viola y Michael Jones en el paper, "Rapid Object Detection using a Boosted Cascade of Simple

Features" in 2001, donde se analiza a mayor profundidad las más de 160000 features correspondientes para la clasificación de objetos mediante estos algoritmos [27]. Sin embargo, son muy utilizadas en distintas aplicaciones por considerarse de fácil implementación y se tienen distintos tipos especializados para objetos específicos.

### 1.3.5.2 Redes neuronales

Al poner en práctica la inteligencia artificial, es posible tener como resultado las redes neuronales, estos elementos dotan a las computadoras u otros sistemas la capacidad de realizar o tomar acciones basándose en la experiencia. Esta experiencia viene de la muestra que se ha tomado al momento de realizar su entrenamiento. Las redes neuronales permiten dar predicciones de elementos con características similares a los que se haya utilizado como base en su fase de entrenamiento [24].

El uso de este tipo de elementos ha aumentado a través de los años debido a un continuo desarrollo de nuevos algoritmos más eficientes y equipos computacionales cada vez más potentes y veloces. Lo que ha causado que cada vez se utilicen más este tipo de redes en diversos campos de aplicación, como lo es la seguridad, la salud, el análisis de datos, el márketing, las redes sociales, tecnologías de la información, entre otros. En este caso, cabe resaltar a las Redes Neuronales Convolucionales que son las más utilizadas en los años recientes. Son modelos que trabajan realizando matemática entre matrices y capas conectadas que le dotan de un excelente comportamiento en problemas que requieren de aprendizaje. Son ideales para trabajar con grandes cantidades de datos y por estas razones son muy utilizadas para clasificación de imágenes, visión por computador y procesamiento de lenguaje natural. La ventaja de trabajar con convolución es que se tiene mucho menor peso y las conexiones que se realizan entre las neuronas de cada capa la vuelven más rápida, analizando solo regiones de las imágenes en lugar del todo [27].



**Figura 1.10** Ejemplo de las capas de convolución en una red neuronal [27].

En la Figura 1.10 se tiene una mejor representación de la división de una imagen en sus diferentes capas, primero se realiza la descomposición en los tres colores base rojo, verde y azul. Estos datos sirven de entrada para la primera capa que realiza operaciones



matemáticas, estos resultados a su vez se convierten en la entrada de la capa siguiente, continuando así de capa en capa hasta obtener el resultado deseado.

De entre la gran variedad de redes neuronales aplicadas a la detección de objetos, se describen dos potenciales candidatas para la aplicación que es parte de este trabajo de titulación: Tiny Yolo y SSD MobileNet. En el Capítulo 2 se realiza una comparación de las ventajas y desventajas de cada una, se analizan sus características y se detallan las razones por las se optó por una de ellas. Centrándose en este tipo de redes para el reconocimiento de objetos, es necesario aclarar que las mismas son entrenadas con la base de datos de nombre de COCO (Common Objects in Context) y utilizando como medio de sintonización fina a PASCAL VOC [28], es decir se cambian parámetros del archivo *.config* en base a este, que es una colección de distintos dataset para la detección de objetos.

Entonces debido a lo descrito, se puede afirmar que los descriptores para la distinción entre las diferentes clases de objetos son aprendidos y configurados intrínsecamente por cada una de las redes neuronales, basándose en las diferentes imágenes del dataset que hayan seleccionado para su entrenamiento; determinando dentro de la “caja negra” que es la red neuronal, las características propias de cada clase y de esta manera poder diferenciar los objetos.

Sabiendo que los procesos correspondientes al entrenamiento en ambas redes son similares, a continuación, se provee un breve resumen de las mismas.

#### **1.3.5.2.1** *Tiny Yolo*

Este modelo está basado en la red neuronal Yolo (You Only Look Once), una red para detección de objetos caracterizada por su proceso de clasificación como regresión, factor determinante para su velocidad. Esta red divide la imagen en cuadrículas para predecir el objeto dando su precisión como una probabilidad de clase. Tiny Yolo es una versión más liviana y rápida de esta red que trabaja de la misma manera presentando un mejor desempeño en computadoras con recursos limitados, esto se logra sacrificando su precisión [29] [30].

#### **1.3.5.2.2** *SSD MobileNet V2*

El Single Shot Detector, también conocido como SSD basa su funcionamiento en una única red neural con estructura de capas organizadas encargadas de dibujar los cuadrados de borde (bounding boxes en inglés) a los objetos para obtener sus características en forma de mapas, haciéndola más rápida y precisa que el tipo de red anteriormente descrito [31].

Esta red es una de las preferidas al momento de realizar aplicaciones en dispositivos portátiles debido a su rapidez y ligereza.

### **1.3.6 ALGORITMOS DE SEGUIMIENTO O TRACKING**

Estos algoritmos como su nombre lo indica, realizan acciones de seguimiento. Su funcionamiento empieza después de que el objeto que se desea seguir es delimitado por un bounding box. Internamente realizan el análisis a una secuencia de imágenes (frames) comparando la ubicación actual de los píxeles que están dentro de la zona delimitada con la ubicación de estos en el siguiente frame, para de cierta manera estimar donde se encontrará el objeto contenido en la bounding box. Este tipo de algoritmos dejan de funcionar si el objeto que se encuentra siguiendo sale del campo de visión o a su vez es ocluido por otro, por tanto pueden dar falsos positivos [32].

Dentro de estos algoritmos están dos que son considerados como potenciales candidatos para la implementación de este proyecto: TLD, y KCF. En el Capítulo 2 se realiza una comparativa entre ellos y se explica cómo y por qué se escogió el usado en este proyecto. A continuación, se da una corta presentación de los mismos.

#### **1.3.6.1 TLD (Tracking – Learning – Detection)**

El TLD es un framework que fue diseñado para seguir a un objeto desconocido durante un periodo de tiempo indeterminado. Está conformado por 3 subetapas que trabajan al mismo tiempo combinándose entre sí: Tracking (seguimiento), Learning (aprendizaje) y Detection (detección).

El tracking permite estimar la posición del objeto entre frames consecutivos siempre que este no sea obstruido o salga del campo de visión. Por otro lado, el detector trata como independiente a cada frame, lo escanea y compara con el anterior para poder encontrar al objeto dentro de una imagen, esto muchas veces da un falso positivo. Esta etapa se basa en el nuevo paradigma de aprendizaje llamado aprendizaje P-N, en el cual el detector se evalúa en cada frame del video y sus respuestas se analizan por 2 tipos de “expertos”: P-experto que reconoce las detecciones fallidas y el N-experto que reconoce las falsas alarmas. Finalmente, el componente learning observa el rendimiento y funcionamiento de las etapas anteriores para estimar los errores del detector y generar ejemplos de entrenamiento con el fin de evitar errores en el futuro. Si se desea leer más información acerca de este algoritmo y su parte matemática puede dirigirse a las publicaciones [32] y [33].

#### **1.3.6.2 KCF (Kernelized Correlation Filter)**

Este algoritmo es ideal para aplicaciones en los que se deba analizar videos en tiempo real gracias a su buena velocidad de procesamiento, además de ser excelente opción para trabajar en tarjetas embebidas (SBC) por su bajo consumo de recursos [34]. El componente principal de este traqueador es un clasificador discriminatorio, este se encarga de distinguir entre el objetivo a seguir y el entorno para poder “identificarlo” en los siguientes frames. Su funcionamiento se basa en métodos de aprendizaje discriminativos implementando transformadas de Fourier rápidas e inversas (FFT e IFFT) y una extensión multicanal de filtros de correlación lineal, lo que aumenta su eficiencia y rapidez en los cálculos matemáticos; sin embargo, no es bueno cuando el objeto se pierde completamente de la imagen [35]. Una explicación detallada de la matemática que está detrás del funcionamiento de este algoritmo se la encuentra en [36].

La aplicación propuesta debe principalmente cumplir con dos parámetros principales: ser un sistema portátil y de bajo costo. Por lo que para cumplirlo es necesario utilizar computadoras que cumplan con ambas características, teniendo por tanto como mejor candidato el uso de SBC o computadoras de placa reducida.

### **1.3.7 SINGLE BOARD COMPUTER (SBC)**



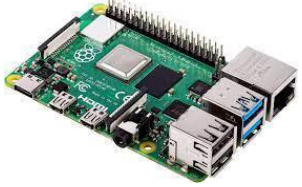
Este acrónimo hace referencia a las computadoras de pequeñas dimensiones que tienen todos sus elementos soldados en una placa, sin dejar sus características que la convierten en una computadora completa. Muchas tienen variedad de puertos y periféricos que permiten versatilidad en sus conexiones, que las hacen aptas para diversidad de aplicaciones. Generalmente son usadas donde un computador de tamaño real no podría ser ubicado ya sea por los recursos desaprovechados, por el consumo de energía o su tamaño, ya que al ser computadores de placa reducida tienen un menor consumo de energía, aunque recursos más limitados. Entre las ventajas más importantes que tienen las SBC son: su precio económico, ser totalmente funcional, cantidad de puertos y periféricos comparables a una computadora de tamaño normal y seguir siendo de fácil transporte e instalación.

Para este trabajo de titulación se ha tomado en consideración principalmente tres alternativas de SBC, que se presentan en la Tabla 1.3, debido a ser las más utilizadas, económicas, fáciles de conseguir, tener un procesador capaz de manejar diferentes tareas y una GPU relativamente buena en comparación a otras. Adicional a esto, las tres alternativas son las SBC más utilizadas para proyectos de desarrollo que incluyen portabilidad, debido a su tamaño y fácil transporte, convirtiéndolas en idóneas para el tipo de tarea que se desarrolla en el presente trabajo de titulación. Gracias a la gran cantidad

de usuarios, cuentan con una amplia comunidad dispuesta a la solución de dudas y problemas que puedan presentarse y esto incluye al apoyo y guías propias de los fabricantes de estas para incentivar el desarrollo de aplicaciones.

En la Tabla 1.3 se realiza una comparativa entre las tres SBC que se tomaron en cuenta para la realización del proyecto, se procedió a analizar las características de cada una para seleccionar la más adecuada para este trabajo

**Tabla 1.3** Comparación de las SBC analizadas para el trabajo [37] [38] [39].

Característica	Raspberry Pi 3b+	Jetson Nano	Raspberry Pi 4
<b>Fotografía</b>			
<b>Descripción</b>	Corresponde a la tercera generación de computadores de tarjeta reducida de raspberry.org	Tarjeta desarrollada por Nvidia especializándose en IA, capaz de correr diferentes redes en paralelo	La última SBC de raspberry.org mantiene las dimensiones, mejorando componentes, procesador, GPU y aumentando RAM
<b>Sistemas Operativos</b>	Raspberry OS Kali Linux Pidora Windows 10 IoT Core Ubuntu Core RISC OS SARPi (Slackware ARM for Raspberry Pi) Arch Linux ARM FreeBSD RetroPie	Ubuntu y otros sistemas operativos Linux	Raspberry OS Twister OS Ubuntu Mate NOOBS y NOOBS Lite RetroPie OSMC Kali Linux Pidora OpenELEC OS RISC Windows IoT Core Ubuntu Core Ubuntu BalenaOS Arch Linux
<b>CPU</b>	Cortex-A53 (ARMv8) 64-bit SoC	Quad-core ARM A57	Cortex-A72 (ARM v8) 64-bit SoC
<b>GPU</b>	VideoCore IV 400 MHz	128-core Maxwell	Broadcom VideoCore VI (Pi 4)

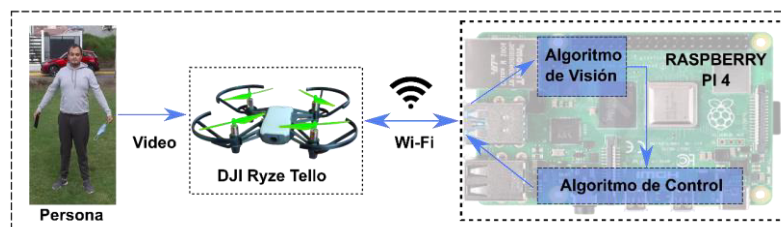
<b>Velocidad del CPU</b>	1.4GHz	1.43 GHz	1.5GHz
<b>RAM</b>	1GB LPDDR2 SDRAM	2 GB, 4 GB LPDDR4 SDRAM	2GB, 4GB or 8GB LPDDR4-3200 SDRAM
<b>Alimentación</b>	5V DC 2.5A	5V DC 2A	5V DC 3A
<b>Conectividad Inalámbrica</b>	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE	No dispone, es necesario conectar elementos a parte	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
<b>Puertos</b>	Extended 40-pin GPIO header Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) Full-size HDMI 4 USB 2.0 ports CSI camera port for connecting a Raspberry Pi camera DSI display port for connecting a Raspberry Pi touchscreen display 4-pole stereo output and composite video port Micro SD port for loading your operating system and storing data Power-over-Ethernet (PoE) support (requires separate PoE HAT)	2x MIPI CSI-2 DPHY lanes Gigabit Ethernet, M.2 Key E HDMI and display port 4x USB 3.0, USB 2.0 Micro-B GPIO, I2C, I2S, SPI, UART	Gigabit Ethernet 2 USB 3.0 ports; 2 USB 2.0 ports. Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards) 2 × micro-HDMI ports (up to 4kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode) OpenGL ES 3.1, Vulkan 1.0 Micro-SD card slot for loading operating system and data storage
<b>Precio</b>	\$59.99	\$ 59.99 - 115.99	\$ 35.00 - 89.99

Por las ventajas específicas que se mencionan en el cuadro comparativo se decide optar por la Raspberry Pi 4 de 8 Gb debido a sus ventajas en periféricos, conexiones, CPU, RAM y la cantidad de diferentes sistemas operativos que puede manejar. Inclusive el contar con pines lógicos de propósito general se considera como una ventaja en la integración de hardware. Adicionalmente y pese a que la tarjeta Jetson Nano tiene un mejor GPU especializada para inteligencia artificial, se decide utilizar la Raspberry Pi 4 porque sus conexiones inalámbricas facilitan la integración con la aeronave en el proyecto de titulación, sin la necesidad de requerir un dispositivo adicional, convirtiéndola en una opción económica, potente e idónea sobre otras alternativas.

## 2 METODOLOGÍA

En el presente trabajo de titulación se investiga información técnica sobre dispositivos y algoritmos para posteriormente analizar, comparar y aplicar los recursos obtenidos para el desarrollo de un prototipo físico, el cual se construye en base a experimentación y finalmente se lo somete a diversas pruebas para su validación. Por esto, este trabajo se divide en cuatro fases para su desarrollo.

La primera fase se encuentra descrita en el Capítulo 1, donde se detalla la teoría relevante en los temas que se tratarán en las demás secciones, se obtienen datos técnicos pertinentes y se describen los algoritmos que serán útiles posteriormente. La segunda fase corresponde al diseño de los algoritmos, se realizan comparaciones para determinar cuáles son los escogidos y se describen los programas que permiten la posterior experimentación con los dispositivos. La fase de implementación utiliza los algoritmos desarrollados en la fase anterior y se los programa en el hardware para su ejecución y análisis, esto en búsqueda de cumplir con el objetivo principal del trabajo de titulación y de esta manera proceder a la última fase. Esta corresponde a la validación donde se realizan las pruebas de los algoritmos anteriormente diseñados e implementados para demostrar su funcionamiento y analizar el desempeño del prototipo.



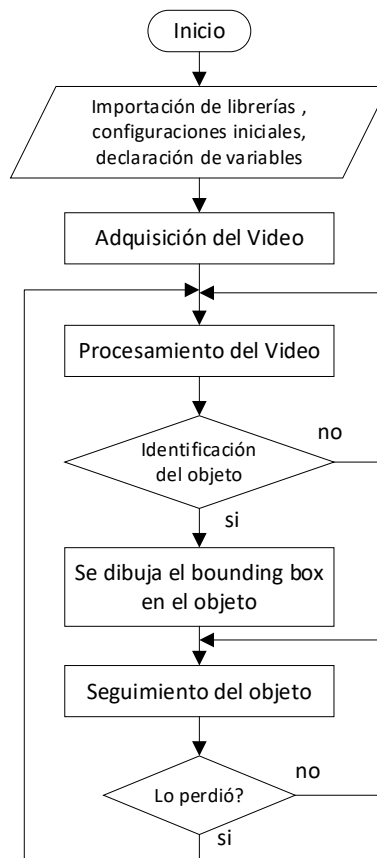
**Figura 2.1** Diagrama general del funcionamiento del prototipo

La Figura 2.1 muestra de manera simplificada el funcionamiento del sistema propuesto, donde el cerebro de la operación se encuentra dentro de la Raspberry, mientras que la cámara o sensor se encuentra integrada en el dron, que es también el actuador que toma las acciones para mantener el seguimiento de la persona. Para la programación se optó por Python, porque es un lenguaje de programación de alto nivel con gran variedad de librerías de código abierto, y como IDE de programación se utilizó el llamado Tonny que viene preinstalado en la Raspberry.

### 2.1 ALGORITMO HÍBRIDO DE VISIÓN POR COMPUTADOR

El algoritmo de visión es el corazón de este trabajo, se encarga de identificar a una persona de interés dentro de una secuencia de imágenes, seleccionarla y seguirla, de manera que

siempre permanezca en el campo de visión de la cámara. En el diagrama de bloques de la Figura 2.2 se puede ver de forma general el algoritmo implementado en este trabajo.



**Figura 2.2** Diagrama general del algoritmo de visión implementado

En las siguientes secciones se describirán las partes que componen este diagrama para entender de mejor manera el funcionamiento en conjunto del algoritmo de visión.

## 2.1.1 LIBRERÍAS PRINCIPALES

Para el desarrollo del algoritmo híbrido en su totalidad fue necesario importar diversas librerías. A continuación, se describe de forma general el paquete especializado para visión artificial OpenCV y la librería necesaria para la conexión y control del dron.

### 2.1.1.1 Librería OpenCV

La librería Open Source Computer Vision Library (librería de código abierto para visión por computador) es una herramienta computacional que incluye una gran cantidad de algoritmos para diferentes y variadas aplicaciones correspondientes a visión por computador, que cuenta con un constante desarrollo y despertando el interés en estudiantes, investigadores y público en general que desee conocer más sobre estos temas [40]. De esta librería se instalaron los paquetes *contrib* que incluyen a los algoritmos

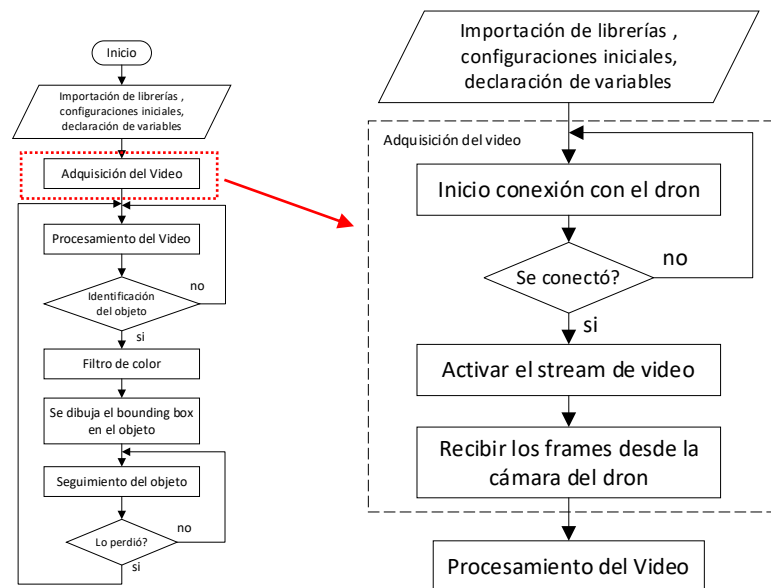
necesarios para la adquisición de video y el trackeador que es parte fundamental de este proyecto ejecutando el comando *pip install opencv-contrib-python*.

### 2.1.1.2 Librería djitellopy

Es una librería de código abierto basada en Python que utiliza el medio de comandos oficial conocido como SDK proporcionado por Tello para propósitos de desarrollo de software. La librería se encuentra en constante desarrollo e invita a programadores a contribuir en mejorarla. Actualmente permite tener la capacidad de enviar comandos, conseguir la transmisión de video del dron en tiempo real, recibir información simple de la aeronave y trabajar con Python gracias a su total integración con este lenguaje de programación [41]. Para incluirla se debe ejecutar el comando *pip install djitellopy* en una ventana de terminal y se la importa con la línea *from djitellopy import\** para trabajar con todos sus componentes.

### 2.1.2 ADQUISICIÓN DEL VIDEO

En esta etapa del algoritmo, véase la Figura 2.3, se realiza la adquisición de la imagen en la Raspberry desde la cámara del dron (las características de la misma se encuentran explicadas en la Sección 1.3.3).



**Figura 2.3** Diagrama de la adquisición del video

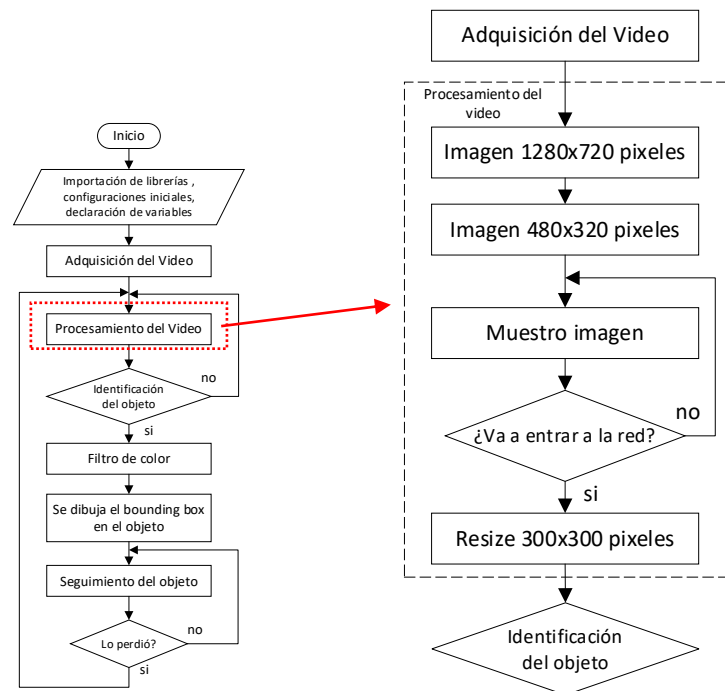
El video obtenido es la principal herramienta para la ejecución del algoritmo completo ya que a partir de aquí se diseña el resto del programa, esto implica que la conexión del dron con la Raspberry debe ser estable y de buena velocidad para que el procesamiento de la imagen no presente o tenga un retraso menor hasta llegar al algoritmo de control. La etapa comienza realizando la conexión WiFi entre la Raspberry y el dron, después se procede a



activar el stream de video para luego recibir estas imágenes y enviarlas a la etapa de procesamiento, esto se resume en la Figura 2.3.

### 2.1.3 PROCESAMIENTO DEL VIDEO

En esta sección se realiza el cambio de tamaño y resolución de la imagen proveniente desde la cámara, es decir se realiza una reducción de la cantidad de píxeles que la conforman. El diagrama de flujo se puede ver en la Figura 2.4.



**Figura 2.4** Diagrama del cambio de tamaño y resolución de la imagen de entrada

Como se observa, este proceso se lo realiza con la imagen recibida para convertirla en otra que cuente con una resolución de 480 píxeles de ancho y 320 píxeles de alto. En el caso específico de la imagen recibida del dron se tiene una resolución de 720p la cual se traduce en 1280 píxeles de ancho con 720 píxeles de alto llegando a ocupar un área en píxeles correspondiente a 921600. De manera general, tener una mejor resolución conviene en aplicaciones de visión, pero en este caso específico existe una desventaja. Esto debido a que el área correspondiente en píxeles es determinada por el recuadro en el que se encuentra la persona de interés, el tener una resolución mayor significa tener una mayor área de trabajo. Por lo que, el área en píxeles tiene una relación directa en la rapidez en la que el algoritmo de seguimiento trabaja, por lo que con una menor área se tiene un procesamiento más rápido en el algoritmo que corresponde al seguimiento.



**Figura 2.5** Cambio de tamaño de la imagen de entrada

El reducir el área de 720p a la resolución propuesta de 480x320 puede no parecer un cambio severo, pero es muy conveniente ya que se traduce a una reducción de un 83.33% en el área ocupada en píxeles en la imagen. Este cambio es evidenciado en la Figura 2.5. Por tanto, la reducción del área total en píxeles causa que el algoritmo encargado del seguimiento tenga una menor carga y por tanto pueda trabajar de manera más rápida.

Adicionalmente, en caso de utilizar una red neuronal de convolución se deben tener en cuenta los parámetros de imagen con los que haya sido entrenada. En este caso la SSD MobileNet V2 requiere que la imagen de entrada sea de 300x300 píxeles, por ende, es necesario realizar otro cambio en la resolución anterior. El motivo principal de no utilizar solamente un cambio en la resolución a 300x300 píxeles es la pérdida de la perspectiva en visión, apartado que puede ser evidenciado en la Figura 2.6. Justamente se observa que al mantener esta relación de cambio se genera un cuadrado donde el usuario no tendría una clara visualización de los acontecimientos presentados por la cámara.



**Figura 2.6** Comparación entre resolución 480x320 vs 300x300

El problema del cambio de resolución se centra en que se genera una imagen cuadrada que no representa un problema para la computadora, pero sí para el usuario final que es un ser humano. Adicional a esto el realizar solo un cambio en la resolución del video no

supone un aumento en la velocidad del algoritmo, pero conviene mucho en la presentación de los datos obtenidos desde la cámara del dron.

#### **2.1.4 IDENTIFICACIÓN DE OBJETOS**

En este apartado se plantea la selección entre tres distintas alternativas que permiten la identificación de objetos, justamente las tres explicadas en la Sección 1.3.5. Estas alternativas se evaluaron y ejecutaron en una laptop y en una Raspberry Pi 4 puesto que, este último, es el cerebro a emplearse en el dispositivo portátil desarrollado en este proyecto y por tanto es mandatorio que el algoritmo pueda ser ejecutado en este sistema. Es importante tomar en cuenta que la identificación de objetos mediante el uso de visión por computador es un apartado que consume recursos computacionales en gran medida, así que este detalle será un factor decisivo para la elección del algoritmo encargado de realizar este proceso.

En este punto se detallan los tres diferentes métodos: Haar Cascade, Tiny Yolo y SSD MobileNet V2 (explicados brevemente en la Sección 1.3.5), y las razones que motivaron al uso de una sobre las otras alternativas.

- Haar Cascade

Para esta aplicación se plantea el uso de las Haar Cascade como identificadores de objetos ya que, como se había descrito anteriormente, existen cascadas específicas para cada objeto. Si se desea identificar algo en específico se buscan características propias de esa clase; es decir que, para cada clase a identificar existe una cascada, en este caso el elemento de estudio es un ser humano, por lo que se utilizará la cascada específicamente entrenada para personas. Se procedió a evaluar las distintas alternativas de Haar Cascade que permiten identificar personas y se concluyó que la mejor opción para el proyecto sería la especializada en detectar caras y cuerpos enteros.

Para la implementación de este tipo de algoritmos es necesario el archivo .xml el cual debe ser implementado en Python para su ejecución. Después de insertar las imágenes de entrada, se obtiene como respuesta la imagen junto con el cuadro que rodea a la persona identificada como se puede ver en la Figura 2.7.



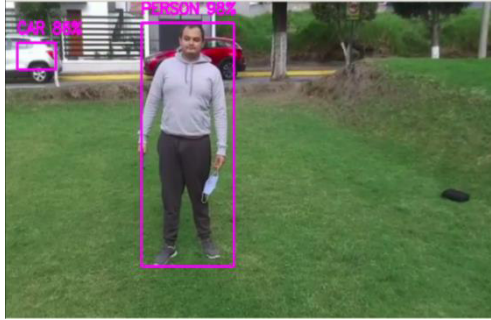
**Figura 2.7** Ejemplo de Haar Cascade para detectar un cuerpo entero. Al lado izquierdo se tiene una identificación correcta, mientras que en el derecho se presenta un error típico de bounding box en estos algoritmos

Es necesario tener en cuenta que hay factores como la posición del rostro o la postura del cuerpo que reducen drásticamente su respuesta, por lo cual, si se decide trabajar con este algoritmo la distancia con respecto a la cámara se vería muy limitada. Otro aspecto importante a tener en cuenta es que cada vez que se necesite cambiar de objeto de estudio obligatoriamente habría que alterar el código de programación. Adicionalmente, al utilizar elementos como las Haar Cascade no se puede controlar la precisión o la confianza del algoritmo, es decir, no es posible cambiar el porcentaje de confianza de detección para evitar errores o que se presenten falsos positivos durante su ejecución.

- Tiny Yolo

En este caso se plantea el uso de la CNN o red neuronal de convolución conocida como Tiny Yolo que permite distinguir objetos de diferentes clases. Al ser una versión ligera o light del algoritmo YOLO, esta se encuentra mejor optimizada para un menor consumo de recursos computacionales, lo que la vuelve una alternativa a tomar en cuenta sobre todo al trabajar en computadoras de placa reducida. En este caso específico Tiny Yolo es capaz de ser ejecutado en la tarjeta Raspberry Pi 4 a 3 fps, velocidad aceptable para estar corriendo en un dispositivo pequeño y de prestaciones limitadas.

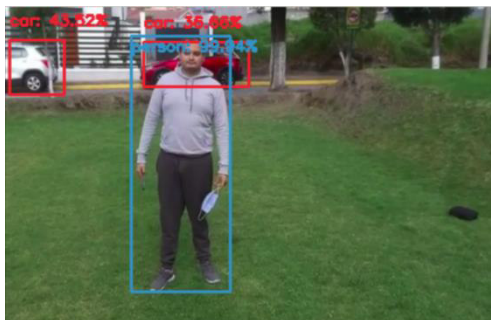
La ventaja de esta alternativa con respecto a la Haar Cascade es que al ser una red neuronal permite identificar diferentes clases de objetos, es decir que sin la necesidad de cambiar o reestructurar el código, es posible obtener información de varios objetos que se encuentren en una misma imagen. Como es posible apreciar en la Figura 2.8, la red identificó dos clases distintas de objetos en la misma imagen con cierta precisión y no se tiene la intermitencia de reconocimiento tan común presente en las Haar Cascade.



**Figura 2.8** Identificación de 2 clases (persona, carro) con la red Tiny Yolo

- SSD MobileNet V2

Esta versión light de la red neuronal SSD es ampliamente utilizada en el campo de la investigación gracias a su buena confiabilidad y resultados sin sacrificar la velocidad de procesamiento. Su capacidad de identificación de varias clases de objetos en la misma imagen sin la necesidad de tener una potente tarjeta gráfica la convierte en la mejor alternativa para este proyecto. Al igual que Tiny Yolo, esta red permite variar el rango de confianza para la identificación de objetos, dando al programador la capacidad de elegir la precisión de la red al obtener información sobre los objetos presentados. En la Figura 2.9 se muestra la imagen después de haber sido procesada por la red SSD MobileNet V2, obteniendo como resultado la identificación de tres objetos de dos clases diferentes (dos autos y una persona)



**Figura 2.9** Identificación de 2 clases (persona, carro) con la red SSD MobileNet V2

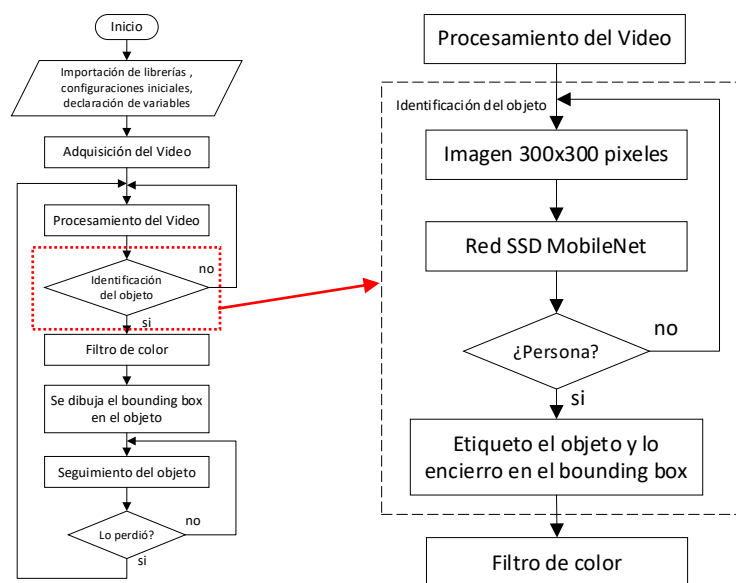
En la Figura 2.10 es posible observar de mejor manera las capacidades que tiene esta red con respecto a la identificación de objetos. En la imagen de la izquierda se puede observar un auto identificado desde 7 metros de altura. En la imagen de la derecha se decide en cambio exponer una muestra de la capacidad que tiene esta red de identificación, logrando obtener en pantalla cuatro distintas clases bien diferenciadas en tamaño y colores. En dicha imagen, se obtiene como resultado la identificación de izquierda a derecha de: una bicicleta, un perro, un automóvil y una persona.



**Figura 2.10** Identificación de varias clases (carro, bicicleta, persona, perro) con la red SSD MobileNet V2

Para la implementación de estas redes neuronales es necesario tener tres archivos principales, el primero consta de una lista de nombres de las clases que la red puede detectar, el segundo archivo contiene las configuraciones de la red y el tercero los pesos o weights. Cada uno de estos elementos es primordial para que las redes que permiten el reconocimiento de objetos funcionen de manera adecuada, por lo que, una alteración de estos archivos afecta de manera directa al funcionamiento de la red, ya que constan de parámetros y códigos proporcionados por los creadores de la red para su uso. Por esto una modificación en el mismo sin el conocimiento adecuado de su estructura causaría un daño permanente en su funcionamiento.

Para este trabajo se han realizado pruebas con las dos redes neuronales descritas dejando de lado a las Haar Cascades por su imprecisión cuando los objetos cambian de posición o postura y se ha escogido al SSD MobileNet sobre la Tiny Yolo por su velocidad y precisión al ser ejecutadas en una SBC.



**Figura 2.11** Diagrama de la identificación de objetos

La explicación con más detalles de las pruebas realizadas para comparar estas dos alternativas y sus resultados se los puede encontrar en las Secciones 3.1.1.1 y 3.1.1.2. La implementación de la red SSD MobileNet para el caso del presente proyecto se explica en la Figura 2.11.

Una propiedad intrínseca de este tipo de algoritmos conocidos como CNN es que es posible cambiar o alterar el grado de confianza en sus parámetros de identificación, lo que se traduce en resultados más precisos y confiables. Puesto que al bajar la confianza es posible tener falsos positivos porque la red no está segura de lo que está visualizando; este resultado es posible visualizarlo en el lado izquierdo de la Figura 2.12, donde se decide disminuir la confianza a un 20% como mínimo y se obtiene de resultado que al perro de la imagen se lo identifica erróneamente como oveja. Mientras que en el lado derecho se coloca la confianza en un valor del 75% de seguridad donde la red es capaz de determinar que el animal de la foto es un perro.



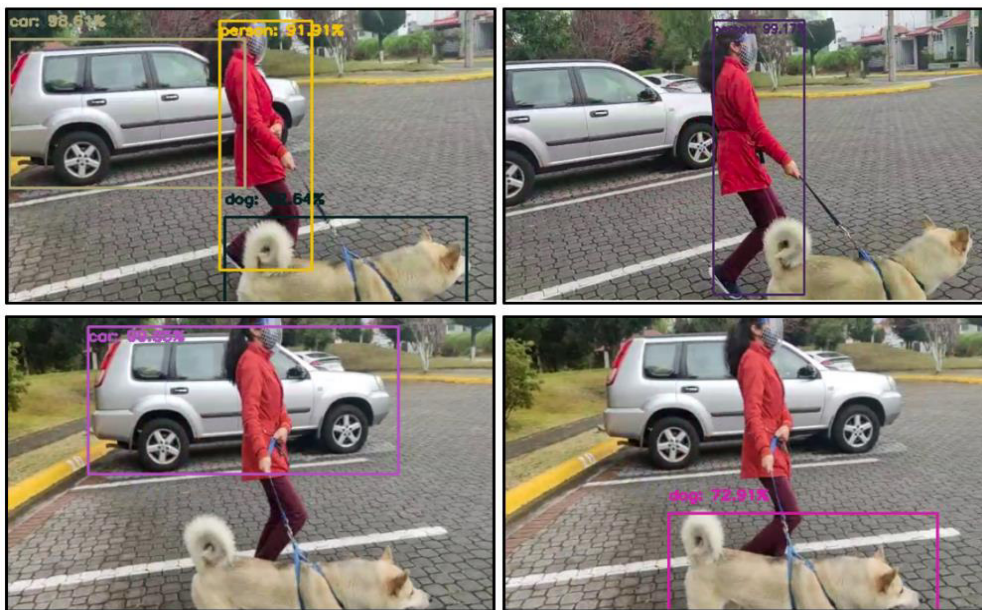
**Figura 2.12** Comparación entre la red con diferentes confianzas

#### **2.1.4.1 Discriminación de clases**

La red neuronal de convolución SSD Mobilenet V2 elegida para este trabajo es capaz de detectar diferentes clases de objetos y marcarlos todos a la vez en una imagen, por lo que para esta aplicación es necesario discriminar y poder solo seleccionar la clase que se necesita en la aplicación; es decir, poder separar y obtener solo la clase de interés conocida como “Persona”. No es posible hacer que la red entregue los resultados de una sola clase por si sola ya que se encuentra entrenada para identificar varias al mismo tiempo, por lo que siempre entrega como resultado la clase con el porcentaje más alto de semejanza, es decir la red utiliza el dato enviado con mayor valor para identificar una clase.

Para discriminar las distintas clases se utiliza una lista con la o las clases que se desea mostrar. La red devuelve como resultado las respuestas de todas las clases detectadas, las mismas que son comparadas con la lista de discriminación y en caso de que la clase detectada por la red se encuentra en la lista de la discriminación, se dibuja el bounding box

(rectángulo que encierra el objeto identificado) y se escribe el nombre de la clase resultante. Con este proceso es posible obtener solo los bounding box y las etiquetas de las clases que se coloquen en la lista de discriminación y por tanto poder segregar los resultados y mostrar solo el objeto de interés pese a que más clases se encuentren en la imagen. Esto se lo puede entender de mejor manera con la Figura 2.13, donde en la primera imagen se muestran tres clases distintas que son un auto, una persona y un perro, mientras que en las siguientes se ha realizado el proceso de discriminación para obtener solo una clase específica en cada imagen.



**Figura 2.13** Discriminación de clases en la red SSD MobileNet V2

Gracias a esta característica de la red SSD Mobilenet V2 es posible, mediante el rectángulo que encierra a la respectiva clase, delimitar el objeto.

#### **2.1.4.2 Delimitación del objeto**

La delimitación del objeto se refiere netamente al elemento conocido en inglés como bounding box, que es un rectángulo con el que se encierra el objeto de interés; para su dibujo es necesario conocer la ubicación de dos de sus esquinas, las que permiten graficar esta figura. La red tiene la capacidad de detectar el objeto y devolver los datos de su posición en forma de coordenadas de píxeles, estas permiten dibujar el cuadrado que rodea al objeto, para posterior a esto realizar el etiquetado de la clase. De esta manera, se obtiene como resultado final la imagen con el objeto identificado dentro de un rectángulo y con el nombre de su clase.



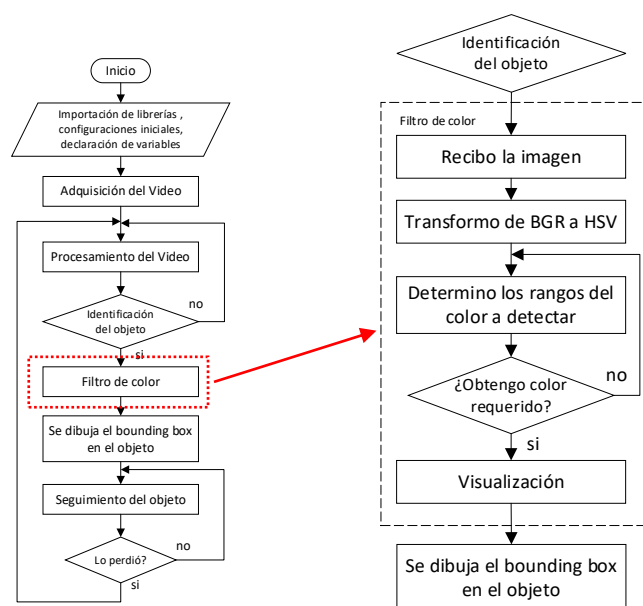
Este tipo de información, dada por la red, permite delimitar la región en la que se ubica el objeto dentro de la imagen. Para una mejor explicación se utiliza la Figura 2.14 como ejemplo, donde, como se ha explicado anteriormente, es necesario contar con las coordenadas de dos esquinas del rectángulo para poder dibujarlo. En este caso la imagen tiene una resolución de 480 píxeles de ancho y 320 píxeles de alto, la misma ha sido introducida en la red para obtener los datos que permiten el dibujo de la bounding box. El resultado es que el perro se encuentra en el recuadro formado por las coordenadas (102, 162) para su esquina superior izquierda y (233, 319) para la esquina inferior derecha. Con esta información es posible dibujar el rectángulo que contiene al objeto de interés.



**Figura 2.14** Coordenadas del bounding box

Esta característica de la red para obtener las coordenadas correspondientes al bounding box, que encierra al objeto, es necesaria para poder colocar los parámetros iniciales que serán los que determinan la zona de la imagen donde se realizará el seguimiento.

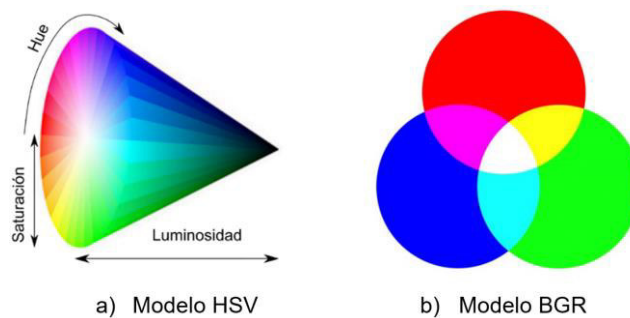
### 2.1.5 FILTRO DE COLOR



**Figura 2.15** Diagrama del filtro de color

El filtro de color se plantea debido a que debe existir un factor extra que permita diferenciar a los individuos que pertenecen a la misma clase, es decir en caso de que se presenten varios individuos en la misma imagen. La forma de identificarlos será por el color de su ropa por lo que se plantea utilizar un color distintivo que lo diferencie del resto. El diagrama del filtro de color aplicado se lo puede ver en la Figura 2.15.

De manera digital se tienen distintos modelos para representar los colores, los cuales tienen su propia forma de ser identificados, siendo los más conocidos el modelo HSV y BGR [42]. Las imágenes obtenidas desde OpenCV están en BGR, por lo que se debe tener en cuenta al momento de asignar colores en este formato. Las siglas RGB corresponden a la representación de cualquier color de manera digital en tres colores principales que son R red o rojo, G green o verde y B blue o azul. Mediante las combinaciones de estos tres colores en diferentes valores es posible representar cualquier color. La diferencia entre el modelo RGB y el BGR es simplemente el orden en que se presentan estos colores.



**Figura 2.16** Espacio de color HSV y BGR

El modelo para representación de color que se utiliza en este proyecto es HSV, Figura 2.16 a), debido a que este filtro determina los colores de acuerdo con propiedades de tono, saturación y luminosidad. Esto permite que sea más versátil en la identificación de color incluso si se presentan variaciones en ciertos tonos debido a la luminosidad del ambiente. Al momento de utilizar HSV como filtro de color es posible separar colores específicos de acuerdo con sus tres valores correspondientes en Hue, Saturación y Luminosidad que van de 0 a 255.

El Hue, también conocido como matiz o tono, hace referencia a la gama de colores que cambian al variar este parámetro. Con esta variación de tono es posible representar los colores primarios y sus mezclas. Mientras que la variable de Saturación controla qué tan vívido o saturado es el color al que se hace referencia, se podría decir que representa a la cantidad de color que se tiene. Finalmente, la Luminosidad, como su nombre lo indica, es la cantidad de luz que tiene el color. En caso de tener valores elevados en este parámetro

la imagen tiende al color blanco mientras que si se disminuye al mínimo se convierte en negro.

Para realizar esta extracción de color se realizan los siguientes pasos:

- Se efectúa la conversión de la imagen de BGR a HSV como se puede apreciar en la Figura 2.17:



**Figura 2.17** Conversión de BGR a HSV

- Se realiza una imagen de máscara donde solo los valores que, comparados con el color asignado, se encuentren en el mismo rango HSV. Por lo que, serán colocados como 1 o marcándose en color blanco, mientras que el resto de la imagen tendrá un valor de 0 o se pintará de color negro, véase la parte izquierda de la Figura 2.18.
- Finalmente se realiza una operación lógica de AND entre la máscara y la imagen original para obtener solo los sectores que correspondan con el color que se desea sea filtrado. En el caso de la Figura 2.18 se separa el color piel en sus diferentes tonos.



**Figura 2.18** Filtro de color piel aplicado a la imagen de la Figura 2.17

En la Figura 2.19 es posible apreciar otras separaciones o filtraciones de color que se realizaron como es separar solo tonos de verde.



**Figura 2.19** Filtro del color verde

En la Figura 2.20 se puede ver la aplicación del filtro para ciertas tonalidades del rojo, obteniendo buenos resultados.



**Figura 2.20** Filtro de color rojo

En este trabajo se necesita que el filtro trabaje con el color naranja ya que la persona a la que debe seguir el dron portará un chaleco de este color, como se puede ver en la Figura 2.21, para esto se setearon los siguientes valores límites en el filtro HSV:

[Hue, Saturación, Luminosidad]

Límite inferior: [0, 109, 115]

Límite superior: [75, 255, 255]



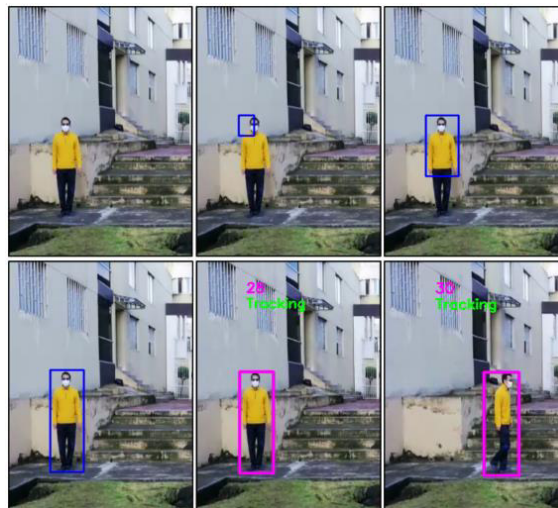
**Figura 2.21** Filtro de color naranja

Con este intervalo de valores se logra separar ciertos tonos del color naranja para que el funcionamiento del prototipo no se vea tan afectado ante los cambios bruscos de luminosidad que se presentan al trabajar en ambientes exteriores.

Para la conversión de BGR a HSV, OpenCV realiza internamente operaciones matemáticas término a término, es decir que tanto el Hue como la Saturación y el Value están en función de los colores azul, verde y rojo (BGR). Las ecuaciones correspondientes se las puede encontrar en [43], por lo que si se desea más información se recomienda leer el artículo completo.

### 2.1.6 SEGUIMIENTO DE OBJETOS

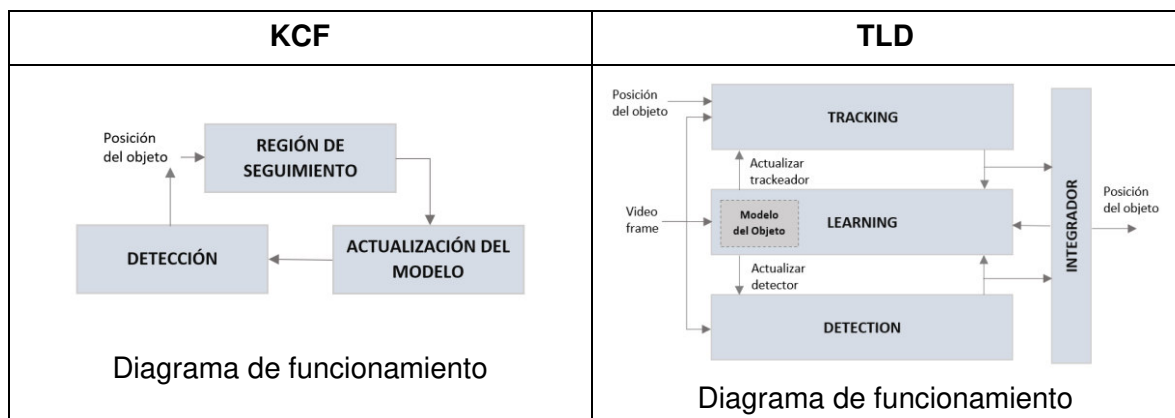
Como ya se explicó en el Capítulo 1, estos algoritmos son los encargados de seguir a un objeto dentro de un conjunto de frames o imágenes secuenciales (video) después de que se lo haya encerrado en un recuadro delimitador o bounding box, como se ve en la Figura 2.22.



**Figura 2.22** Dibujar el bounding box alrededor del objeto a seguir

En esta sección se realizará una descripción de las principales características de los dos algoritmos de seguimiento expuestos en la Sección 1.3.6 y se explicarán las razones por las que se escogió el KCF para la implementación del algoritmo híbrido.

**Tabla 2.1** Principales características del KCF y TLD [44] [33]



Utiliza las muestras positivas y negativas para un mejor funcionamiento	Tres etapas de funcionamiento: Trackeador, Detector y aprendizaje
Trabaja con transformadas rápidas e inversas de Fourier para aumentar su velocidad	Puede recuperarse después de una oclusión total o parcial del objeto sacrificando la estabilidad
No escanea los píxeles fuera del bounding box, este algoritmo analiza el histograma de gradiente para mejorar la precisión del seguimiento	Capacidad de aprendizaje en línea
Es ideal para aplicaciones que requieran analizar videos en tiempo real	La combinación de módulos de seguimiento y detección lo hace confiable para videos de larga duración
Consume pocos recursos computacionales	Consume una mayor cantidad de recursos computacionales en comparación con el algoritmo KCF
Tiene buena robustez y baja probabilidad de presentar falsos positivos.	La etapa de detección puede causar que el algoritmo presente falsos positivos
El objeto que se desea seguir debe ser seleccionado en el primer frame	
Los algoritmos no necesitan conocer con anterioridad al objeto que van a seguir	

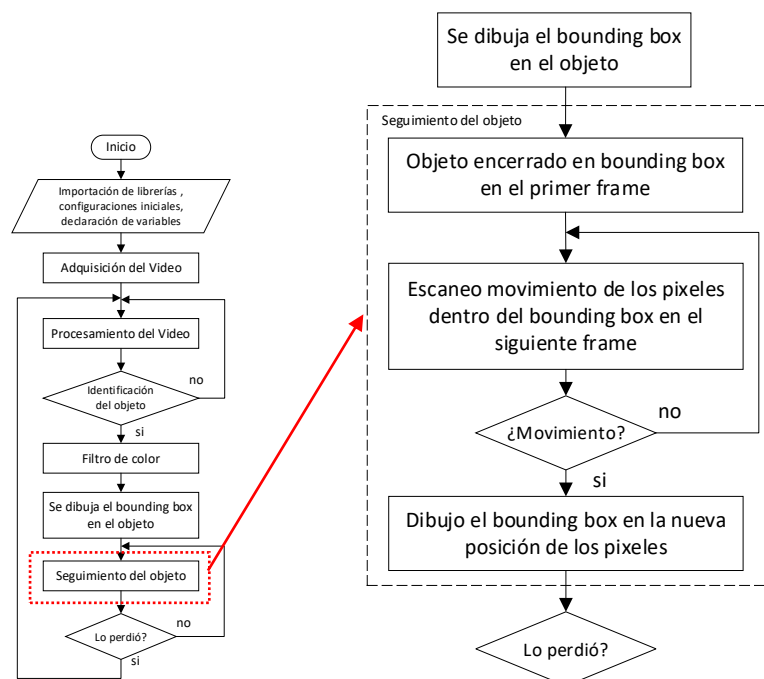
Como se puede ver en la Tabla 2.1, ambos algoritmos tienen buenas características de funcionamiento; sin embargo, se requiere más información sobre el rendimiento real para poder escoger el más adecuado para este trabajo. Por lo que, se decidió buscar trabajos en donde se los haya implementado y ver cual ha presentado un mejor desempeño. En la Tabla 2.2 se resumen los resultados experimentales presentados en [35]. Dichos resultados se obtuvieron en una computadora propiamente dicha, es decir de tamaño normal, por lo tanto, sirven de referencia para realizar una comparación de como serían los posibles resultados al ejecutarlos en una SBC. Como se explicó en los alcances de este proyecto, los algoritmos se deben implementar en una Raspberry Pi 4, esto significa que los recursos computacionales no son los mismos.

**Tabla 2.2** Comparación de los resultados obtenidos experimentalmente en [35].

Parámetros	TLD	KCF
FPS	18.77	230.26
Falla por oclusión	26.7%	36.1%
Falla por salida del campo de visión	26.2%	47.0%

Luego de varias pruebas que están disponibles en la Sección 3.1.2 se observó que si bien TLD permite rastrear, detectar e incluye capacidades de aprendizaje, y es capaz de recuperarse de la oclusión, devuelve un número considerable de falsos positivos. Por otro lado, el KCF es más rápido que el TLD, pero no puede recuperarse de la oclusión. De hecho, activa una bandera de "Objeto perdido" cuando deja de rastrear. Por estas razones y las demás que serán expuestas en la Sección 3.1.2.2 se seleccionó KCF. Justamente, la bandera de "Objeto perdido" puede ser empleada para ejecutar nuevamente la red neuronal y reiniciar el proceso de seguimiento. Adicionalmente, ha demostrado menos porcentaje de falsos positivos que el TLD, véase la Tabla 2.2.

### 2.1.6.1 Funcionamiento del algoritmo de seguimiento



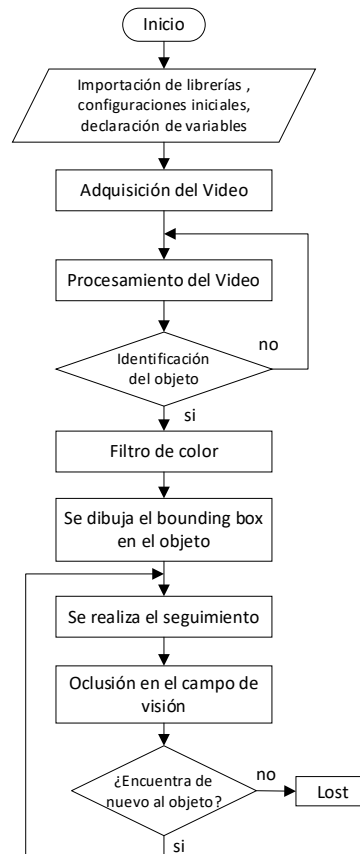
**Figura 2.23** Diagrama del trackeador KCF dentro del diagrama general

El proceso de trackeo descrito en la Figura 2.23 empieza luego de que la red de identificación encuentre a la persona con chaleco naranja y la encierre en un bounding box. En particular, el algoritmo de seguimiento escanea si existe movimiento de los pixeles dentro del recuadro delimitador entre cada frame del video y si lo hay vuelve a dibujar el bounding box en la nueva posición.

Los datos de la posición del centro del bounding box son enviados al algoritmo de control para que se realice una comparación con el punto de referencia fijado en el campo de visión del dron y se envíen las acciones de control necesarias para maniobrarlo y no perder de vista al objeto.

### 2.1.6.2 Pérdida del objeto en el algoritmo de seguimiento

Existen diversos factores como: el viento, la velocidad y movimientos bruscos de la persona, objetos que obstruyan la visión, etc., que pueden causar problemas mientras el dron se encuentra siguiendo a su objetivo haciendo que este salga del campo de visión y detenga el proceso. El diagrama de flujo del funcionamiento del algoritmo de seguimiento se lo puede ver en la Figura 2.24.



**Figura 2.24** Situación de pérdida del algoritmo de seguimiento

Por ejemplo, el problema de oclusión se lo puede ver de mejor manera en la Figura 2.25, donde la persona se encuentra caminando frente a la cámara, luego queda ocluida por un objeto y el algoritmo termina perdiéndola.



**Figura 2.25** Pérdida de la persona por oclusión



En algunos casos el algoritmo de seguimiento es capaz de estimar la dirección de movimiento del objeto para poder localizarlo de nuevo; sin embargo, cuando la pérdida es completa se necesita volver a enfocarlo dentro del campo de visión y encerrarlo de nuevo en un bounding box y es aquí donde entra de nuevo la red de detección para realizar ese trabajo. Como se puede ver, existe la necesidad de implementar un código que fusione los dos tipos de algoritmos para que el dron pueda seguir a una persona de forma automática, sin la intervención del usuario en el proceso. La explicación y desarrollo del algoritmo híbrido creado se explican a continuación.

### **2.1.7 DESARROLLO DEL ALGORITMO HÍBRIDO**

El algoritmo central correspondiente a la visión debe cumplir con dos requisitos que son: realizar la detección de la persona y hacer el seguimiento de la misma. Al utilizar una computadora de placa reducida el factor limitante es la capacidad computacional que tiene la GPU por lo que para su implementación no se recomienda utilizar las redes de convolución como único elemento debido a la carga computacional a la que se sometería causando calentamiento excesivo y pérdida de velocidad. Este punto se complementa con el hecho de que las redes son capaces de detectar objetos en la imagen o conjunto de imágenes que se presenten, pero no realizan un seguimiento propiamente dicho porque no está dentro de su alcance o función. Por lo que, se decide realizar un algoritmo que cumpla con los requisitos planteados para este trabajo de titulación, que son: el algoritmo debe ser capaz de realizar la identificación de una persona específica para después realizar seguimiento de esta en el cuadro correspondiente a la imagen, ejecutarse en una computadora de placa reducida causando el menor impacto en su carga computacional, de manera que sea capaz de recuperarse y volver a encontrar a su objetivo en caso de interferencia u oclusión. Otro aspecto adicional importante es mantener la temperatura estable a un rango que permita trabajar sin ralentizar su funcionamiento.

Por esto se implementó un algoritmo híbrido que sea el encargado de elegir y alternar entre la detección y el seguimiento del objeto y se desarrolló una fusión de dos algoritmos, que originalmente tienen propósitos diferentes, para este fin común. En específico, se utiliza SSD MobileNet V2 para la parte correspondiente al reconocimiento de la persona, mientras que el algoritmo de seguimiento KCF es el encargado de realizar el seguimiento.

#### **2.1.7.1 Funcionamiento del algoritmo híbrido**

Este par de algoritmos trabajan en conjunto para sacar ventaja de las características de cada uno. Este proceso se realiza alternando su uso, siendo uno a la vez el protagonista y encargado de cumplir con su función. Existen dos eventos que causan la pérdida del objeto

desencadenando el cambio entre un algoritmo y el otro: la oclusión y el movimiento brusco de la cámara o del objeto.

Como se ha especificado anteriormente, para arrancar la etapa de trackeo o seguimiento es necesario delimitar la zona perteneciente a la fotografía o video en la que el algoritmo efectuará su operación; es decir, es necesario que se dibuje un rectángulo que delimite el área donde se encuentra el objeto de interés. Así mismo se ha explicado que la red neuronal convolucional (CNN) al identificar objetos genera una bounding box, la cual cumple con los parámetros de encerrar un objeto de interés en un rectángulo. Teniendo estos puntos en cuenta, es posible que el dibujo del área donde el trackeador efectuará su trabajo puede ser realizado de manera automática, siempre y cuando otro componente, en este caso la CNN, sea capaz de identificar al objeto y enviar la información necesaria para poder dibujar el rectángulo y comenzar el proceso de tracking.



**Figura 2.26** Cambio desde la etapa de detección a la de seguimiento

Tal como se puede ver en la Figura 2.26, la persona es identificada por el algoritmo de reconocimiento de objetos y dibuja el rectángulo que lo delimita o bounding box. Estos datos que vienen del algoritmo encargado de la identificación de objetos son usados como base para el inicio del algoritmo encargado del seguimiento.

Del mismo modo, en caso de que la etapa de seguimiento sufra de una oclusión o pérdida del objeto al que se encuentra siguiendo, se utiliza esta información como una bandera o disparador para que se ejecute nuevamente la etapa de identificación de objetos hasta que encuentre al objeto de estudio y envíe los datos del bounding box correspondiente y de esta manera arrancar el algoritmo de trackeo nuevamente.

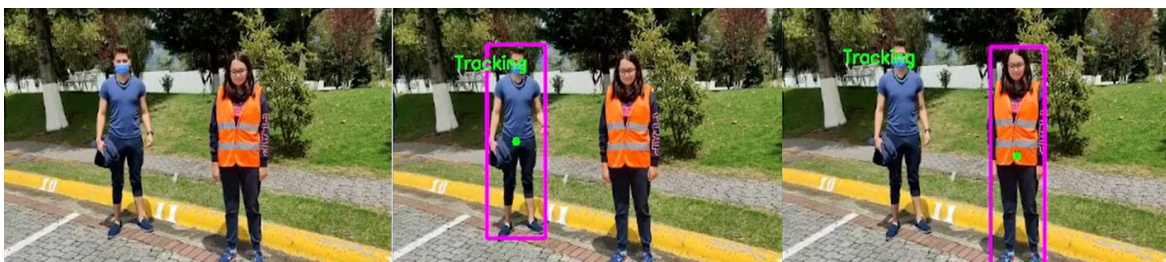
### **2.1.7.2 Consideraciones durante el desarrollo del algoritmo**

- Consideración 1: filtrado de un objeto entre varios de la misma clase en una imagen

Uno de los problemas principales encontrados en esta propuesta es que la red encargada de la identificación de objetos puede discriminar entre distintos elementos, pero en caso de presentarse diferentes objetos pertenecientes a la misma clase, como se puede ver en la Figura 2.27, por ejemplo, si están distintas personas en una misma imagen el algoritmo empezará a seguir a la persona a la cual encuentre primero. Esto es, no se tiene un control sobre la elección del objeto al que se desea seguir, dado el caso que se encuentren varios pertenecientes a la misma clase. Es por esto que se usa un chaleco de un color distintivo para realizar el filtro de color como factor extra, diferenciador entre distintos objetos de una misma clase.

Se analizará el área correspondiente en píxeles como una limitante para que el algoritmo detecte a la persona que va a seguir, es decir, si tiene suficiente área de un color distintivo, el algoritmo arrancará y seleccionará solo a esa persona pese a que existan más en la misma imagen como se puede ver en la imagen (c) de la Figura 2.27.

El área en píxeles que es ocupada por el chaleco de color distintivo es variable y depende de la altura, por lo que, al mantener la misma a un valor fijo es posible estimar el área necesaria para determinar la presencia de la persona que lo está portando.



**Figura 2.27** a) imagen de referencia, b) seguimiento sin filtro de color, c) seguimiento con filtro de color naranja.

Como es posible apreciar en la imagen sin el filtro de color, Figura 2.27 (b), reconoce primero a la persona ubicada en la izquierda, mientras que, con el filtro de color, Figura 2.27 (c), ignora a la persona que no utiliza el chaleco y el código arranca el seguimiento con la que está puesta el chaleco.

La detección del color distintivo se realiza de la siguiente forma. Primero, cuando se encuentra el objeto de la clase seleccionada se realiza el filtro de color explicado anteriormente en la Sección 2.1.5 y en caso de tener dentro del bounding box cierta área en píxeles perteneciente al color distintivo entonces se inicializa con esos datos al algoritmo encargado del seguimiento. Caso contrario de no cumplir con esta característica, se sigue

buscando continuamente otro elemento de la clase en cuestión que cumpla con la característica de tener en la imagen el color distintivo, en este caso el chaleco.

Para evitar posibles conflictos que se presenten cuando exista en la escena más de un objeto que sea del color distintivo propuesto, se decide segmentar la zona en la que se realiza el filtro de color. La zona de la imagen que será analizada por el filtro de color vendrá determinada por el algoritmo encargado del reconocimiento de objetos, es decir, solo va a buscar el color distintivo en la imagen que se encuentre dentro del bounding box correspondiente a una persona, esto puede verse de mejor manera en la Figura 2.28.



**Figura 2.28** Filtro de color dentro del bounding box

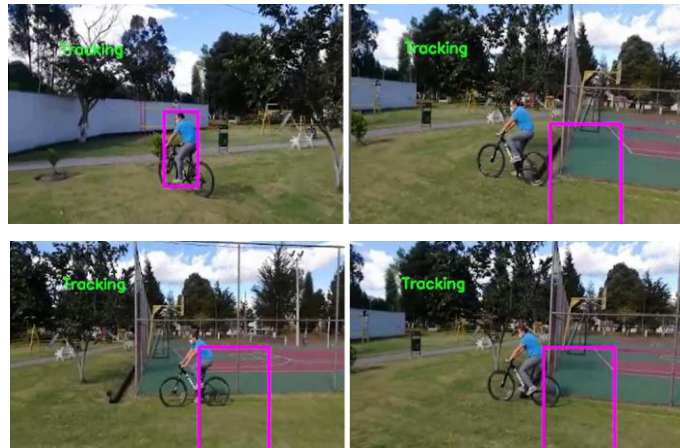
- Consideración 2: pérdida del objeto por movimientos desconocidos

Este problema radica en las propiedades intrínsecas de los algoritmos de seguimiento, puesto que en caso de que el objeto se mueva de una forma irregular que no cause su pérdida de escena o que se aleje, existe la posibilidad de que el algoritmo no sea capaz de identificar este movimiento y se quede enfocado a un sitio donde la persona ya no se encuentra. Este problema puede ser evidenciado en la Figura 2.29, donde la persona primero se encuentra en una posición central, pero se va alejando de la posición de la cámara.



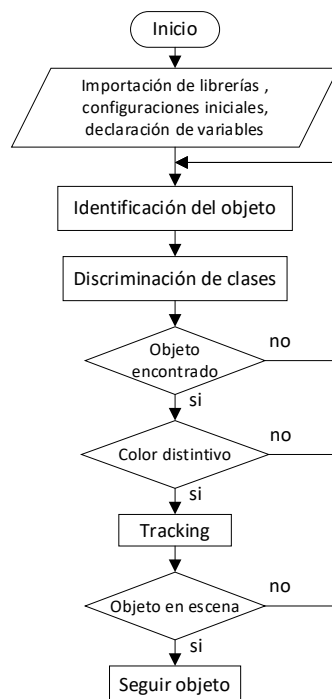
**Figura 2.29** Problema con el área del bounding box

Al alejarse el área que el trackeador utiliza para ejecutar el seguimiento empieza a ocupar más espacio que el de la persona en sí, causando que solo se realice el seguimiento a parte de la persona. Por esto, el cuadrado empieza a encerrar a gran parte de la imagen que no corresponde a la persona, lo que se aprecia en la Figura 2.30. Debido a este error existente en la estructura interna de los algoritmos de tracking se quedaría seleccionada la zona que no corresponde al objeto.



**Figura 2.30** Bounding box encerrando zona incorrecta

Para solucionar estos problemas se tiene la etapa de detección de objetos que le permite al prototipo manejar este tipo de inconvenientes, es decir, que vuelva a encontrar al objeto dibujando una nueva área de trackeo y que vuelva a iniciar la etapa de seguimiento.



**Figura 2.31** Diagrama de flujo del algoritmo híbrido de visión implementado

Esto viene representado por las distintas flechas de decisión correspondientes al No de la Figura 2.31, donde si se presenta alguna de estas condiciones: no encontrar al objeto, no tener el color distintivo o perderlo de la escena, el algoritmo regresa hacia la etapa de identificación de objetos.

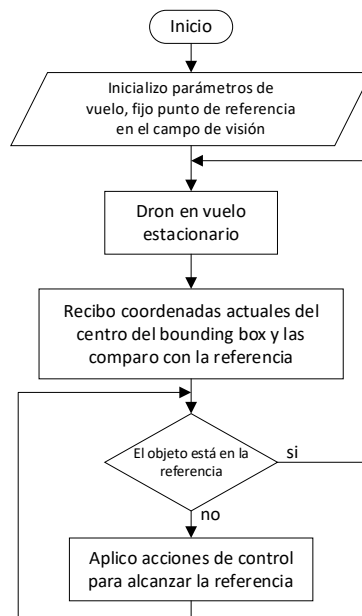
### **2.1.7.3 Descripción completa del algoritmo híbrido.**

Después de haber explicado el funcionamiento y desarrollo del algoritmo híbrido propuesto junto con las consideraciones que se deben tener y las partes que lo conforman, es necesario hacer una recapitulación desde el inicio de ejecución de este algoritmo, paso a paso, y proporcionando detalles extra que no se encuentran en el diagrama de la Figura 2.31 para una mejor comprensión del mismo como un todo.

- Antes de ejecutar el programa, se tienen ciertas consideraciones que son la selección del objeto que se va a seguir (en este caso a una persona) y la configuración del filtro de color (naranja) junto con el área que ocupa en píxeles el color distintivo (área variable dependiendo de la altura de vuelo y distancia de la persona al dron).
- Cuando el proceso inicia, entra en ejecución el proceso de adquisición del video (Sección 2.1.2) desde la cámara para presentarlo
- El video es enviado hacia la red neural la que se encarga de procesarlo y obtener como resultado los porcentajes de seguridad de cada clase que aparezca en escena (Sección 2.1.3 y Sección 2.1.4).
- Internamente se realiza la discriminación con respecto a la clase seleccionada anteriormente y solo se despliega en pantalla el bounding box y el porcentaje correspondientes a la clase en cuestión (Sección 2.1.4.1).
- En caso de no encontrar al objeto se sigue enviando y procesando el video en la red para obtener la situación en donde el objeto perteneciente a la clase seleccionada aparezca en el rango de visión de la cámara.
- Cuando el objeto es encontrado se extraen los parámetros correspondientes a la zona de la imagen donde se encuentra el objeto, valores necesarios para la ubicación y dibujo del bounding box (Sección 2.1.4.2).
- La imagen dentro del bounding box es enviada para ser procesada por el filtro de color, donde se determina si este cumple con la condición de tener suficiente área en píxeles correspondientes al color distintivo (Sección 2.1.5).

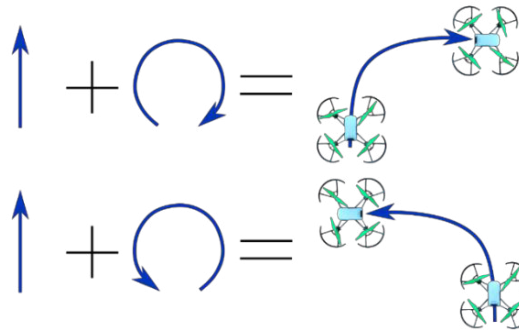
- Cumpliendo la condición de ser el objeto correspondiente a la clase seleccionada y contar con el color distintivo, se envían los datos del bounding box para el seteo de los parámetros iniciales con los que arranca el algoritmo de seguimiento (Sección 2.1.7).
- El algoritmo de seguimiento se encarga de actualizar los valores correspondientes al bounding box para poder seguir al objeto en la imagen.
- Cuando el objeto se pierde sea por oclusión o salir de escena del video se repite todo el procedimiento antes mencionado.

## 2.2 ALGORITMO DE CONTROL



**Figura 2.32** Diagrama general del algoritmo de control

Por la información presentada por los fabricantes de drones, se sabe que gran cantidad de empresas desarrolladoras de UAVs y de autopilotos trabajan con controladores de tipo PID, siempre teniendo buenos resultados. Por esto se plantea realizar el control para el seguimiento de la persona con dos controladores de este tipo, uno que se encargue de desplazar a la aeronave hacia adelante y hacia atrás (Pitching), junto con otro que controle la rotación del dron en su propio eje (Yaw), para que pueda completar el movimiento necesario y así seguir a una persona. Este movimiento es posible apreciarlo de mejor manera en la Figura 2.33, donde se observa que, para seguir a una persona en la tierra, es necesario solamente seguir hacia adelante y rotar sobre el propio eje cuando sea necesario para mantener a la persona en el rango de visión de la cámara. En la Figura 2.32 se puede ver el diagrama de flujo general del algoritmo de control implementado.



**Figura 2.33** Movimientos que realiza el dron para seguir a su objetivo

### 2.2.1 EJES DE CONTROL

El control del dron para el seguimiento de la persona depende netamente del sensor capaz de captar la imagen. Como se indicó en la Sección 1.3.3, el dron cuenta con la cámara de 5 MPX@720p que sirve como elemento para realimentación de la información en el lazo de control. Al trabajar con imágenes captadas con esta cámara, es necesario utilizar o idear un método que permita la creación de una referencia para que el controlador tome las acciones correspondientes. Por esta razón se decide utilizar la imagen obtenida y de manera similar al dibujo del bounding box orientar a toda la imagen como un sistema de coordenadas donde cada píxel tiene su ubicación en el plano correspondiente a la cámara.

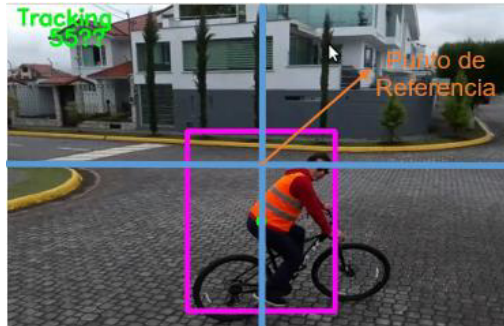
Para tener un parámetro de comparación entre la referencia y el objeto a seguir, se decide tomar el centro del bounding box como punto clave para realizar el control (véase la Figura 2.34), es decir, se deben realizar acciones con el dron para que el centro del objeto (centro del bounding box) se mantenga en la referencia de la imagen.



**Figura 2.34** Centro del bounding box como punto clave

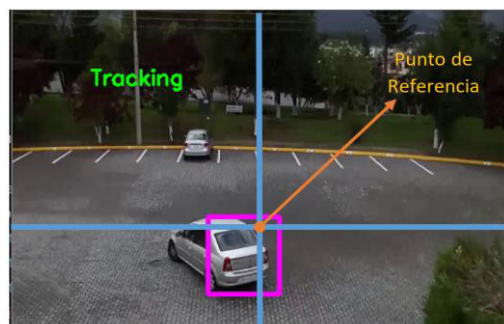
Para una mejor explicación se utiliza la Figura 2.35 como referencia. Como es correcto pensar en el eje de las X la imagen tendría que estar en el centro para su seguimiento (píxel 240). Por lo que sería necesario controlar la rotación del dron para mantener al objeto en el centro.





**Figura 2.35** Punto de referencia en el centro del eje X y Y

En el caso del eje de las Y se plantea no utilizar el centro del eje, puesto que la cámara del dron enfoca hacia abajo. Si se emplea como punto de referencia tanto el centro del eje X como el de las Y se tendría que la misma está muy alta con respecto al centro del bounding box y debido a que no se está realizando un control sobre la altura del dron, no sería posible llegar a la referencia. Es decir que, para que el objeto esté en el punto medio de la imagen se necesita que el dron esté muy lejos del mismo haciendo difícil su reconocimiento en caso de que se salga de la etapa de seguimiento por algún motivo. Para visibilizar un ejemplo de este caso se utiliza la Figura 2.35, en donde la persona se encuentra por debajo de la referencia haciendo que el dron se mueva hacia atrás intentando hacerla coincidir el centro del bounding box causando que la distancia entre la cámara y la persona sea muy grande. Tomando en cuenta este factor se decidió colocar como referencia a la tercera parte del tamaño total de la fotografía en el eje Y (píxel 213) y mantener el centro del eje X (píxel 240). Ambas decisiones dan como resultado el “Punto de referencia” representado en la Figura 2.36 haciendo que el dron no se aleje tanto del objeto para hacer coincidir el centro del bounding box con el punto señalado.

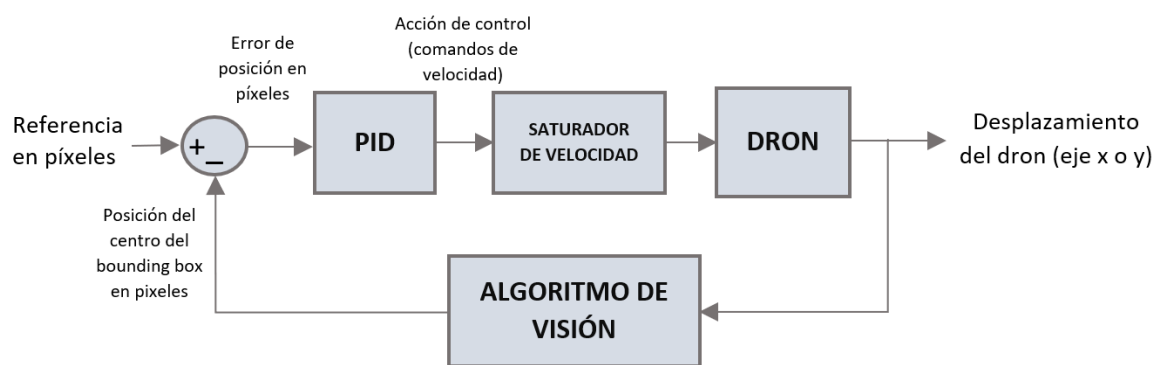


**Figura 2.36** Punto de referencia en el campo de visión del dron

## 2.2.2 INTERACCIÓN DEL ALGORITMO DE VISIÓN CON EL DE CONTROL

El algoritmo de visión es parte fundamental del lazo de control puesto que gracias a este es posible obtener los parámetros enviados hacia el algoritmo de control, es decir, es

posible obtener los errores o las diferencias en píxeles entre el punto de referencia utilizado para centrar la imagen con el centro del bounding box del objeto. Para su interacción hay que tener claro que existen errores en el eje de las X tanto como en el de las Y por lo que es necesario tener dos controladores que permitan centrar el objeto. En este lazo, el algoritmo de visión hace el papel de sensor que realimenta al sistema, este entrega la posición en píxeles del centro del bounding box al restador y así obtener la señal de error que ingresa al controlador, este a su vez entrega las acciones de control al dron para que realice los desplazamientos necesarios y así poder seguir a una persona de manera automática. Véase la Figura 2.37.



**Figura 2.37** Interacción del algoritmo de control con el de visión.

### 2.2.3 CONSIDERACIONES EN EL CONTROL

Cabe mencionar que el dron al ser de tamaño reducido y de prestaciones limitadas no cuenta con elementos que permitan realizar un mejor control. Por ejemplo, este dron cuenta con pequeños motores DC que permiten que vuele, pero no existe una manera de sobre exigirlos para que sobrepasen la velocidad máxima de 8m/s. Además, la aeronave moviéndose a dicha velocidad no es capaz de luchar con un viento fuerte por lo que, aunque se envíe la señal de control al máximo, no se lograría mantener el seguimiento.

Para poder superar esta situación desfavorable se decide utilizar valores tope o valores de saturación. Esta decisión se toma porque al superar la velocidad normal de operación en el dron se pierde totalmente la estabilización por software que tiene la cámara. Por tanto, se decide colocar valores limitadores en la velocidad hasta 100 cm/s que permitan su funcionamiento y poder seguir a la persona sin perder la estabilización del video. Al trabajar hasta este punto de velocidad se la considera como el modo normal de operación, si se supera este punto el dron entra a trabajar en el modo sport donde se pierde la estabilización del video.

Otro requerimiento en el lazo de control surge cuando la persona a seguir circula a una velocidad mayor a la que el dron puede responder sin afectar a la estabilización de la imagen. Siendo un sistema de control que tiene como único sensor para la realimentación del lazo su cámara a bordo es primordial tener una buena estabilización en la imagen obtenida, la cual es afectada de forma directa cuando la velocidad de movimiento del objeto a seguir es alta y provoca movimientos sumamente bruscos en la imagen obtenida.

Si la persona a la que el dron se encuentra siguiendo supera la velocidad colocada como limitador en el movimiento para no perder la estabilización del video, va a causar que la misma salga del rango de visión, por lo que, la alternativa propuesta es que al presentarse esta situación el dron se quede estático en su última posición. Cuando la persona de interés vuelva a aparecer en la escena, este va a ser capaz de seguirla de nuevo; mientras que si no la encuentra se quedará sobrevolando hasta que se envíe la señal de aterrizar.

## **2.3 CONEXIÓN ENTRE LA RASPBERRY PI 4 Y EL DRON**

Gracias a que el dron DJI Ryze Tello cuenta con dos antenas de 2.4 GHz y la Raspberry Pi 4 tiene antenas tanto de 2.4 GHz y de 5 GHz, la posibilidad de la conexión existe y por tanto se la realiza obteniendo un enlace inalámbrico que puede llegar hasta los 30 metros, sin obstáculos y en línea de vista. Adicionalmente, para la conexión entre estos dos principales dispositivos es necesario la creación de un código que permita enviar y recibir comandos entre ambos.

Justamente, entre las ventajas que tiene el dron DJI Ryze Tello es que existe un SDK desarrollado por los mismos fabricantes para enviar comandos simples. Gracias a conocer la existencia de este paquete, se obtiene información pertinente al envío de comandos y como se lo puede emplear para establecer una conexión, mediante WiFi, con una computadora. Para realizar esta conexión es necesario crear un código en Python y utilizar la librería *djitellopy* junto con ciertos complementos. La tarjeta Raspberry Pi 4 cuenta con un compilador para Python 3 integrado que se llama Thonny, el cual es el encargado de ejecutar códigos mediante una interfaz limpia y amigable que consume pocos recursos. Para trabajar con el DJI Ryze Tello es necesario instalar la librería antes mencionada (*djitellopy*) que permite el envío y recepción de datos hacia el dron en los que constan:

- El video propiamente dicho, el cual permite utilizar la cámara a bordo del dron para enfocar a la persona de interés y mediante a la información obtenida tomar acciones.

- Datos del estado de la batería. Estos datos son relevantes puesto que el sistema del dron al estar menos del diez por ciento de carga aterriza de manera automática si se encuentra en vuelo y si está en la tierra no permite su despegue.

Los datos enviados hacia el dron corresponden a:

- Los comandos utilizados para al despegue y aterrizaje automáticos de la aeronave.
- La señal que habilita la recepción del video de la cámara abordo.
- Las indicaciones sobre la velocidad en el eje de desplazamiento delantero y trasero junto con las que permiten al dron que rote sobre su propio eje, información que es proporcionada por el lazo de control y enviada hacia la aeronave.

### 2.3.1 DESARROLLO DE LA INTERFAZ

Para la comunicación del usuario con el código es necesario la utilización de un sistema que permita cumplir con este objetivo, por lo que se propone una interfaz mayormente realizada en software que es complementado por dos componentes de hardware que facilitan esta interacción de la persona con el código programado, de una manera simple.

#### 2.3.1.1 Hardware

Por el lado del Hardware para que el dispositivo sea portátil se decide utilizar una pantalla propia de los desarrolladores del sistema Raspberry, puesto que con este se tiene una gran sinergia permitiendo mostrar los datos y dando a la interfaz la capacidad de ser manejada mediante una pantalla táctil. La pantalla utilizada es conocida con el nombre de “Raspberry Pi 7” Touch Display” la cual permite a la Raspberry funcionar como una especie de Tablet gracias a su pantalla de siete pulgadas y una resolución de 800x400. La pantalla se conecta mediante los pines del puerto GPIO de la tarjeta, para su transmisión de datos y alimentación. Este dispositivo se puede adquirir con su case plástico oficial para una correcta instalación con la Raspberry Pi 4 como se puede ver en la Figura 2.38.



**Figura 2.38** Raspberry Pi Touch Display + Case protector

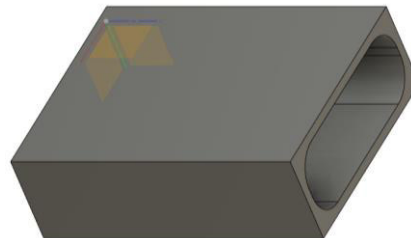
Para la alimentación del sistema se utiliza una batería genérica de 10000 mA (Figura 2.39) con una salida de 2.5 amperios que permiten al conjunto formado por la SBC y la pantalla funcionar con normalidad.



**Figura 2.39** Power Bank de 10000mA

Para un mejor acople de la batería con la pantalla se decide hacer un modelado para imprimir en plástico PLA para que la batería se quede fija y permita al sistema ser transportado de manera fácil, manteniendo al equipo portátil.

Se plantea entonces la impresión del modelo encontrado en la Figura 2.40, este modelo fue realizado y diseñado para el desarrollo del proyecto para poder transportar la batería y que sirva de base a la Raspberry Pi 4. El modelo cual será colocado en la parte inferior de la pantalla para tener acceso a todos los puertos de la Raspberry en caso de que se necesite y que permita al mismo tiempo ser un soporte y base del sistema cuando se coloque la batería en el mismo. El elemento fue diseñado en el software conocido como Autodesk Fusion 360.



**Figura 2.40** Case diseñado en 3D para el power bank

Finalmente se imprime el elemento antes mencionado y se lo acopla al sistema de pantalla y Raspberry Pi 4. En la Figura 2.41 es posible observar los elementos antes mencionados como: la pantalla conectada a la Raspberry Pi 4 y ambas siendo alimentadas por la batería mediante un cable USB tipo C.

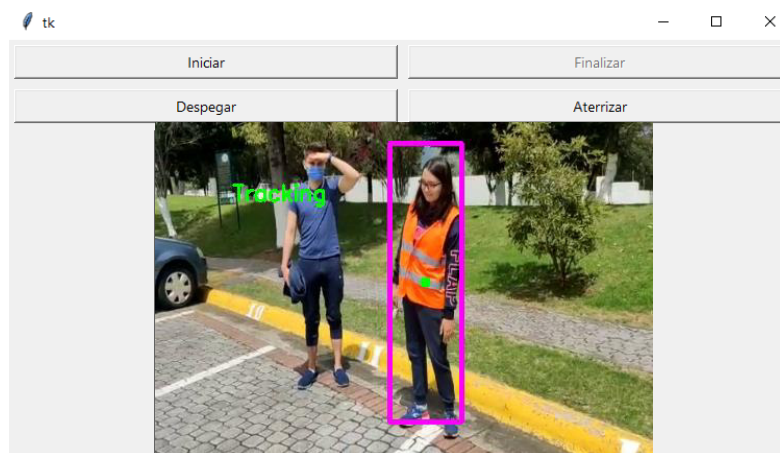


**Figura 2.41** Prototipo final implementado

### 2.3.1.2 Software Tkinter

Para el desarrollo de la interfaz se decide utilizar la librería perteneciente a Python conocida como Tkinter. Generalmente viene incluida al realizar la instalación de Python en su gran mayoría de versiones, sin embargo, en ciertos casos hace falta realizar su instalación de forma manual ejecutando la línea de comando `pip install Tk` en una ventana de terminal. Esta librería permite crear GUIs de manera fácil mediante el uso de programación importándola con la línea `from tkinter import *` al código y la interacción con el usuario se la ejecuta mediante el uso de botones o con la presentación de cuadros de diálogo, texto o imágenes [45].

Para su ejecución, Tkinter se basa en presentar una ventana que funciona mediante un lazo el cual muestra los objetos que se hayan programado para interactuar con el usuario. Este lazo permite que la ventana se encuentre en constante lectura asemejando a un lazo “while True” donde siempre se encuentra en constante supervisión para detectar si alguno de los botones cambia de estado.



**Figura 2.42** Interfaz realizada para el prototipo

En la Figura 2.42 se presenta la interfaz propuesta para este proyecto que consta básicamente de cuatro botones:

- *Iniciar*: Este botón permite correr el código causando que se ejecute el programa con el algoritmo mejorado que permite el reconocimiento y seguimiento de la persona de interés, adicionalmente a esto activa la transmisión de video desde el dron.
- *Despegar*: Botón que permite que la aeronave despegue y se quede en el aire sin desplazarse en otras direcciones.

- *Finalizar*: Termina la ejecución del programa, cierra la transmisión de video y aterriza la aeronave.
- *Aterrizar*: Ejecuta la acción de aterrizaje del dron en caso de emergencia.

Para un correcto funcionamiento del prototipo los botones deben ser ejecutados primero realizando el despegue y después el botón de Iniciar; mientras que, los botones de aterrizaje y el de finalizar pueden ser presionados en cualquier momento.

### **3 PRUEBAS, RESULTADOS Y DISCUSIÓN**

En este capítulo se presentan los resultados de las pruebas realizadas para comprobar el funcionamiento tanto de los algoritmos desarrollados como de todo el sistema que conforma el prototipo final entregable. Se inicia describiendo las pruebas que se realizaron respecto al algoritmo híbrido de visión, empezando desde los relacionados con la selección de los algoritmos de detección y seguimiento hasta las pruebas de rendimiento y funcionamiento al fusionar ambos. Luego se describen las pruebas que se realizaron para obtener una conexión estable entre la Raspberry Pi 4 y el dron y conseguir un control básico de la aeronave, ya que si esta falla se perdería la comunicación entre las dos partes quedando el dron a la deriva pudiendo ocasionar un accidente.

En tercer lugar, se describen las pruebas que se realizaron para obtener las constantes de los PID implementados en el algoritmo de control, y también para evaluar al funcionamiento del sistema en conjunto. Este algoritmo es el encargado de comandar al dron para que pueda seguir a la persona con chaleco de color naranja, esto acorde al procesamiento y la respuesta obtenido a través de la utilización del algoritmo híbrido de visión. Finalmente se describen las circunstancias adecuadas en las que el prototipo final tendrá un funcionamiento correcto, especificando de mejor manera las condiciones de operación del prototipo a las que se llegó luego de que se realizaran todas las pruebas.

#### **3.1 PRUEBAS DEL ALGORITMO HÍBRIDO DE VISIÓN POR COMPUTADOR**

Como se explicó en la Sección 2.1, el algoritmo híbrido de visión por computador implementado consta de dos etapas: detección y seguimiento. En cada etapa se realizaron diferentes pruebas que ayudaron a determinar los algoritmos más adecuados para cada una de ellas y que faciliten la integración del sistema propuesto en este trabajo.

Las pruebas en esta sección se realizaron en una computadora portátil con la finalidad de tener un punto de partida o una referencia acerca del comportamiento que se espera de estos algoritmos cuando se los implemente en la Raspberry Pi 4. Esto debido a que si se las hiciera directamente en la SBC el tiempo de procesamiento sería muy extenso por los pocos recursos computacionales con los que se contaría y los resultados sirvieron como referencia para su posterior ejecución en la Raspberry Pi 4.



### 3.1.1 ETAPA DE DETECCIÓN

Como se ha mencionado anteriormente se descartó el uso de las Haar Cascade por los errores que se tiene al trabajar con estas y se optó por utilizar redes de detección de objetos, sin embargo, para su análisis se decide primero utilizar la computadora y luego la Raspberry Pi 4 por mayor facilidad.

En redes de detección de objetos, la cantidad de alternativas se reduce principalmente porque es necesario que el algoritmo sea capaz de ejecutarse en la Raspberry Pi 4 y que sea capaz de un procesamiento de video en tiempo real lo cual limita la cantidad de candidatos. Por tanto, la decisión debe tomarse entre Tiny-Yolo V2 y SSD MobileNet V2 porque las Haar Cascades tienen un mayor tiempo de ejecución y el reconocimiento de los objetos se ve afectado por la posición o postura que estos tengan (como ya se explicó previamente en la Sección 2.1.4). En particular, para la aplicación propuesta la persona a la que se va a seguir no se encontrará en una misma posición todo el tiempo, por lo que una mejor alternativa es optar por la identificación de objetos basada en redes neuronales.

#### 3.1.1.1 Comparación entre los algoritmos de detección

Como se mencionó al inicio de este capítulo, para comparar el funcionamiento de estas dos redes neuronales se realizaron pruebas en una laptop. La muestra para la prueba será un conjunto de 120 fotos con 262 objetos de diferentes clases (personas, autos, bicicletas, perros, etc.). Se optó por realizar videos con estas imágenes para procesarlos con ambas redes y comparar la cantidad de objetos identificados con el número obtenido en el reconocimiento manual realizado previamente. De esta forma se puede obtener un porcentaje de precisión de las redes, así como los fps a los que se ejecutan. Los resultados se presentan en la Tabla 3.1 y los videos utilizados para estas pruebas están disponibles en Youtube ([46]).

**Tabla 3.1** Resultado de las pruebas realizadas a las redes en computador portátil

Parámetros	Tiny Yolo V2	SSD MobileNet V2
Fps de salida	23	20
Precisión	56.95%	70.45%

#### 3.1.1.2 Selección del algoritmo de detección

En base al resultado anterior de la Tabla 3.1, se podría asumir que la mejor opción es la red Tiny Yolo V2, por su rapidez. Sin embargo, el factor que corresponde a la precisión para reconocer objetos es menor en comparación con la SSD MobileNet V2; por lo que, si

se desea mayor confianza en el prototipo se debería optar por esta última. Con el fin de seleccionar el algoritmo de detección a emplear, se realizó la misma prueba pero en la Raspberry Pi 4, puesto que esta SBC no cuenta con las características y prestaciones que tiene una laptop. Para esta prueba se utilizó el mismo conjunto de imágenes procesado en la computadora y sus resultados se muestran en la Tabla 3.2

**Tabla 3.2** Resultado de las pruebas realizadas a las redes en la Raspberry Pi 4

<b>Parámetros</b>	<b>Tiny Yolo V2</b>	<b>SSD MobileNet V2</b>
Fps de salida	1	3
Precisión	56.95%	70.45%

En esta tabla podemos observar que la diferencia de fps al ejecutar estas redes en la Raspberry Pi 4 se invierte en comparación a los resultados obtenidos en la computadora, esto es debido a que la SBC no cuenta con una tarjeta gráfica tan potente como si lo tiene una PC. Justamente por esto el rendimiento de estas dos redes se basa más en la capacidad de su procesador y este es mejor aprovechado por la SSD MobileNet V2. La velocidad para reconocer objetos será un parámetro esencial en el proyecto puesto que de esto depende gran parte del algoritmo híbrido propuesto.

De las pruebas realizadas, la SSD MobileNet V2 es tres veces más rápida a la obtenida en la Tiny Yolo V2, tomando como unidad de medida los fps, ya que muestra de buena forma la rapidez con la que el video sale de la red sea si esta encontró algún objeto o no, es decir, que no se tiene variación según la cantidad de objetos presentes en la imagen. Esto representa una diferencia importante porque se tendrá una imagen más fluida que se puede catalogar como “en vivo” enviada desde la cámara del dron. De tener un mayor tiempo de procesamiento o una menor velocidad de ejecución significaría un problema al momento de realizar el control o maniobrar la aeronave.

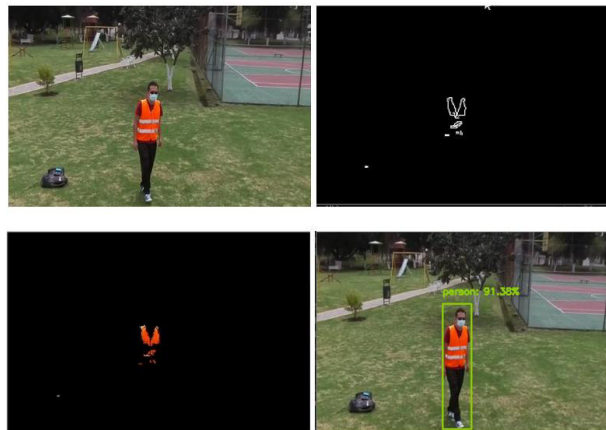
Otro parámetro que se tuvo en cuenta para la selección de la red es la precisión, como se puede observar de la Tabla 3.1 y Tabla 3.2 es el mismo porcentaje que se obtuvo al realizar la prueba en una laptop. Esto se debe a que la SSD MobileNet V2 no varía su precisión por la falta de recursos computacionales. Esta es una característica propia de cada red y como se observa, la SSD MobileNet V2 tuvo un resultado 13.5% mejor que la Tiny Yolo V2.

En base a los resultados presentados en esta sección se optó por elegir la SSD MobileNet V2 como la más adecuada para este trabajo, debido a que el algoritmo completo debe ejecutarse en la Raspberry Pi 4 para contar con un equipo portátil.

### 3.1.1.3 Filtro de color

Luego de añadir el filtro detallado en la Sección 2.1.7.2 a la etapa de detección se procedió a hacer pruebas con diferentes videos pregrabados de personas portando el chaleco para acondicionar los valores del filtro HSV. Este parámetro será la marca de “color distintivo” que permite al código diferenciar a un objeto específico de otros de la misma clase y mediante procesamiento de imagen es posible obtener el área que ocupa en píxeles de acuerdo con la altura a la que se esté sobrevolando.

Para la prueba se toma un video grabado a 2,5 metros de altura donde una persona se encuentra usando el chaleco naranja como se observa en la Figura 3.1. Cada vez que reconozca a la persona, se toma la parte de la imagen que se encuentra dentro del bounding box y se analiza el área que corresponde en píxeles al color para el que se ha configurado el filtro, luego se almacenan los datos correspondientes al área en un documento con formato .csv.



**Figura 3.1** Ejemplo de prueba realizada para el cálculo del área del color distintivo

Se realizó un sobrevuelo de la aeronave a alturas entre 2.5 y 3.5 metros, en un día con buena iluminación. Se observó que a menor altura se tiene una mayor área, mientras que a una mayor altura esta variable disminuye. Se tomaron 431 muestras y al realizar un promedio de los datos se obtuvo como resultado 110 píxeles. Este número es utilizado como indicador del área en píxeles que debe tener el color naranja dentro del bounding box que encierra a la persona para considerarse que la misma es a la que el dron debe seguir.

#### 3.1.1.3.1 Consideraciones a tener en cuenta

- El código propuesto hace que la aeronave se eleve a una distancia superior a los dos metros y medio de altura, esto para que pueda seguir a la persona sin que se

corra el riesgo de que el dron se estrelle con ella. Esta altura también es la utilizada en las pruebas para la configuración del área del color distintivo dentro del bounding box por lo que en caso variarla será necesario modificar este parámetro en el código y si no se lo hace, el algoritmo propuesto no será capaz de arrancar la etapa de seguimiento.

- Debido a que el color es el reflejo de la luz, el parámetro correspondiente al color es alterado de acuerdo a la hora del día, esto porque en la tarde la iluminación disminuye y el filtro de color está configurado para detectar el naranja brillante. Por lo que, cuando el sol se empieza a poner pasado las cuatro de la tarde dejaría de funcionar de manera correcta y sería necesario una nueva identificación de valores para el filtro de color. Por esto se recomienda utilizar el sistema cuando el sol presente una buena iluminación, desafortunadamente el trabajar con visión computacional en entornos al aire libre se sufre por complejas características que se deben encarar como lo son los fondos dinámicos (como árboles en movimiento, agua corriendo, etc) y principalmente los cambios de iluminación que son algunas de los eventos más difíciles de clasificar correctamente [47].
- Del mismo modo, al tener un día muy soleado este parámetro se ve afectado ya que el chaleco es reflectivo y en lugar de presentar colores naranjas vivos presenta tonos cercanos al blanco o al amarillo cuando la persona se encuentra de cara al sol.

Con la información anteriormente expuesta se decide optar por seleccionar tres distintas configuraciones del filtro HSV y de esta manera obtener una mejor detección de color de acuerdo a las condiciones ambientales que se encuentren al momento de realizar el ensayo, cada una se diferencia y adecúa dependiendo del ambiente exterior en el que se va a trabajar con el prototipo: día muy soleado, día con buena iluminación (no necesariamente soleado) y finalmente días nublados.

En la Tabla 3.3 se muestran las configuraciones o rangos del filtro HSV programados para las distintas condiciones ambientales.

**Tabla 3.3** Ambientes considerados para la configuración del filtro HSV

<b>Ambiente exterior</b>	<b>Límite bajo</b>	<b>Límite alto</b>
Nublado	[0, 110, 230]	[255, 255, 255]
Buena iluminación	[0, 110, 200]	[155, 255, 255]
Soleado	[0, 109, 115]	[75, 255, 255]

Mediante el código de forma manual es posible seleccionar cualquiera de estos ambientes predefinidos comentando las líneas que no correspondan al necesario, de esta manera el prototipo funcionaría de forma más robusta.

- Como última observación, se considera necesario aclarar que la cámara del dron no es de buena calidad ya que al ser una aeronave económica tiene prestaciones de hardware y software limitadas, es decir que no cuenta con las características comunes en los drones de alta gama (autoenfoco o una buena interpretación de los colores). Tanto el contraste como el brillo se ven opacados o saturados al trabajar con el vídeo de esta cámara, esto se da porque la estabilización de imagen se realiza por software reduciendo la calidad visual que se obtiene en la imagen y es una diferencia notable al comparar un video grabado por el dron y uno grabado mediante un celular. Esto se puede evidenciar en la Figura 3.2.



**Figura 3.2** Imagen izquierda tomada con el celular, Imagen derecha tomada con el dron

### 3.1.2 ETAPA DE SEGUIMIENTO

En esta etapa se realizaron pruebas con los dos algoritmos de seguimiento disponibles en OpenCV, que son el KCF y el TLD, que fueron explicados en la Sección 1.3.6 y resumidos, por ejemplo, en la Tabla 2.1.

Durante el desarrollo de este trabajo se procesaron 20 videos de 10 segundos de duración promedio, en los que el algoritmo debe seguir a una persona. Para estas pruebas no fue necesario aún el uso chaleco de color naranja, puesto que se desea ver las capacidades que tiene el algoritmo por sí solo para el seguimiento, siendo el usuario el que encierre manualmente a la persona dentro del bounding box.

#### 3.1.2.1 Comparación entre algoritmos de seguimiento

Dado que en la bibliografía revisada se encontró que los dos algoritmos KCF y TLD son los más adecuados para trabajar en una SBC, por lo que no fue necesario realizar pruebas en la computadora sino directamente en la Raspberry Pi 4 y así determinar cuál es la mejor opción para este trabajo.

En la Tabla 3.4 se muestran los resultados que se obtuvieron al ejecutar los dos trackeadores. Las pruebas se llevaron a cabo procesando los videos al aire libre de diez segundos donde una persona aparece en escena, después de un tiempo desaparece detrás de un objeto y luego vuelve a reaparecer. Los datos se obtienen analizando el tiempo que se rastreó a la persona sobre el tiempo total que estuvo en la escena del video. De esta forma se consigue expresar el tiempo de seguimiento como un porcentaje en relación con el tiempo total que debía realizar el seguimiento lo que sirve para una mejor comparación entre ambas alternativas. El valor porcentual de falsos positivos corresponde a cuando el algoritmo se encontraba realizando el seguimiento, pero la persona ya no se encontraba en escena o a su vez no era encerrada por el bounding box.

**Tabla 3.4** Resultado de las pruebas realizadas a los trackeadores en una Raspberry Pi 4.

Parámetros	TLD	KCF
Fps de salida	10	29
Seguimiento del objeto	67.25 %	70.30 %
Falsos positivos	34.73 %	5.12 %

Gracias a los resultados es posible aseverar que la velocidad del algoritmo KCF es más de dos veces superior a la del TLD, obteniendo una mayor cantidad de fotogramas por segundo. Para un mejor entendimiento se decide colocar la Figura 3.3 y la Figura 3.4 donde es posible apreciar el comportamiento de ambos algoritmos en las pruebas.



**Figura 3.3** Pruebas con el algoritmo de seguimiento TLD



**Figura 3.4** Pruebas con el algoritmo de seguimiento KCF

También se observa que el algoritmo TLD es capaz de recuperarse de la oclusión, ya que puede volver a encontrar el objeto seleccionado. Sin embargo, se tiene el problema de que desde el inicio no mantiene el seguimiento a la persona de manera correcta, dando falsos positivos en gran cantidad. En el transcurso de todo el video el TLD va cambiando el bounding box, saltando y encerrando diferentes regiones de la imagen y no envía una señal de pérdida, sino que intenta siempre señalar algo que puede o no ser el objeto a seguir, por lo que se obtienen resultados muy desfavorables en un seguimiento continuo. Por otro lado, el algoritmo KCF también presenta cierta cantidad de falsos positivos, pero mayormente corresponden a cuando el objeto empieza a ser ocluido y cuando esto pasa, el algoritmo informa o alerta de la pérdida y no vuelve a encerrar nada durante el resto del video.

### **3.1.2.2 Selección del algoritmo de seguimiento**

La diferencia de fps entre los dos algoritmos es porque el TLD tiene que completar las tres fases, incluidas la detección y el aprendizaje que pueden no ser precisas, esto consume más recursos en comparación con el KCF. Justamente, el TLD permite rastrear, detectar e incluye capacidades de aprendizaje, además, es capaz de recuperarse de la oclusión, pero devuelve un número considerable de falsos positivos. Pese a que el algoritmo TLD tiene la ventaja de poderse recuperar de la oclusión en ciertas ocasiones, es posible tomar la característica del algoritmo de seguimiento KCF como una ventaja puesto que la información de cambio de estado que brinda al perder el objeto es el disparador perfecto de la fase de detección que se plantea como complemento para el desarrollo de este algoritmo híbrido a ser ejecutado en la Raspberry Pi 4.

Por consiguiente, se determinó que el algoritmo KCF es el más adecuado para esta aplicación gracias a su velocidad y precisión al momento de seguir a cualquier objeto, la baja cantidad de falsos positivos que presenta y la bandera de objeto perdido que puede trabajar como disparador de la etapa de detección cuando ocurra una oclusión o el objeto salga del campo de visión.

## **3.2 PRUEBAS DE IMPLEMENTACIÓN DEL ALGORITMO HIBRIDO DE VISIÓN EN LA RASPBERRY PI 4 JUNTO CON EL DRON**

Para la conexión de la tarjeta con el dron se decide avanzar de manera progresiva, es decir el primer paso es establecer la conexión con el dron. Para este punto se debe ubicar la red Wi-Fi creada por el DJI Ryze Tello y se establece la conexión con la Raspberry Pi 4 como

si se tratara de cualquier otra red. La única diferencia es que es una conexión directa con el dron sin cables y sin acceso a internet.

Después de lograr la conexión es necesario realizar el envío y recepción de datos desde la aeronave a la SBC y viceversa, para esto se hace uso de la librería de Python *djitellopy* descrita en la Sección 2.1.1.2 y el comando *me.connect()* en el código. Luego se plantean diferentes tipos de pruebas que serán detalladas de mejor manera más adelante.

### 3.2.1 VUELO DEL DRON

Lo primordial de la conexión entre la aeronave y la tarjeta es conseguir que el dron despegue enviando el comando respectivo desde la SBC. Una vez en el aire, se procede a enviar una serie de comandos para que la aeronave se desplace por rutas predefinidas o se deje guiar mediante un teclado de forma manual por el usuario.

#### 3.2.1.1 Pruebas de control manual

El término “manual” quiere decir que se realizaron diferentes pruebas en las que no se tiene un algoritmo de control de vuelo programado en la Raspberry Pi 4 que permita a la aeronave realizar tareas de manera automática. Por esto, todo el control en estas pruebas será hecho por el usuario de forma manual empleando un teclado.

La primera prueba consiste en hacer despegar y aterrizar el dron, para esto se envían los comandos *takeoff()* y *takeoff()* respectivamente mediante código con las teclas “t” y “q”. El propósito de esto es probar la comunicación entre la Raspberry Pi 4 con el dron y su representación gráfica se la puede ver en la Figura 3.5.



**Figura 3.5** Prueba de aterrizaje y despegue

Esta prueba fue concluida con éxito durante 10 ocasiones consecutivas. Sin embargo, después de 12 minutos el proceso se interrumpió puesto que la aeronave cuenta con un sistema de seguridad que no permite el despegue si no se cuenta con suficiente batería.



La siguiente prueba planteada fue verificar el manejo del dron, esta se la realiza enviando distintos comandos de velocidad mediante un teclado USB conectado a la Raspberry Pi 4; es decir, se asignan teclas específicas para que cada una sea la encargada de enviar el comando `me.send_rc_control(self, left_right_velocity: int, forward_backward_velocity: int, up_down_velocity: int, yaw_velocity: int)` según corresponda para que provoquen movimientos en las distintas direcciones o grados de movilidad en las que el dron puede desplazarse como se puede ver en la Tabla 3.5 a una velocidad máxima de 100 cm/s.

**Tabla 3.5** Teclas asignadas para el control manual del dron

Tecla	Comando	Movimiento
←	<code>me.send_rc_control(-20, 0, 0, 0)</code>	Izquierda
→	<code>me.send_rc_control(20, 0, 0, 0)</code>	Derecha
↑	<code>me.send_rc_control(0, 0, 20, 0)</code>	Arriba
↓	<code>me.send_rc_control(0, 0, -20, 0)</code>	Abajo
W	<code>me.send_rc_control(0, 20, 0, 0)</code>	Adelante
S	<code>me.send_rc_control(0, -20, 0, 0)</code>	Atrás
A	<code>me.send_rc_control(0, 0, 0, -20)</code>	Giro Antihorario
D	<code>me.send_rc_control(0, 0, 0, 20)</code>	Giro Horario
Q	<code>me.land()</code>	Aterrizaje
Barra espaciadora	<code>me.takeoff()</code>	Despegue
T	<code>cv2.imwrite("foto.png", img)</code>	Tomar fotos

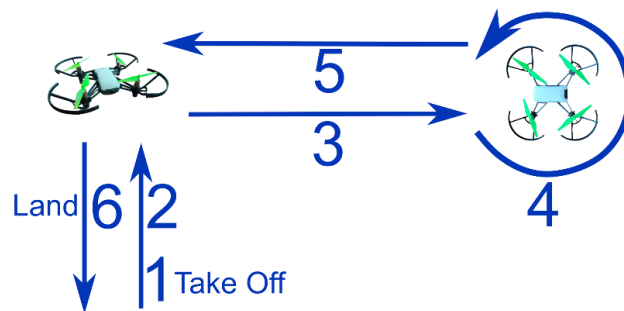
El dron se moverá durante el tiempo que se tenga presionada la tecla correspondiente a una velocidad de 20 cm/s como se muestra en la Tabla 3.5. De manera adicional al control del dron, también se utilizan los mismos procedimientos para la recepción del porcentaje de batería restante con el comando `me.get_battery()` y la captura de fotografías con el comando `cv2.imwrite("foto.png", img)` de Python para verificar la conexión con la cámara. De esta manera se pudo probar los movimientos y funciones básicas del dron necesarias para la implementación de este proyecto.

### 3.2.1.2 Pruebas de vuelo predefinidos

Para evaluar el seguimiento de una secuencia de comandos fue necesario realizar pruebas de vuelo definiendo previamente varias rutas y direcciones que debe seguir, para esto se realizó un código en la Raspberry Pi 4 que envía los comandos necesarios a la aeronave para que cumpla con un patrón de vuelo preprogramado, con el fin de determinar si la

aeronave puede ejecutar un conjunto de acciones, una después de otra sin problemas. Por ejemplo, para este tipo de pruebas se plantean dos distintos patrones de vuelo a seguir, los cuales son:

- Despegue, elevación hasta una altura superior a dos metros, desplazamiento hacia adelante durante cuatro segundos, giro de 180 grados sobre su propio eje, nuevo desplazamiento hacia adelante durante cuatro segundos y aterrizaje. Esta prueba se la puede observar de manera gráfica en la Figura 3.6. Los detalles de los vuelos se presentan en la Tabla 3.6.

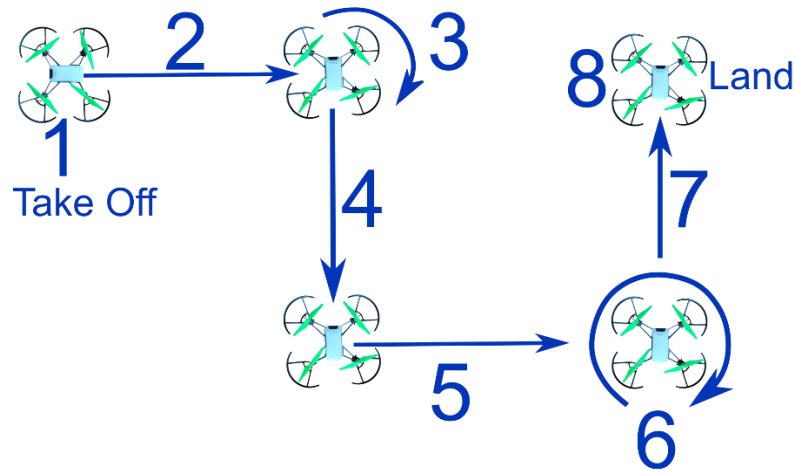


**Figura 3.6** Patrón 1 de vuelo

**Tabla 3.6** Pruebas de vuelo siguiendo el patrón 1

Prueba	Presencia de Viento	Ambiente	Llegó al lugar esperado	Distancia de aterrizaje al punto de despegue
1	No	Interior	Si	12cm
2	No	Interior	Si	9cm
3	No	Interior	Si	11cm
4	No	Interior	Si	8cm
5	No	Interior	Si	10cm
6	Si	Exterior	No	36cm
7	Si	Exterior	Si	20cm
8	No	Exterior	Si	17cm
9	Si	Exterior	Si	25cm
10	No	Exterior	Si	23cm
<b>Promedio:</b>				17.1cm
<b>Promedio %Error:</b>				17.1%

- La siguiente prueba realizada plantea un despegue y se mantiene a la altura que el dron adquiere al despegar, seguir hacia adelante 2 segundos, rotar 90 grados hacia la derecha, avanzar durante tres segundos, luego desplazarse hacia la izquierda sin enviar comandos del dron para rotación alguna por dos segundos, rotar 360 grados sobre su propio eje, dirigirse hacia atrás tres segundos y finalmente para luego aterrizar. El camino definido para el vuelo puede verse en la Figura 3.7. Los detalles de los vuelos se presentan en la Tabla 3.7.



**Figura 3.7** Prueba 2 de vuelo planificado

**Tabla 3.7** Pruebas de vuelo siguiendo el patrón 2

Prueba	Presencia de Viento	Ambiente	Llegó al lugar esperado	Distancia de aterrizaje al punto de despegue
1	No	Interior	Si	10cm
2	No	Interior	Si	13cm
3	No	Interior	Si	9cm
4	No	Interior	Si	12cm
5	Si	Exterior	Si	21cm
6	Si	Exterior	No	33cm
7	No	Exterior	Si	23cm
8	No	Exterior	Si	27cm
9	Si	Exterior	No	40cm
10	No	Exterior	Si	29cm
<b>Promedio:</b>				21.7cm
<b>Promedio %Error:</b>				21.7%

Ambas pruebas fueron realizadas en 10 ocasiones cada una teniendo resultados favorables y demostrando la capacidad del dron para seguir secuencias de comandos definidas por la base en tierra pese a que estas utilicen diferentes movimientos o se realicen en diferentes ejes. Puesto que se cumplió con el seguimiento de las instrucciones predefinidas, para el cálculo del error en el aterrizaje se consideró trazar un círculo de 2 metros de diámetro como error máximo posible, teniendo como centro al punto esperado para el aterrizaje. Se considera que la aeronave llegó al punto esperado siempre que aterrice a una distancia de 30 cm a la redonda del punto central, es decir, se acepta un error máximo del 30%.

Pese a cumplir con las acciones enviadas, el mini dron enfrentó problemas en ciertas ocasiones donde el viento era muy fuerte y la deficiencia de un sistema de localización o posicionamiento causaba que en la primera prueba no regresara exactamente al mismo punto de partida teniendo un error promedio del 17.1%. De igual manera en la segunda prueba el viento fuerte causaba que la posición final de la aeronave no sea la misma y se tengan ciertas variaciones, llegando a ser inclusive de hasta 40 centímetros de diferencia de la posición esperada. El error promedio en esta prueba fue del 21.7%.

### 3.2.2 TRANSMISIÓN DE VIDEO

Para realizar pruebas de transmisión de video se decide solamente realizar el control del dron mediante teclado y en lugar de realizar la captura de imágenes para tomar fotos, abrir un canal de comunicación y enviar la secuencia de imágenes que se recibe (video) para ser presentado en la pantalla de la Raspberry Pi 4.

La prueba consiste en encender la transmisión de video del dron desde el despegue, hacer que la aeronave suba a una distancia superior a los dos metros y desplazarse alejándose de la Raspberry Pi 4, observando el estado de la transmisión para determinar si existe una pérdida de imágenes visible, puesto que no es posible determinar los frames por segundo más que a simple vista. En la Figura 3.8 es posible apreciar el mensaje de confirmación en el terminal informando que la conexión de video se estableció con éxito entre el dron y la Raspberry Pi 4.

```
[INFO] tello.py - 422 - Send command: 'streamon'  
[INFO] tello.py - 446 - Response streamon: 'ok'  
Conexion de video establecida
```

**Figura 3.8** Conexión de video establecida

Entonces se decidió hacer una apreciación meramente cualitativa esperando tener una imagen que pueda llamarse video sin tener saltos o retaros en la transmisión ya que esto

sería un inconveniente porque el algoritmo a implementar depende completamente de la imagen obtenida desde la cámara. La distancia que se ha alcanzado obteniendo una imagen útil para trabajar en el procesamiento del video con el algoritmo híbrido corresponde a una altura superior a los 2 metros y una distancia de hasta 11 metros, lo que sería aproximadamente 12 metros del dron a la estación en línea recta. Si se desea hacer que el dron se eleve más es necesario no sobrepasar esta distancia de 12 metros entre el dron y la estación, puesto que, se presentan interferencias, pérdidas de imagen y saltos que no permiten al algoritmo de visión trabajar correctamente.



**Figura 3.9** Fotos con el dron a 3 metros, 6 metros y 11 metros de altura

En la Figura 3.9 se puede apreciar el dron a una distancia de 3 metros de la estación, a 6 metros y a 11 metros, donde se aprecia la disminución en la resolución que se tiene de las imágenes.

### **3.2.3 PRUEBAS ALGORITMO HÍBRIDO IMPLEMENTADO**

En el algoritmo híbrido implementado se decide realizar distintas pruebas que permiten observar las ventajas que brinda con respecto a la utilización individual de un algoritmo de seguimiento o una red neuronal para la identificación de objetos. Con estas pruebas se buscan analizar parámetros importantes para el correcto funcionamiento del sistema. El primero es la temperatura, elemento crucial al trabajar con SBCs, puesto que un incremento en la misma causa ralentización en el procesamiento y a largo plazo puede provocar daños físicos en el sistema. Las siguientes pruebas permiten evaluar el uso del CPU y la cantidad de memoria RAM de la Raspberry Pi 4 utilizados al ejecutar el algoritmo propuesto. Finalmente se decide realizar una comparación entre la precisión que tiene el algoritmo propuesto para identificar y seguir a un objeto contra los algoritmos mencionados anteriormente, pero empleados de manera individual.

#### **3.2.3.1 Pruebas de temperatura**

Para realizar las pruebas de temperatura se evalúan los siguientes tres algoritmos: el híbrido propuesto, el KCF de seguimiento y la red para identificación de objetos SSD MobileNet V2. Estos algoritmos serán ejecutados durante 10 minutos sin interrupción y se

tomarán los datos correspondientes a la temperatura en grados centígrados los cuales vendrán dados por el sistema y sensores internos de la Raspberry Pi 4.

Para ejecutar los diferentes códigos la imagen viene dada por una cámara USB conectada a la Raspberry Pi 4, el video corre de manera continua y el tiempo de inicio se marca desde el inicio de la ejecución del código. Se toman los datos en el arranque y luego a los 2, 5 y 10 minutos. Los resultados obtenidos se los puede observar en la Tabla 3.8.

**Tabla 3.8** Prueba de temperatura de los algoritmos en la Raspberry Pi 4

<b>Tiempo</b>	<b>SSD MobileNet V2</b>	<b>KCF</b>	<b>SSD MobileNet V2 + KCF</b>
0 minutos	36 °C	36 °C	38 °C
2 minutos	51 °C	42 °C	43 °C
5 minutos	57 °C	46 °C	46 °C
10 minutos	62 °C	51 °C	51 °C

Es necesario mencionar que pese a que se deje a los algoritmos ejecutándose durante más tiempo los valores de temperatura no fueron mayores a los medidos a los 10 minutos, por lo que la prueba se detuvo en ese punto.

Al analizar los valores obtenidos en la Tabla 3.8 se puede observar que los mayores valores de temperatura son obtenidos al trabajar con la red SSD MobileNet V2 por sí sola, inclusive en intervalos pequeños de tiempo como son a los 2 minutos de ejecución. Mientras que, los otros dos algoritmos presentan valores similares en temperatura en cualquier intervalo. El algoritmo de seguimiento KCF es el que al ejecutarse produce la menor cantidad de calor en el CPU de la Raspberry Pi 4.

La opción híbrida, al ser la que aprovecha y tiene las ventajas de los dos algoritmos (SSD MobileNet V2 + KCF), genera una menor temperatura que la red neuronal sola. Esto se debe a que como el algoritmo que se ejecuta la mayor cantidad del tiempo corresponde a la parte del seguimiento, es lógico que sus datos de temperatura en el tiempo se asemejen más a los obtenidos al correr el KCF de manera individual. La diferencia de temperatura con el algoritmo de seguimiento radica en el arranque puesto que la red es la que inicia el algoritmo híbrido por lo que la temperatura tiende a ser ligeramente mayor en ese momento, comparado con cualquiera de los dos algoritmos por separado.

### **3.2.3.2 Pruebas de carga en RAM y CPU**

Se analizó de manera porcentual la carga computacional que tiene el CPU y del mismo modo la cantidad de memoria RAM que se encuentra en uso al trabajar con cada uno de

los algoritmos. De igual manera que en las pruebas de temperatura, se utilizó la información que dan los sensores internos de la Raspberry Pi 4, como indicadores para esta prueba y una cámara USB para la adquisición del video. El único programa que se estará ejecutando es el compilador de Python Thonny, junto con la interfaz del sistema operativo de la computadora de placa reducida.

**Tabla 3.9** Pruebas de CPU y RAM de los distintos algoritmos

Tiempo Segundos	SSD MobileNet V2		KCF		SSD MobileNet V2 + KCF	
	CPU	RAM	CPU	RAM	CPU	RAM
0 segundos	3%	2.16%	3%	2.16%	3%	2.16%
10 segundos	97%	3.11%	42%	3.19%	91%	3.11%
30 segundos	90%	3.11%	42%	3.19%	42%	3.19%

En esta prueba se deciden tomar los valores en tres puntos de tiempo: el inicial antes de ejecutar el código, después de 10 segundos de ejecución y finalmente a los 30 segundos. Cuando se superó este tiempo de ejecución continua, no se observó cambios en la cantidad porcentual de RAM y CPU que utilizan los tres algoritmos, por lo que no se consideró necesario incluirlos. Los resultados se resumen en la Tabla 3.9.

De esta prueba es posible afirmar que la cantidad de RAM disponible en la Raspberry Pi 4 es más que suficiente para ejecutar cualquiera de estos tres algoritmos. El problema en sí radica en la capacidad que tiene su CPU para ejecutarlos, teniendo por ejemplo que el de seguimiento resulta ser muy ligero computacionalmente, pese a estar trabajando con video en vivo, se usa solo el 42% de su capacidad total. Por otro lado, el utilizar la red de detección de objetos por un tiempo indeterminado genera una gran carga computacional llegando a tener un pico de 97% de carga al CPU para luego estabilizarse en 90% siendo un algoritmo que genera gran estrés al CPU, esto pese a que la cantidad de RAM utilizada es similar en comparación al algoritmo de seguimiento.

Finalmente se analizaron los resultados obtenidos por el algoritmo híbrido propuesto, donde se observa que la cantidad de memoria RAM para su ejecución concuerda con los algoritmos que lo conforman. Además, la ventaja principal es que a pesar de que el pico de carga computacional en el CPU llegue al 91%, debido principalmente a su componente SSD MobileNet V2, logra reducirlo hasta llegar a 42% y mantenerse estable en este valor, esto gracias a que el otro componente que lo conforma, el algoritmo de seguimiento KCF.

### 3.2.3.3 Pruebas de precisión

Para las pruebas de precisión se utilizaron 20 videos grabados con el dron y algunos de ellos están disponibles en Youtube [48].



**Figura 3.10** Capturas de las pruebas realizadas al algoritmo de visión

En estos videos se observa a una persona con chaleco naranja aparecer y desaparecer en la imagen y en algunas escenas se tiene a más de una persona caminando o cruzándose para observar cual es el comportamiento del algoritmo híbrido ante estos casos. Algunas capturas de esta prueba se pueden ver en la Figura 3.10.

El algoritmo mostró la capacidad de identificar solo a la persona que porta el chaleco naranja y luego rastrearla incluso cuando ocurre una oclusión o la persona sale y vuelve a entrar en la escena. Finalmente, los valores presentados en la Tabla 3.10 son la relación entre el tiempo que se rastrea a la persona y el tiempo que se muestra en la escena.

**Tabla 3.10** Porcentaje medio de precisión del algoritmo propuesto

Video	Porcentaje de Identificación	Video	Porcentaje de Identificación
1	66.18 %	11	62.20 %
2	100.0 %	12	98.92 %
3	75.17 %	13	70.83 %
4	99.89 %	14	75.44 %
5	83.83 %	15	98.88 %
6	100.0 %	16	94.31 %
7	43.77 %	17	51.92 %
8	64.43 %	18	96.00 %
9	51.94 %	19	75.38 %
10	100.0 %	20	53.09 %
<b>Promedio = 78.11 %</b>			



Como se puede observar, el promedio de precisión del algoritmo híbrido de visión implementado es de un 78.11%, siendo este un buen resultado para un prototipo que se ejecuta en una Raspberry Pi 4 y trabaja con una cámara de 5 megapíxeles que pertenece al dron DJI Ryze Tello, para la adquisición de la imagen.

### **3.2.4 ALGORITMO DE CONTROL**

Las pruebas correspondientes al algoritmo de control fueron realizadas siguiendo a una persona con chaleco empleando el dron y donde la Raspberry Pi 4 se encuentra ejecutando el código propuesto de manera completa, incluyendo la interfaz de usuario.

#### **3.2.4.1 Calibración de controladores**

Se realizan vuelos de prueba para conseguir la sintonización de los algoritmos PID ya sean para el eje X o el eje Y correspondientes de la cámara. La estrategia a seguir para la sintonización de este par de controladores es sintonizar primero el del eje X que corresponde a la rotación del dron sobre su propio eje y luego en eje Y que corresponde al avance y retroceso del dron. Se inicia con el eje X, porque de volverse inestable el control no se compromete la integridad de la persona de interés o del dron, ya que en el peor de los casos la aeronave giraría en círculos.

La sintonización de los controladores corresponde a la variación de las constantes proporcional, integral y derivativa para que sea capaz de tomar las acciones de control pertinentes y hacer que el centro del bounding box de la persona de interés se encuentre sobre el punto de referencia, véase Sección 2.2.1.

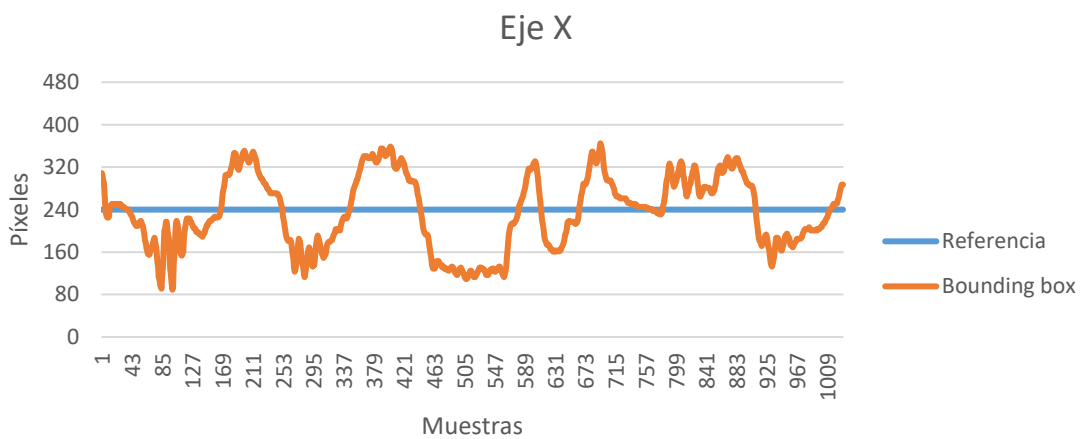
El algoritmo de control envía los parámetros de velocidad de cada eje al dron mediante el comando `me.send_rc_control(self, left_right_velocity: int, forward_backward_velocity: int, up_down_velocity: int, yaw_velocity: int)` explicado en la Sección 3.2.1.1 en base al error de posición del centro del bounding box.

Para la constante proporcional se estima la velocidad de giro del dron a la que este responde para intentar seguir a la persona, es decir, la rapidez con la que la aeronave hace el giro para alcanzar la referencia. Sin embargo, hay que tomar en cuenta que esta constante, pese a que permita alcanzar la referencia, causa oscilaciones al utilizarlo como única variable del controlador. La constante correspondiente a la parte derivativa mejora la estabilidad del sistema, aumenta el tiempo de establecimiento, pero no corrige el error en estado estable; sin embargo, es necesario tener cierta precaución porque también tiene el efecto de ampliar las señales de ruido. Finalmente, la constante de la parte integral causa que el sistema responda un poco más rápido a los cambios de referencia, reduce el error

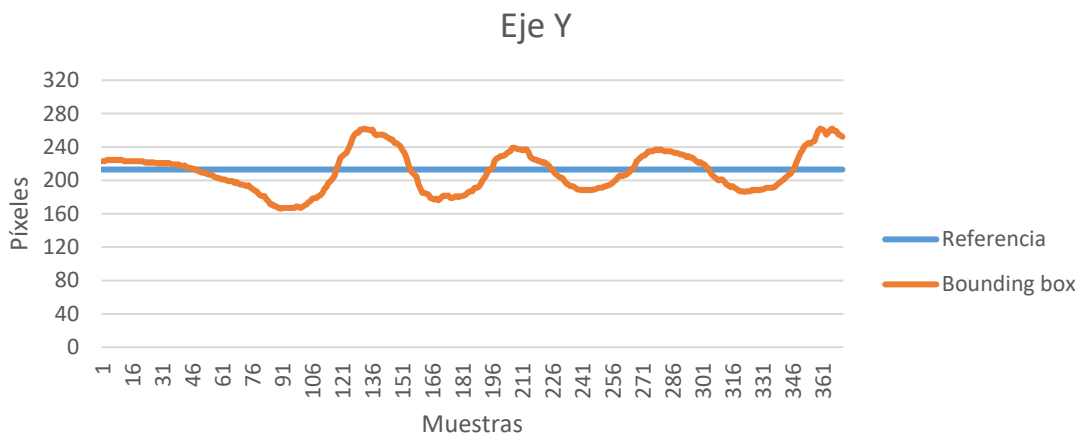
en estado estable, pero puede causar la inestabilidad del sistema si se usan valores incorrectos.

Para la sintonización de los controladores, se ve necesario trabajar con los ejes de manera separada. Siendo necesario en primer lugar sintonizar los valores correspondientes al controlador del eje X y luego hacer el mismo procedimiento para los del eje Y. Para una mejor organización se presentarán las imágenes correspondientes a ambos ejes cuando se cambien los valores de cada constante.

Las pruebas para la sintonización consistieron primero en variar los valores correspondientes a la parte proporcional para hacer que el dron logre acercarse a la referencia en el eje X sin causar un gran sobrepico o demasiadas oscilaciones. Este cambio puede ser evidenciado en las Figura 3.11 y Figura 3.12, donde se altera el valor de la constante proporcional del controlador tanto en el eje X como en el eje Y.

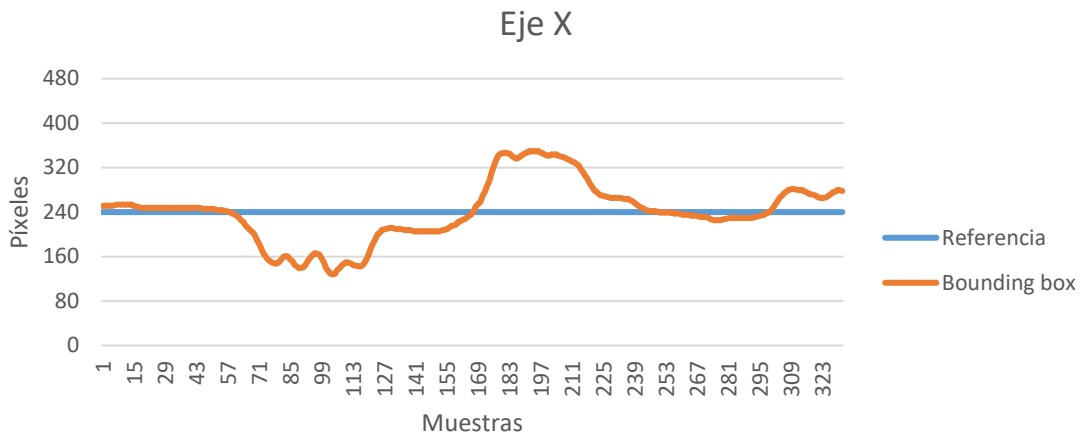


**Figura 3.11** Calibración del  $K_p$  en el eje X, valor  $K_p = 0.32$

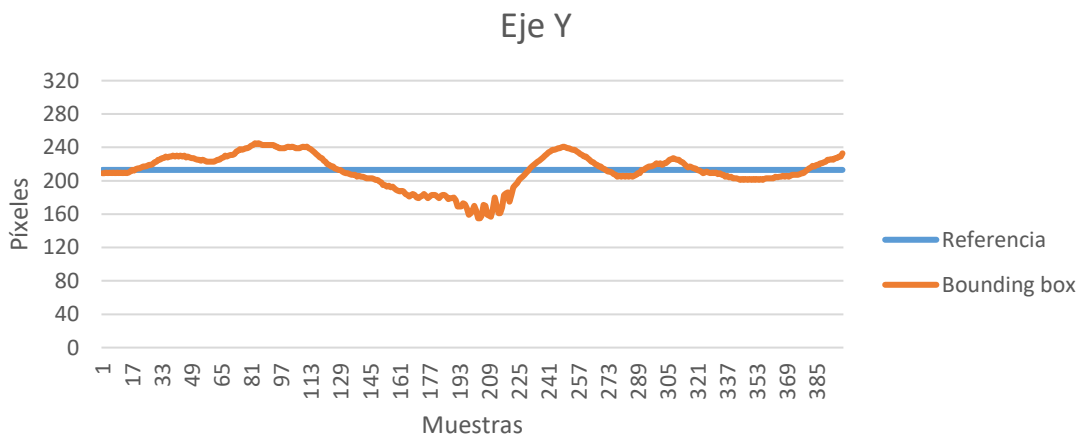


**Figura 3.12** Calibración del  $K_p$  en el eje Y, valor  $K_p = 1.02$

Se observó que luego de modificar esta primera parte, el dron presentaba oscilaciones al intentar seguir a la persona por lo que se procedió a modificar la parte derivativa para reducirlas a coste de aumentar un poco el tiempo de establecimiento. Este cambio en los dos ejes se lo puede apreciar en las Figura 3.13 y Figura 3.14.

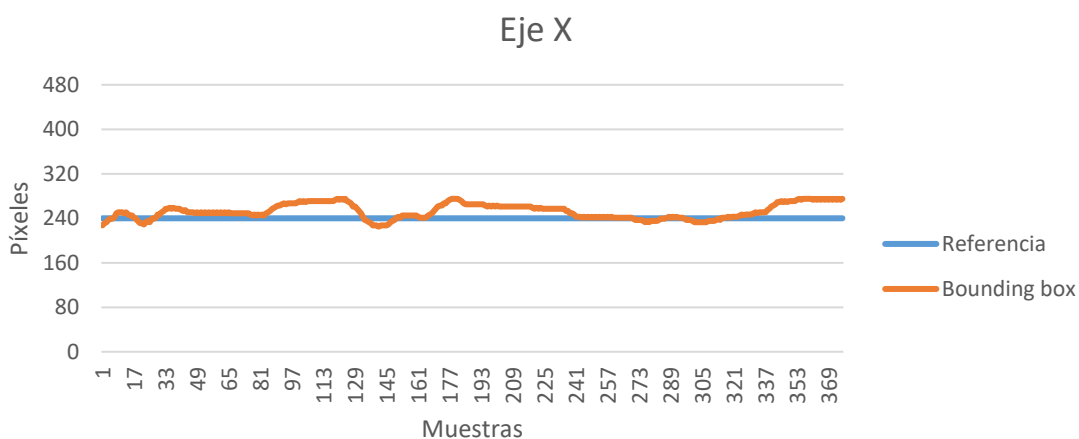


**Figura 3.13** Calibración del Kd en el eje X , valor  $K_p = 0.30$  y  $K_d = 0.1$

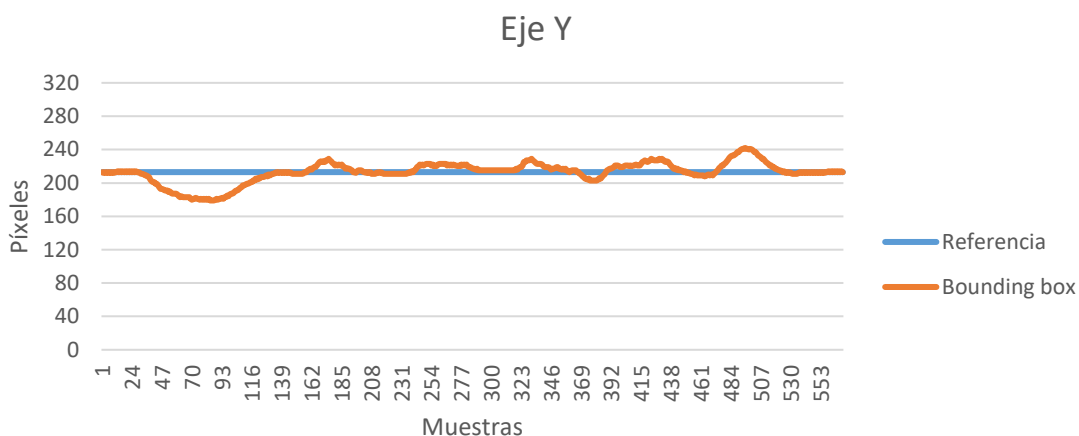


**Figura 3.14** Calibración del Kd en el eje Y, valor  $K_p = 0.95$  y  $K_d = 0.12$

Teniendo cada vez un mejor resultado se observó que el dron no llegaba a la referencia, entonces se decidió modificar la parte integral para conseguir que el dron llegue a la referencia, véase las Figura 3.15 y Figura 3.16.



**Figura 3.15** Calibración del Ki en el eje X, valor  $K_p = 0.26$ ,  $K_d = 0.08$  y  $K_i = 0.005$



**Figura 3.16** Calibración del Ki en el eje y, valor  $K_p = 0.9$ ,  $K_d = 0.09$  y  $K_i = 0.015$

Se debe tener en cuenta que al modificar la parte integral el sistema puede volver a presentar pequeñas oscilaciones, por lo que se debe calibrar nuevamente la parte derivativa hasta conseguir un buen resultado.

Los resultados correspondientes al controlador final presentado junto con pruebas que permiten evaluar la precisión de estos se encuentran en la sección siguiente, donde se analiza el comportamiento del controlador en diferentes situaciones.

### 3.2.4.2 Pruebas de precisión

Se plantean tres distintos experimentos que tienen la finalidad de poner a prueba las capacidades del dron junto con las del algoritmo correspondiente para un posterior análisis de datos y explicación de los mismos:

- La primera prueba evaluará la capacidad de identificar y seguir a la persona que se encuentra portando el chaleco naranja.
- La segunda prueba permitirá observar la capacidad que tiene el dron para seguir a la persona en una trayectoria definida.
- Finalmente, la prueba tres hará énfasis en seguir a la persona en cualquier dirección que esta decida tomar.

Durante cada prueba se consideró necesario realizar al menos 5 vuelos para una mejor adquisición de datos y se presenta el mejor de los resultados obtenidos para el análisis junto con una tabla resumen de los otros ensayos. Si se desea ver el resto de los gráficos obtenidos durante esta prueba se los puede encontrar en el ANEXO B.

Los valores finales que se utilizaron en el controlador PID en el eje X corresponden a las siguientes  $K_p = 0.24$ ,  $K_d = 0.06$  y  $K_i = 0.005$ , mientras que las constantes del eje Y poseen los siguientes valores:  $K_p = 0.85$ ,  $K_d = 0.07$  y  $K_i = 0.015$ .

#### 3.2.4.2.1 Prueba uno

La primera prueba consiste en tener a 2 personas frente a la cámara del dron a una distancia de 3 metros, una de ellas se encuentra portando el chaleco de color distintivo, véase Figura 3.17.

El dron despegará para luego identificar y diferenciar a la persona de interés, la cual se quedará estática durante 5 segundos para que el dron mueva la cámara de tal manera que llegue a la referencia.



**Figura 3.17** Posición original de la persona con chaleco

Pasado este tiempo la otra persona se irá acercando con un objeto grande que le cubra hasta llegar a ocluir a la persona del chaleco como se ve en la Figura 3.18. Estando ambas ocultas tras el objeto, se moverán en conjunto seis pasos hacia la izquierda o la derecha de la imagen capturada por el dron.



**Figura 3.18** Persona con chaleco ocluida

Posteriormente se dejará sola a la persona de interés en otra posición como se ve en la Figura 3.19. Esto para evaluar la capacidad del dron para identificarla y centrarla nuevamente en la referencia.



**Figura 3.19** Persona con chaleco cambió de posición

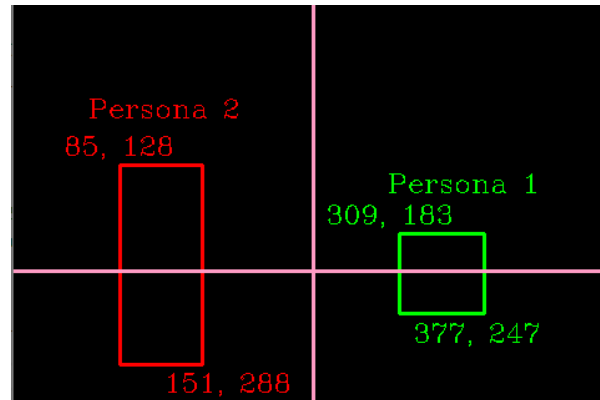
Se espera durante 5 segundos para que el dron se estabilice y se realiza el mismo procedimiento en la dirección contraria para tener un total de dos oclusiones del objeto junto con tres identificaciones.

Los resultados de los vuelos de prueba se los encuentra en la Tabla 3.11.

**Tabla 3.11** Resultados de la prueba 1

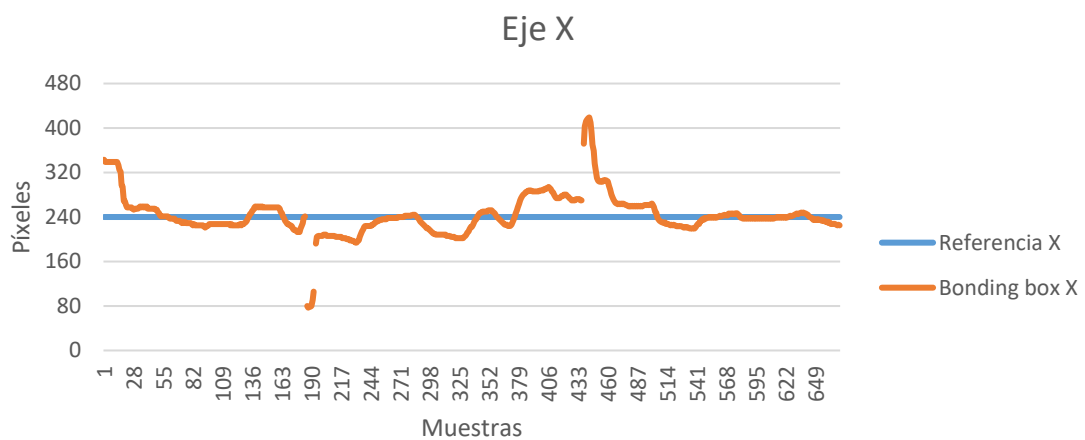
Vuelo	Siguió a la persona	Volvió a encontrar la persona	Velocidad del viento [m/s]	Cumplió con la misión
1	Si	Si	0.8	Si
2	Si	Si	0.6	Si
3	Si	Si	0.7	Si
4	Si	Si	0.6	Si
5	Si	Si	1.0	Si

Para el análisis se decidió utilizar los datos del video 2 en donde se obtienen los siguientes parámetros: persona 1 ( $x = 309, y = 183, w = 377, h = 247$ ), persona 2 ( $x = 85, y = 128, w = 151, h = 288$ ). Para un mejor entendimiento de estos se realizó la Figura 3.20.



**Figura 3.20** Ubicación inicial de ambas personas en la imagen para la prueba 1

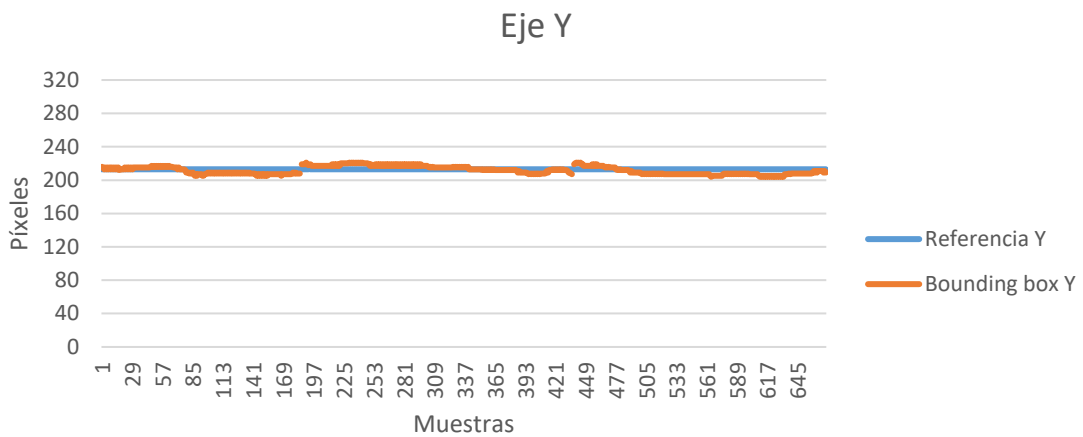
Cabe tomar en cuenta que en esta prueba se hace énfasis en la existencia de dos personas en la escena por lo que se toma también los datos correspondientes al rectángulo que encierra la ubicación ambas en la imagen



**Figura 3.21** Movimientos del dron en el eje X durante la prueba 1

La persona 1 es la que se encuentra portando el chaleco por lo que el dron tiene que realizar las acciones necesarias para hacer que el centro del bounding box verde se quede en el cruce de las líneas color rosado. Estos movimientos serán reflejados a mayor medida en la Figura 3.21, donde se observa que el valor inicial ronda los 340 píxeles haciendo que el controlador haga girar a la aeronave hasta que llegue a la referencia, luego se procede a ocultar a la persona con chaleco y llevarla al otro lado de la imagen donde el algoritmo la vuelve a encontrar y procede a centrar su bounding box de nuevo en la referencia. Esta ocultación se la realiza 2 veces en la misma prueba pudiendo evidenciarse en los picos y

saltos de la gráfica alrededor de las muestras 187 y 466 que significan que el dron hace un giro en sentido horario o antihorario hasta llegar a la referencia. Los errores y pequeñas perturbaciones que se observan entre la referencia y la persona son causadas por el viento.



**Figura 3.22** Movimientos del dron en el eje Y durante la prueba 1

En el eje Y no se tiene gran variación pese a los movimientos realizados, puesto que están basados en movimientos en el eje X. Las variaciones observadas entre la referencia y el objeto mayormente corresponden al viento que tuvo una velocidad de 0.6 m/s al momento de realizar la prueba, véase la Figura 3.22.

### 3.2.4.2.2 Prueba dos

Para la prueba dos se plantea el seguimiento de la persona con chaleco por una trayectoria definida (véase la Figura 3.23). La aeronave despegua y encuentra a la persona de interés, luego esta debe esperar 4 segundos hasta que el bounding box llegue a la referencia. Este tiempo de espera será recurrente cada vez que la persona termine un desplazamiento para que el dron tenga tiempo de llevar al bounding box hasta la referencia nuevamente.



**Figura 3.23** Trayectoria de la persona en la prueba 2



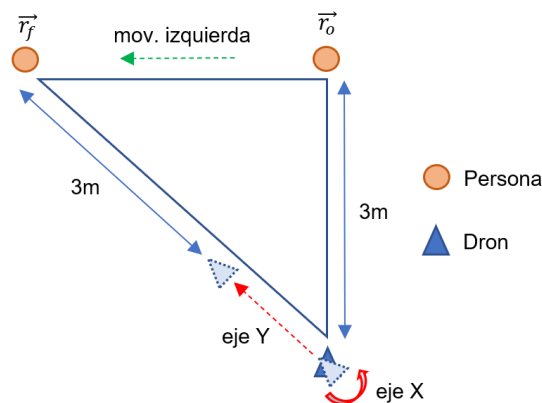
El primer movimiento a realizar es desplazarse 3 pasos hacia la dirección del dron (adelante), luego moverse hacia la izquierda 6 pasos, después moverse hacia atrás 9 pasos y finalmente hacia la derecha 12 pasos. Con esto se evaluará la capacidad de desplazamiento del dron en ambos ejes para mantener a la persona en seguimiento durante una trayectoria definida.

Del mismo modo que en la prueba uno se decide presentar la Tabla 3.12 con el resumen de los vuelos efectuados y se escoge el primero para el análisis de datos.

**Tabla 3.12** Resultados de la prueba 2

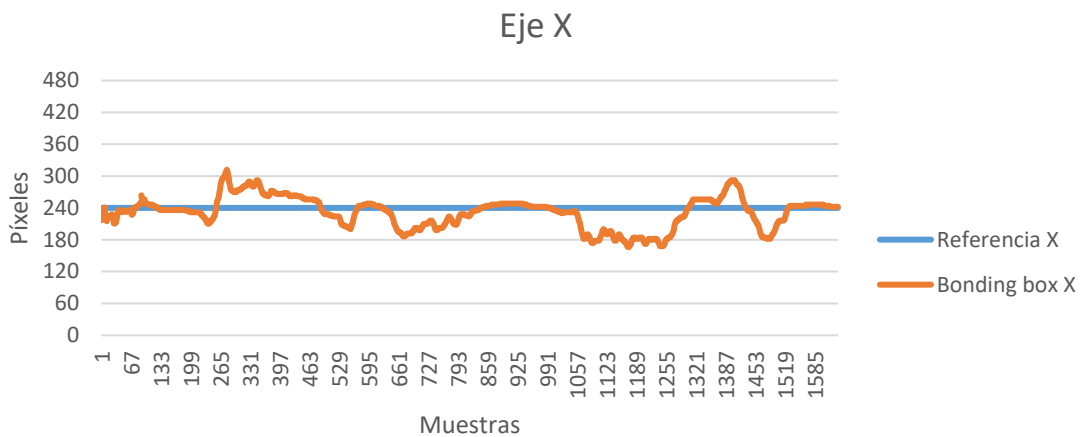
Vuelo	Siguió a la persona	Perdió a la persona	Volvió a encontrar	Velocidad del viento [m/s]	Cumplió con la misión
1	Si	No	No	0.6	Si
2	Si	No	No	0.8	Si
3	Si	Si	Si	0.6	Si
4	Si	Si	Si	0.6	Si
5	Si	No	No	0.4	Si
6	Si	Si	Si	0.4	Si

Pese a que los movimientos de la persona deberían afectar solamente al eje en el que se lo realiza, los 2 desplazamientos horizontales van a ser reflejados tanto en X como en Y. Esto debido a que, si la persona se mueve en línea recta hacia la derecha o izquierda, se está formando un triángulo entre el dron, la posición inicial de la persona ( $\vec{r}_o$ ) y su posición final ( $\vec{r}_f$ ), afectando así también al eje Y ya que la aeronave tratará de mantener siempre la misma distancia con la persona. Para una mejor explicación de esto se puede ver en la Figura 3.24.



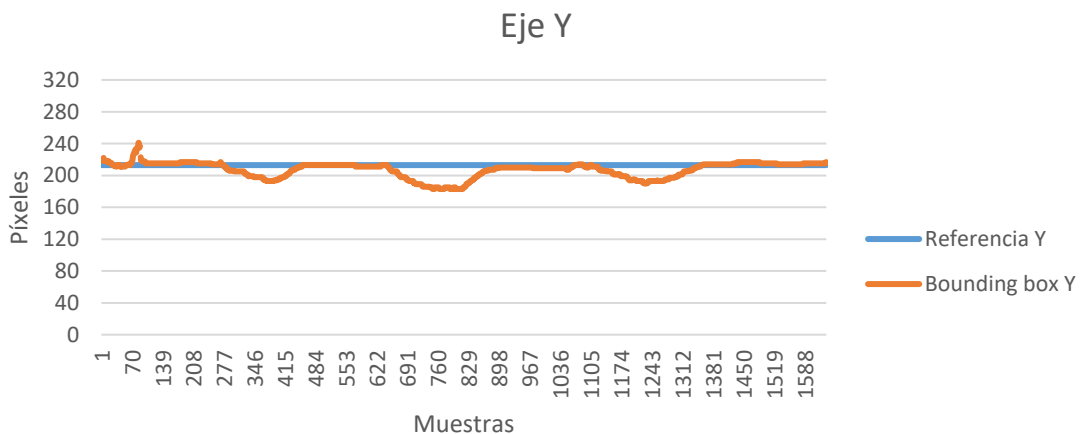
**Figura 3.24** Movimiento del dron en ambos ejes

Al analizar la Figura 3.25, que representa los desplazamientos de la persona en el eje X, se puede observar que cuando esta se mueve hacia su izquierda, los valores del bounding box son superiores a la referencia (240 píxeles), mientras que cuando se tiene valores menores a la referencia la persona se encuentra moviéndose hacia la derecha. Las variaciones que se tiene en este eje corresponden en gran manera al viento que lo desplazaba.



**Figura 3.25** Movimientos del dron en el eje X durante la prueba 2

En el eje Y es posible apreciar estos movimientos con una menor cantidad de ruido o desviaciones gracias a que el viento no soplaba demasiado en esa dirección. En la Figura 3.26 existen netamente 4 picos en las muestras 76, 376, 751 y 1276, correspondientes a cada uno de los movimientos realizados por la persona de interés, el primero corresponde al acercamiento y por eso sus valores mayores a la referencia, mientras que el movimiento de la persona alejándose del dron corresponden al tercer pico de la imagen con dirección hacia debajo de la referencia (píxel 213). Los picos 2 y 4 corresponden a los movimientos horizontales que ya se explicaron en la Figura 3.24.



**Figura 3.26** Movimientos del dron en el eje Y durante la prueba 2

### 3.2.4.2.3 Prueba tres

La prueba tres da más libertad a la persona de interés puesto que le permite desplazarse en la dirección que desee para observar los resultados que pueden ser obtenidos. Esto incluye perderse de la escena a propósito para luego volver a aparecer y que el dron logre encontrarla nuevamente. En esta prueba, es necesario acotar, que la mayoría del tiempo no se llegará a la referencia puesto que depende netamente de la velocidad y movimiento que la persona de interés desee realizar en ese momento, inclusive desaparecer detrás de objetos más grandes. Cabe mencionar que durante esta prueba la persona con chaleco no se quedó estática en ningún momento.

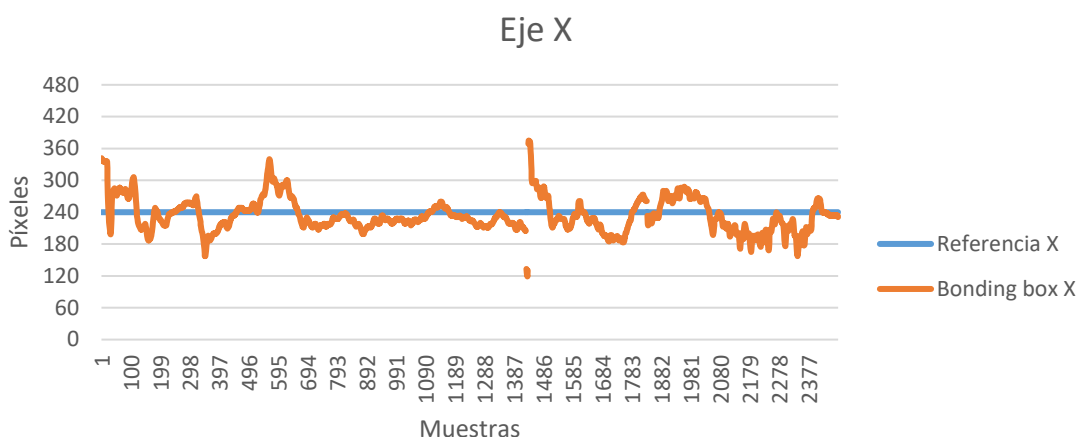
De igual manera que en las pruebas anteriores, a continuación, se presenta la Tabla 3.13 con el resumen de los vuelos en esta prueba.

**Tabla 3.13** Resultados de la prueba 3

Vuelo	Siguió a la persona	Perdió a la persona	Volvió a encontrar	Velocidad del viento [m/s]	Cumplió con la misión
1	Si	No	No	1.0	Si
2	Si	Si	Si	0.7	Si
3	Si	Si	Si	1.4	Si
4	Si	Si	Si	1.2	Si
5	Si	Si	Si	0.8	Si
6	Si	Si	Si	0.7	Si

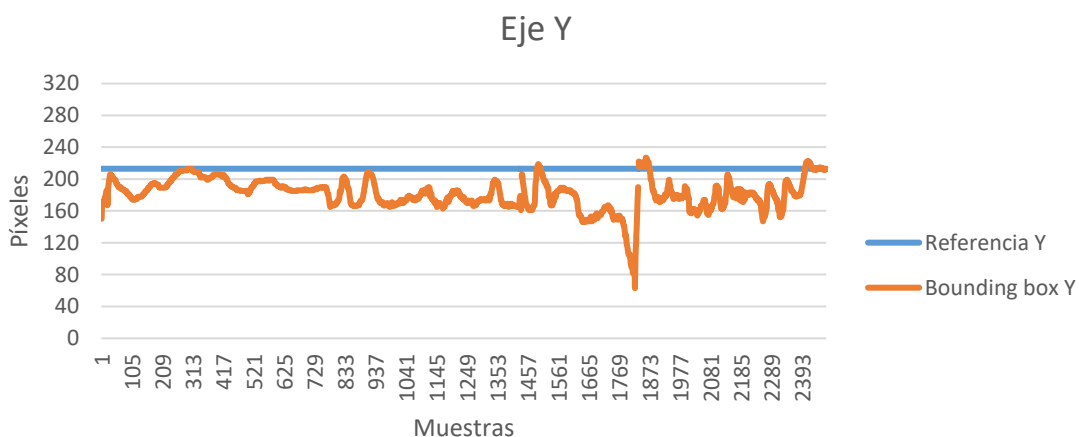
En esta ocasión para el análisis de datos se decide utilizar el último vuelo puesto que en este se siguió a la persona de interés por una distancia aproximada de 50 metros en varias direcciones, pero siempre hacia adelante obteniendo una mayor cantidad de datos que en el resto de los vuelos.

Gracias a los datos del eje X mostrados en la Figura 3.27 es posible visualizar que el dron perdió de vista a la persona de interés en ciertas ocasiones reflejadas alrededor de las muestras 1470 y 1809, lo cual no representó un problema ya que pudo encontrarla nuevamente y continuar el seguimiento. El viento en esta ocasión se convirtió en un problema mayor puesto que afectó de gran manera en el centrado que tuvo la cámara del dron para realizar el seguimiento y pese a los movimientos de la persona y la ventisca del ambiente la aeronave siempre intentaba alcanzar la referencia.



**Figura 3.27** Movimientos del dron en el eje X durante la prueba 3

Con los datos del eje Y mostrados en la Figura 3.28 es más claro apreciar el seguimiento que se realizó puesto que la mayoría de los valores son menores a la referencia (213 píxeles) queriendo decir que la persona de interés se encontraba la mayoría del tiempo alejándose del dron, a excepción de los momentos donde la misma disminuyó la velocidad o a su vez se detuvo en una posición fija, esto se puede apreciar cuando la señal naranja alcanza a la referencia.



**Figura 3.28** Movimientos del dron en el eje Y durante la prueba 3

El pico existente entre la muestra 1827 corresponde a que la persona con chaleco adelantó bastante a la aeronave haciendo que esta la pierda por un momento, pero gracias a la etapa de detección del algoritmo se la pudo volver a encontrar arrancando nuevamente la etapa de seguimiento.

### 3.3 CONDICIONES DE OPERACIÓN DEL PROTOTIPO

Existen consideraciones que son necesarias a tomar en cuenta antes de ejecutar el prototipo, las mismas vienen dadas por las pruebas realizadas y las experiencias que se han obtenido al trabajar continuamente con el prototipo, por tanto, son elementos necesarios para la utilización del sistema. Estas consideraciones se las enumera a continuación:

- El cable que sirve para la alimentación de la Raspberry Pi 4 debe ser de buena calidad para soportar al menos tres amperios. Se recomienda utilizar los cables de carga rápida ya que manejan más corriente, caso contrario se tendrá problemas en la Raspberry Pi 4 al ejecutar el código como, por ejemplo: lentitud en el procesamiento, micro cortes de energía que provocan que el algoritmo no trabaje adecuadamente, pérdida de comunicación entre el dron y la Raspberry Pi 4, etc. La alerta de energía insuficiente se presenta en la esquina superior derecha de la pantalla del dispositivo como se puede observar en la Figura 3.29



**Figura 3.29** Alerta de baja corriente

- Se debe mantener ventilado el CPU, GPU y memoria RAM de la Raspberry Pi 4 para que la temperatura no sea un problema durante la ejecución de los algoritmos. Durante todas las pruebas realizadas se mantuvo conectado un pequeño ventilador marca Canakit de 5V @ 0.12 Amperios y gracias a este la mayor temperatura alcanzada inclusive con la red fue de 62 grados centígrados como se muestra en la Tabla 3.8. En caso de no contar con este elemento de enfriamiento activo las temperaturas en la SBC se elevan a más de 75 grados causando que la Raspberry Pi 4 no funcione de manera correcta e incluso se podría provocar daños en el hardware si su uso es continuo.
- La persona a seguir debe ubicarse frente a la cámara del dron a una distancia entre 3 y 4 metros como máximo para que la etapa de detección no presente problemas

al identificarla. En caso de tener distancias menores, el bounding box que encierra a la persona de interés será demasiado grande, lo cual causaría que el dron tenga errores al realizar el seguimiento.

- Cualquier tipo de prueba deberá ser realizada en ambientes con buena iluminación debido a que el sistema óptico de posicionamiento del dron DJI Ryze Tello requiere que la superficie sobre la que vuela esté bien iluminada y no presente sombras, caso contrario la aeronave volará sin control pudiendo provocar accidentes.
- El filtro de color deberá ser configurado nuevamente dependiendo el tipo de ambiente (nublado, buena iluminación o soleado) que se presente al momento de hacer la prueba, esto comentando las líneas de código que contengan los valores límites HSV de los ambientes que no correspondan.
- Como se ha explicado anteriormente, otra condición de operación muy importante es el viento, el dron al ser tan pequeño y liviano es vulnerable a las corrientes de aire que circulen por la zona causando que no pueda mantenerse en una posición fija o no pueda moverse correctamente.
- La persona de interés debe moverse con normalidad, sin necesidad de apresurar el paso, esto debido a que los controladores están calibrados para seguir a una persona que se encuentre caminando a una velocidad normal. Si esta empieza a correr, lo más probable es que le dron la pierda de escena porque no se ha configurado para estas situaciones y se quedará flotando quieto en la última posición

## 4 CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones a las que se ha llegado en base a las pruebas y resultados obtenidos luego de haber desarrollado este trabajo de titulación.

También se mencionan ciertas recomendaciones sobre el prototipo final entregable que pueden ser tomadas en cuenta para futuras investigaciones.

Adicionalmente, cabe indicar que en base a lo desarrollado en este proyecto de titulación se ha enviado un artículo científico para su consideración dentro de las XXX Jornadas en Ingeniería Eléctrica y Electrónica, organizadas por la Escuela Politécnica Nacional.

### 4.1 CONCLUSIONES

- Se ha realizado la revisión bibliográfica de al menos dos algoritmos de seguimiento incluidos en la librería de visión artificial OpenCV, y de tres métodos de reconocimiento de objetos para posteriormente haber escogido los mejores candidatos de cada tipo que puedan ser combinados y ejecutados en una SBC con la finalidad de detectar y seguir a una persona en movimiento.
- Se demostró mediante pruebas que los mejores candidatos para esta aplicación en específico fueron el algoritmo de seguimiento KCF por su rapidez y la red de reconocimiento de objetos conocida como SSD MobileNet V2 por su precisión y velocidad de procesamiento.
- Se encontró y analizó la información correspondiente a las características técnicas de la tarjeta SBC Raspberry Pi 4 incluyendo el procesador, capacidad en memoria, tarjeta gráfica, puertos, conexiones, integración con Python, entre otras, para conocer sus ventajas y limitaciones, comprobando que es un dispositivo muy versátil y apto para trabajar con aplicaciones de visión artificial.
- Se recopiló información técnica correspondientes al dron DJI Rize Tello para constatar sus ventajas, posibles aplicaciones, conexión con Python y sus limitaciones, lo cual lo convierte en un dron de fácil acceso al público por su precio económico y cuenta con buenas prestaciones, clasificándolo como un dron multipropósito ideal para la aplicación planteada.
- Se vio la necesidad de incluir un algoritmo de identificación de color para que el algoritmo sea capaz de diferenciar entre distintos objetos pertenecientes a la misma clase presentes en una imagen.

- Se desarrolló un algoritmo híbrido de visión capaz de aprovechar las fortalezas de un algoritmo de detección y uno de tracking, enfocado a identificar y seguir a una persona que lleve puesto un chaleco que tenga un color distintivo.
- El algoritmo híbrido diseñado logró tener una precisión para el seguimiento de una persona de interés correspondiente al 78.11% en las pruebas realizadas, lo cual es considerablemente bueno puesto que no se tiene un ambiente controlado.
- Se utilizó el algoritmo híbrido como generador de acciones que fueron tomadas por el de control, el cual envió los comandos al autopiloto del Dron DJI Ryze Tello para que el dron siga a la persona en movimiento.
- El algoritmo híbrido de visión ejecutado en la Raspberry Pi 4 fue capaz de identificar y realizar el seguimiento a una persona durante las diferentes pruebas planteadas, siendo estas de diversa índole para poner a prueba las capacidades del algoritmo y cumplir su objetivo.
- Todos los algoritmos ya sean para la visión, el control o la interfaz fueron desarrollados en Python ya que es un lenguaje que cuenta con gran versatilidad en funciones y en los últimos años ha conseguido gran popularidad por su utilidad. Adicionalmente se tiene gran ventaja al utilizar este lenguaje gracias al continuo desarrollo de las librerías de visión artificial en OpenCV.
- Se determinó mediante las pruebas realizadas que para que el prototipo sea capaz de encontrar a la persona de interés ubicada frente al dron, esta debe estar a una distancia entre 2.5 a 4 metros como máximo. Esto debido a que cuando se tiene mayores distancias existe la posibilidad de que la etapa de detección no sea capaz de identificarla y, en caso de tener distancias menores, el bounding box que encierra a la persona de interés será demasiado grande, lo cual causaría que el dron tenga errores al realizar el seguimiento.
- Gracias a las pruebas de vuelo realizadas se pudo determinar que el algoritmo híbrido junto con el de control del dron puede ser ejecutado en una Raspberry Pi 4 para seguir a una persona de interés a alturas entre 2.5 metros a 3.5 metros sin presentar mayores inconvenientes. En caso de subir a una mayor altura es necesario cambiar el área en píxeles que ocupa el color distintivo.
- Se encontró necesario diseñar una interfaz de usuario que incluye al hardware y software para una mejor interacción entre el prototipo diseñado y el usuario que estará transportando el prototipo, haciendo uso de una pantalla táctil que va



conectada a la Raspberry Pi 4 y una batería instalada en la parte inferior como fuente de alimentación principal.

- En base a las pruebas y resultados obtenidos, es posible concluir que se diseñó e implementó exitosamente un sistema portátil y económico para la detección y seguimiento de una persona a través de un cuadricóptero de tamaño reducido empleando visión artificial capaz de ser ejecutado en una Raspberry Pi 4 cumpliendo de manera satisfactoria con los objetivos y alcances de este trabajo.

## 4.2 RECOMENDACIONES

- Para trabajos futuros se recomienda utilizar las nuevas versiones de Tiny Yolo y de SSD MobileNet ya que presentan mejoras significativas en la velocidad de procesamiento y en la precisión para identificar objetos, esto permitiría desarrollar prototipos más robustos y eficientes.
- Se recomienda investigar otras opciones de sistemas operativos disponibles para la Raspberry Pi 4 capaces de ejecutar el framework ROS dedicado a la robótica con el fin de migrar este trabajo a esa nueva plataforma.
- En caso de contar con un mejor sistema de enfriamiento se recomienda aumentar la velocidad de procesamiento de la SBC con el procedimiento conocido como “overclocking” y se esperaría tener un mejor funcionamiento.
- Siempre utilizar un cable para alimentación de buena calidad que soporte una corriente de 3 amperios mínimo puesto que, de no ser utilizado, la Raspberry Pi 4 presentará mal funcionamiento.
- Si el viento supera la velocidad de 1.8 m/s no se recomienda hacer volar el dron puesto que no podrá llegar a la referencia con facilidad y se le dificultará seguir a la persona de interés causando inestabilidad en el algoritmo. Por otro lado, si la velocidad del viento supera los 2.5 m/s no se debe elevar el dron por ningún motivo porque sus motores no tienen la fuerza necesaria para vencer ese viento y se perdería el control de la aeronave por completo pudiendo causar un accidente.
- Se recomienda utilizar el prototipo en las mañanas ya que el viento no tiene mucha velocidad, adicional a esto se tiene una buena iluminación la cual es necesaria por dos razones, la primera: el filtro de color es muy relativo a la iluminación del ambiente y la segunda: el dron utiliza posicionamiento óptico y si no cuenta con buena iluminación en el piso puede perder el control.

## 5 REFERENCIAS BIBLIOGRÁFICAS

- [1] M. A. Albornoz Chicango y D. P. Calahorrano Chauca, “Seguimiento de objetos basado en visión artificial para quadricóptero Parrot AR.Drone 2.0”, dic. 2016, Consultado: el 31 de agosto de 2020. [En línea]. Disponible en: <http://bibdigital.epn.edu.ec/handle/15000/16978>
- [2] V. Fonseca y L. Miguel, “Desarrollo de algoritmos para el seguimiento de trayectorias de un quadricóptero utilizando técnicas modernas de control con álgebra lineal”, jun. 2015, Consultado: el 31 de agosto de 2020. [En línea]. Disponible en: <http://bibdigital.epn.edu.ec/handle/15000/11668>
- [3] R. Rico Díaz, “Seguimiento automático de objetivos con drones mediante algoritmos de tracking”, jul. 2016, Consultado: el 1 de septiembre de 2020. [En línea]. Disponible en: <https://repositorio.uam.es/handle/10486/675129>
- [4] DJI, “DJI Active Track: Make the Drones Follow You”, *DJI Guides*, el 18 de diciembre de 2017. <https://store.dji.com/guides/film-like-a-pro-with-activetrack/> (consultado el 1 de marzo de 2021).
- [5] “Seguimiento de objetos usando OpenCV (C ++ / Python) | Aprende OpenCV”. <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/> (consultado el 20 de mayo de 2020).
- [6] “Tello”. <https://www.ryzerobotics.com/tello/specs> (consultado el 28 de julio de 2021).
- [7] J.-S. Um, *Drones as Cyber-Physical Systems: Concepts and Applications for the Fourth Industrial Revolution*. Singapore: Springer Singapore, 2019. doi: 10.1007/978-981-13-3741-3.
- [8] M. T. Calderón Jácome y D. J. Maldonado Andrade, “Control por visión de un quadricóptero utilizando ROS”, oct. 2014, Consultado: el 21 de abril de 2020. [En línea]. Disponible en: <http://bibdigital.epn.edu.ec/handle/15000/8723>
- [9] T13, *Espectacular show de drones en Tokio 2020*. Consultado: el 27 de julio de 2021. [En línea]. Disponible en: [https://www.youtube.com/watch?v=rd4-9FUEDvw&ab\\_channel=T13](https://www.youtube.com/watch?v=rd4-9FUEDvw&ab_channel=T13)
- [10] “▶ 【 Tipos de drones 】 » HispaDrones”, *HispaDrones*, el 26 de enero de 2019. <https://www.hispadrones.com/principiantes/aprendizaje-consejos/tipos-de-drones/> (consultado el 20 de julio de 2021).
- [11] “▶ Tipos de drones: ¿Cuántos tipos de drones existen en el mercado?”, *Aerial Insights*, el 27 de octubre de 2017. <https://www.aerial-insights.co/blog/tipos-de-drones/> (consultado el 21 de abril de 2020).
- [12] S. R. Muela Guaicha y K. V. Rivera Guerrero, “TELEOPERACIÓN CON REALIMENTACIÓN DE FUERZA PARA UNA FORMACIÓN DE ROBOTS MANIPULADORES AÉREOS BASADOS EN ALGORITMOS DE CONTROL DISTRIBUIDO”, p. 151.
- [13] F. J. García Muñoz, “Técnicas de control de vuelo sobre un quadricóptero”, p. 157.

- [14] S. O. Cayo Guamushig y I. D. Changoluisa Caiza, "Integración y automatización de un sistema de seguimiento de un uav para establecer un enlace de comunicación con una estación de monitoreo en tierra", sep. 2018, Consultado: el 22 de abril de 2020. [En línea]. Disponible en: <http://bibdigital.epn.edu.ec/handle/15000/19755>
- [15] "Tecnología en seguridad: Avances y nuevos retos para la industria | OMNITEMPUS", el 27 de noviembre de 2019. <https://omnitempus.com/2019/tecnologia-en-seguridad/> (consultado el 16 de mayo de 2021).
- [16] "Drones: ventajas y usos potenciales para la Policía, Seguridad Privada, Emergencias y Bomberos", *LISA Institute*. <https://www.lisainstitute.com/blogs/blog/drones-usos-policia-seguridad-emergencias-bomberos> (consultado el 9 de agosto de 2021).
- [17] "Proyecto de Inteligencia policial se apoya en drones para reducir robos en Ñaquito – Ministerio de Gobierno". <https://www.ministeriodegobierno.gob.ec/proyecto-de-inteligencia-policial-se-apoya-en-drones-para-reducir-robos-en-inaquito/> (consultado el 9 de agosto de 2021).
- [18] BBVA, "Drones para frenar la epidemia de COVID-19 | BBVA", *BBVA NOTICIAS*, el 16 de abril de 2020. <https://www.bbva.com/es/drones-para-frenar-la-epidemia-de-covid-19/> (consultado el 9 de agosto de 2021).
- [19] C. Espinosa, "Drones vigilan el distanciamiento social en las playas de Ecuador", *El Comercio*, el 30 de noviembre de 1d. C. <https://www.elcomercio.com/video/actualidad/ecuador-drones-vigilan-distanciamiento-playas.html> (consultado el 10 de agosto de 2021).
- [20] "Spark - DJI", *DJI Oficial*. <https://www.dji.com/spark> (consultado el 10 de agosto de 2021).
- [21] J. Molleda Meré, "Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3D de productos laminados", Biblioteca de la Universidad de Oviedo, Oviedo, 2009.
- [22] P. Patel y A. Thakkar, "The upsurge of deep learning for computer vision applications", *Int. J. Electr. Comput. Eng. IJECE*, vol. 10, núm. 1, p. 538, feb. 2020, doi: 10.11591/ijece.v10i1.pp538-548.
- [23] A. G. Marcos *et al.*, *Técnicas y Algoritmos Básicos de Visión Artificial*. Universidad de La Rioja, 2006. Consultado: el 18 de mayo de 2021. [En línea]. Disponible en: <https://investigacion.unirioja.es/documentos/5c13b22ac8914b6ed3778a6a>
- [24] A. Fernández Villán, *Mastering OpenCV 4 with Python A practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7*. BIRMINGHAM - MUMBAI: PACKT.
- [25] Y. Amit, P. Felzenszwalb, y R. Girshick, "Object Detection", 2020, pp. 1–9. doi: 10.1007/978-3-030-03243-2\_660-1.
- [26] J. Howse, *OpenCV Computer Vision with Python*, First Published April 2013. PACKT open source.

- [27] S. Albawi, T. A. Mohammed, y S. Al-Zawi, "Understanding of a convolutional neural network", en *2017 International Conference on Engineering and Technology (ICET)*, ago. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.
- [28] J. Redmon y A. Farhadi, "YOLOv3: An Incremental Improvement", *ArXiv180402767 Cs*, abr. 2018, Consultado: el 18 de enero de 2022. [En línea]. Disponible en: <http://arxiv.org/abs/1804.02767>
- [29] R. Huang, J. Gu, X. Sun, Y. Hou, y S. Uddin, "A Rapid Recognition Method for Electronic Components Based on the Improved YOLO-V3 Network", *Electronics*, vol. 8, núm. 8, p. 825, jul. 2019, doi: 10.3390/electronics8080825.
- [30] G. Oltean, C. Florea, R. Orghidan, y V. Oltean, "Towards Real Time Vehicle Counting using YOLO-Tiny and Fast Motion Estimation", en *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, oct. 2019, pp. 240–243. doi: 10.1109/SIITME47687.2019.8990708.
- [31] M. A. Qurishee, "Low-cost deep learning UAV and Raspberry Pi solution to real time pavement condition assessment", *Masters Theses Dr. Diss.*, may 2019, [En línea]. Disponible en: <https://scholar.utc.edu/theses/601>
- [32] Z. Kalal, K. Mikolajczyk, y J. Matas, "Tracking-Learning-Detection", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, núm. 7, pp. 1409–1422, jul. 2012, doi: 10.1109/TPAMI.2011.239.
- [33] T. Xu, C. Huang, Q. He, G. Guan, y Y. Zhang, "An improved TLD target tracking algorithm", en *2016 IEEE International Conference on Information and Automation (ICIA)*, Ningbo, China, ago. 2016, pp. 2051–2055. doi: 10.1109/ICInfA.2016.7832157.
- [34] J. Wan Park, S. Kim, Y. Lee, y I. Joe, "Improvement of the KCF Tracking Algorithm through Object Detection", *Int. J. Eng. Technol.*, vol. 7, núm. 4.4, p. 11, sep. 2018, doi: 10.14419/ijet.v7i4.4.19594.
- [35] N. Raghava, K. Gupta, I. Kedia, y A. Goyal, "An Experimental Comparison of Different Object Tracking Algorithms", en *2020 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, jul. 2020, pp. 0726–0730. doi: 10.1109/ICCSP48568.2020.9182101.
- [36] J. F. Henriques, R. Caseiro, P. Martins, y J. Batista, "High-Speed Tracking with Kernelized Correlation Filters", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, núm. 3, pp. 583–596, mar. 2015, doi: 10.1109/TPAMI.2014.2345390.
- [37] T. R. P. Foundation, "Buy a Raspberry Pi 3 Model B+", *Raspberry Pi*. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/> (consultado el 27 de julio de 2021).
- [38] "NVIDIA Jetson Nano", *NVIDIA*. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/> (consultado el 15 de junio de 2021).
- [39] T. R. P. Foundation, "Raspberry Pi 4 Model B specifications", *Raspberry Pi*. <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/> (consultado el 30 de mayo de 2021).

- [40] “OpenCV: Introduction”. <https://docs.opencv.org/4.5.2/d1/dfb/intro.html> (consultado el 16 de mayo de 2021).
- [41] D. F. Escoté, *DJITelloPy*. 2021. Consultado: el 23 de agosto de 2021. [En línea]. Disponible en: <https://github.com/damiafuentes/DJITelloPy>
- [42] M. Loesdau, S. Chabrier, y A. Gabillon, “Hue and Saturation in the RGB Color Space”, en *Image and Signal Processing*, Cham, 2014, pp. 203–212. doi: 10.1007/978-3-319-07998-1\_23.
- [43] K. B. Shaik, P. Ganesan, V. Kalist, B. S. Sathish, y J. M. M. Jenitha, “Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space”, *Procedia Comput. Sci.*, vol. 57, pp. 41–48, ene. 2015, doi: 10.1016/j.procs.2015.07.362.
- [44] T.-W. Mi y M.-T. Yang, “Comparison of Tracking Techniques on 360-Degree Videos”, *Appl. Sci.*, vol. 9, núm. 16, p. 3336, ago. 2019, doi: 10.3390/app9163336.
- [45] “tkinter — Python interface to Tcl/Tk — Python 3.9.7 documentation”. <https://docs.python.org/3/library/tkinter.html> (consultado el 30 de septiembre de 2021).
- [46] “Videos de prueba Tesis 2021-A - YouTube”. [https://www.youtube.com/playlist?list=PL7dlviBy\\_Hz1fTe3Gx\\_DNdu-4cdHeemaU](https://www.youtube.com/playlist?list=PL7dlviBy_Hz1fTe3Gx_DNdu-4cdHeemaU) (consultado el 30 de septiembre de 2021).
- [47] B. Klare y S. Sarkar, “Background subtraction in varying illuminations using an ensemble based on an enlarged feature set”, en *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Miami, FL, jun. 2009, pp. 66–73. doi: 10.1109/CVPRW.2009.5204078.
- [48] Bryan Castelo, *Algoritmo Híbrido de Visión Artificial*, (el 6 de noviembre de 2021). Consultado: el 6 de noviembre de 2021. [En línea]. Disponible en: <https://www.youtube.com/watch?v=xxGgIP3Y1J4>

## **ANEXOS**

Anexo A: Manual de usuario

Anexo B: Gráficos completos de las pruebas finales

## ANEXO A

### MANUAL DE USUARIO

En algunas aplicaciones de seguridad y monitoreo se requiere que una persona, vehículo u objeto de interés sea seguido de manera permanente durante cierto periodo de tiempo para garantizar su seguridad hasta llegar al destino. Es por esto que se ha desarrollado este prototipo capaz de seguir a una persona en ambientes exteriores durante un periodo de tiempo máximo de 10 minutos utilizando un dron económico DJI Ryze Tello en conjunto con una Raspberry Pi 4 que hará el papel del cerebro del sistema

A continuación, se detallan las instrucciones de uso para que el dispositivo funcione de forma correcta.

#### INDICACIONES DE USUARIO

- **Chaleco**

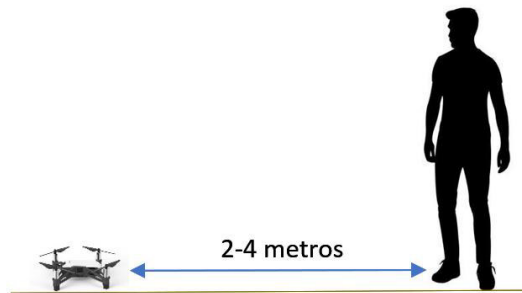
Una de las condiciones para el funcionamiento de este prototipo es que la persona a la que se desea seguir con el dron tendrá que portar un chaleco de color naranja como el de la Figura A.1 para que sirva como distintivo entre más personas que puedan aparecer en escena.



**Figura A.1** Chaleco de color naranja

- **Distancias**

Antes de iniciar el funcionamiento del programa en la Raspberry Pi 4 y levantar vuelo con la aeronave es necesario que la persona a la que se desee seguir se ubique frente a la cámara del dron a una distancia de entre 3 y 4 metros máximo como se ve en la Figura A.2. Esto es para que el filtro de color no tenga problemas con el área del chaleco.



**Figura A.2** Distancia de la persona al dron

## INDICACIONES DEL PROTOTIPO

### - Alimentación

La alimentación del dispositivo es un punto importante, se debe conectar el prototipo a la batería de 5V @ 2A instalada en la base utilizando un cable de USB tipo A a USB tipo C que soporte altas corrientes (de preferencia uno de carga rápida) esto para evitar que la pantalla y la Raspberry Pi 4 se vean afectadas por falta de energía, Véase la Figura A.3.



**Figura A.3** Alimentación del prototipo

Una vez que se haya conectado el dispositivo a la batería, este se encenderá de manera automática.

### - Dron

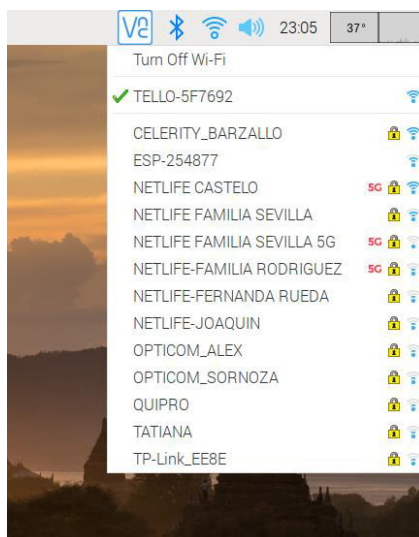
Para encender el dron se debe pulsar el único botón que se encuentra al lado izquierdo del fuselaje. A continuación, el led indicador junto a la cámara empezará a titilar de diferentes colores, hay que esperar unos segundos hasta que la luz parpadeante se vuelva de color blanco señalando que el dron está listo para realizar la conexión con la Raspberry Pi 4, véase la Figura A.4.





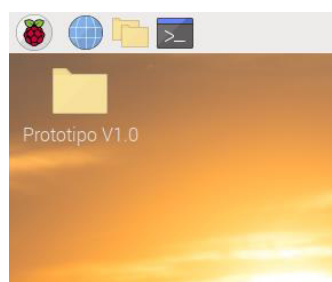
**Figura A.4** a) Botón de encendido, b) Led indicador

Una vez que se tenga encendido el dron y la Raspberry con su pantalla se procede a hacer la conexión WiFi entre ambos. Para ello en la opción de conexiones WiFi de la Raspberry buscamos la red libre llamada *TELLO-5F7692* y se procede a conectarla como se aprecia en la Figura A.5.



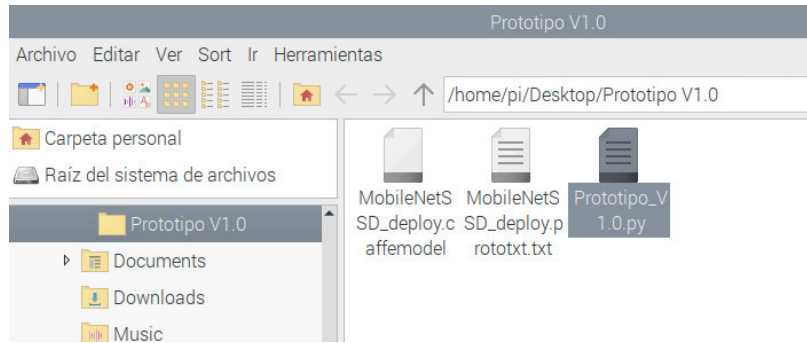
**Figura A.5** Conexión de la Raspberry Pi 4 con la red WiFi emitida por el dron

Cuando ya se tenga una conexión estable se procede a abrir la interfaz del algoritmo desarrollado, para ello debemos entrar en la carpeta *Prototipo V1.0* ubicada en el escritorio. (véase Figura A.6).



**Figura A.6** Carpeta *Prototipo V1.0*

Luego damos doble clic en el archivo *Prototipo\_V1.0.py* que se puede ver en la Figura A.7.

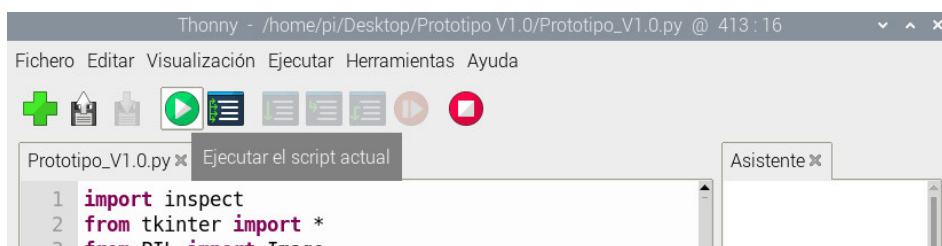


**Figura A.7** Archivo *Prototipo\_V1.0.py*

Se abrirá el compilador Tonny Python IDE con el código del archivo, aquí debemos descender en el programa hasta llegar a la función *filtrocolor*, aquí debemos comentar o descomentar (agregando o quitando #) las líneas que corresponden al tipo de iluminación presente en el ambiente como se puede ver en la Figura A.8 y luego dar clic en el botón verde llamado *Ejecutar el script actual* que se ve en la Figura A.9.

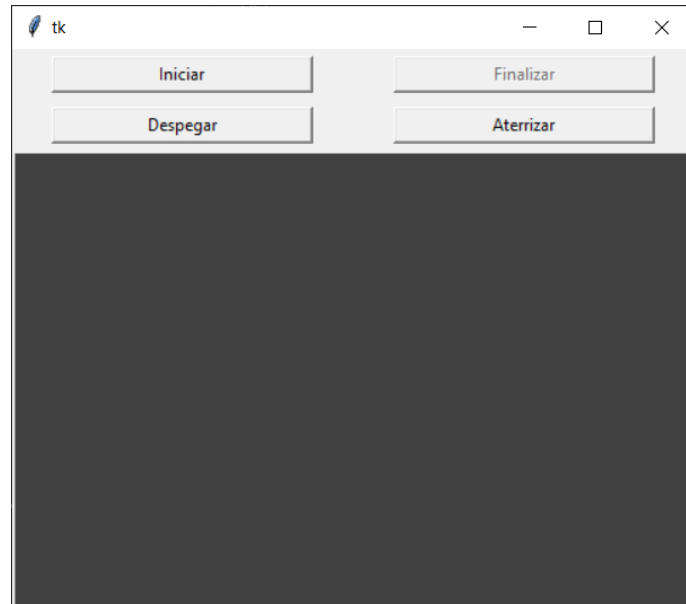


**Figura A.8** Selección del tipo de iluminación presente en el ambiente



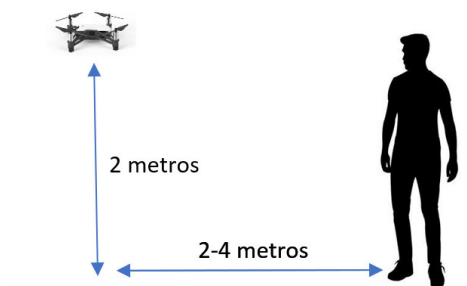
**Figura A.9** Botón *Ejecutar el script actual* del compilador Tonny Python IDE

A continuación, se abrirá la interfaz de la Figura A.10, allí se podrá encontrar cuatro botones: *DESPEGAR*, *INICIAR*, *ATERRIZAR* y *FINALIZAR* y una pantalla negra en donde posteriormente se mostrará la imagen recibida desde el dron.



**Figura A.10** Interfaz de usuario

Para iniciar el proceso, el usuario deberá hacer clic sobre el botón *DESPEGAR*, esto hará que el dron empiece a elevarse sin transmitir video. Se deberá esperar a que el dron supere la altura de 2m antes de dar clic en el botón *INICIAR* como se ve en la Figura A.11.

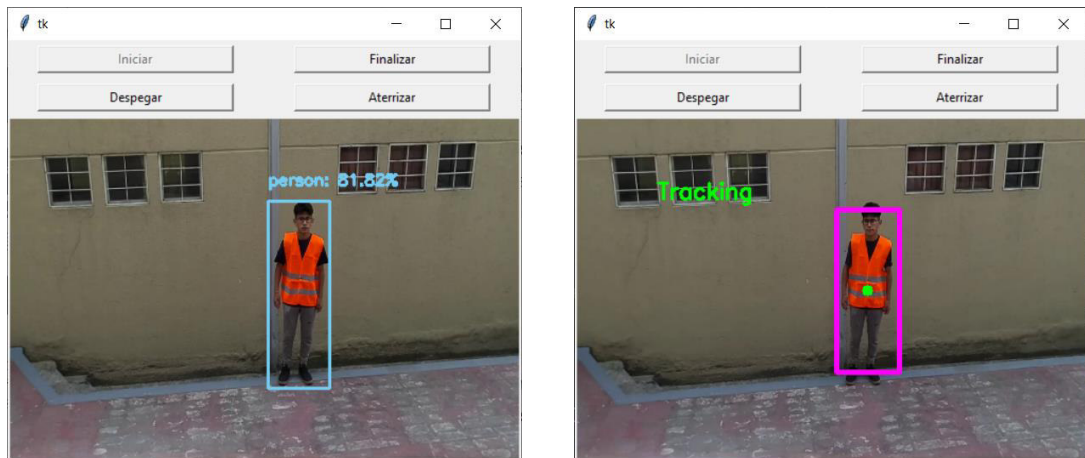


**Figura A.11** Altura de vuelo antes de dar clic al botón iniciar

Una vez se haya presionado el botón *INICIAR* el dron continuará elevándose por unos instantes más hasta llegar a una altura de entre 2.5 a 3,5 metros antes de empezar a transmitir el video.

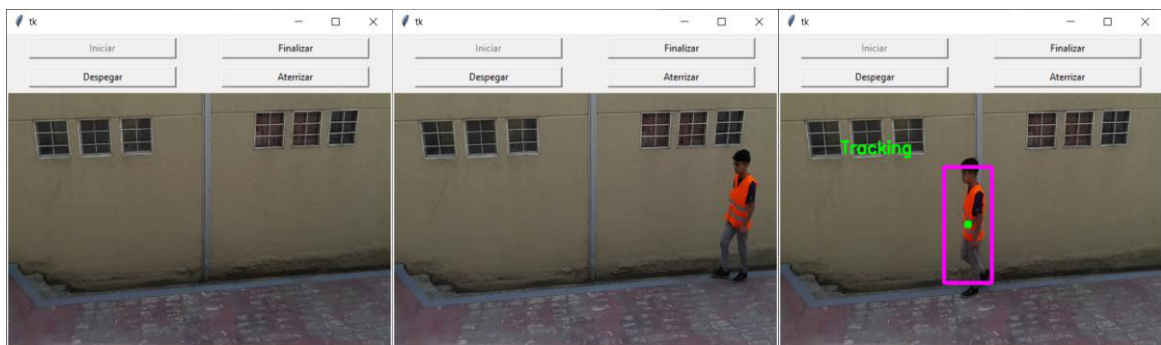
En el instante en que el dron comienza a transmitir la imagen en "tiempo real" entrará en funcionamiento la etapa de detección. El usuario deberá permanecer quieto hasta que el

algoritmo lo encuentre y encierre en un bounding box de color rosado arrancando la etapa de seguimiento, véase la Figura A.12.



**Figura A.12** Inicio de transmisión de video y arranque del algoritmo

Si la persona sale de escena el algoritmo hará que el dron se quede volando en el último lugar que la vio hasta que vuelva a aparecer en la imagen, para ello el usuario deberá caminar nuevamente frente al dron para que este lo vuelva a reconocer y arranque de nuevo el sistema. Otra situación posible es que la persona pase detrás de un objeto y luego vuelva a aparecer en la misma imagen, en este caso no hace falta caminar nuevamente frente al dron ya que el algoritmo se encargará de volverla a encontrar nuevamente sin mayor dificultad, véase la Figura A.13.



**Figura A.13** Persona sale de escena

Si se requiere que el dron aterrice se debe presionar el botón *ATERRIZAR*, este hará que el vehículo descienda y aterrice. En caso de emergencia se deberá presionar el botón *FINALIZAR*, este hará que el dron detenga su desplazamiento y aterrice de inmediato en el lugar que se encuentre.

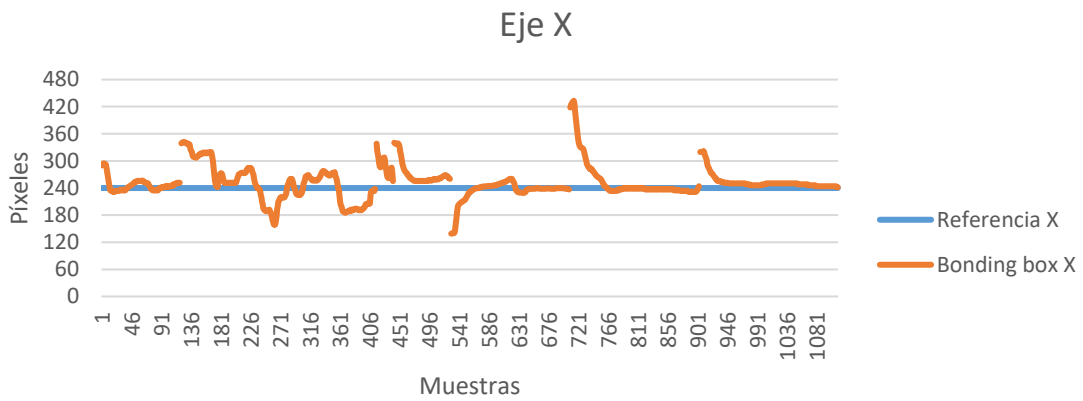
## ANEXO B

### GRÁFICOS COMPLETOS DE LAS PRUEBAS FINALES

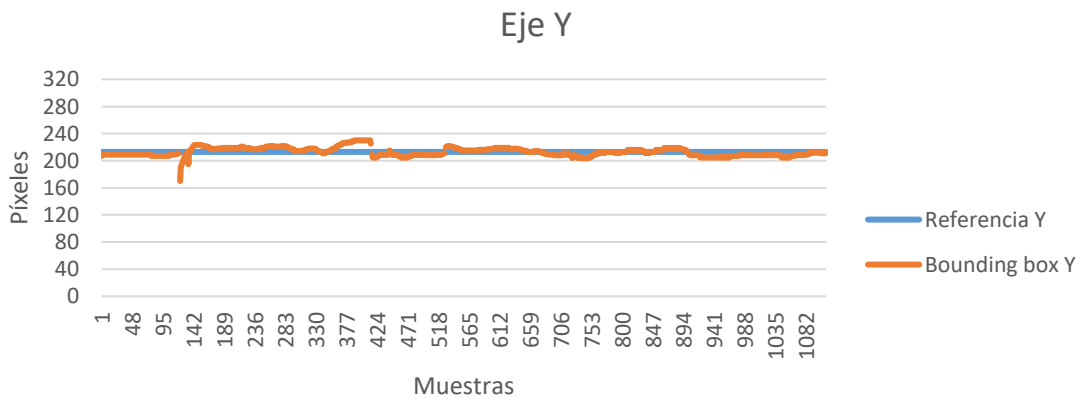
#### PRUEBA 1

La prueba consistió en tener a 2 personas frente a la cámara del dron a una distancia de 3 metros del lugar del despegue. Una sola de ellas se encontraba portando el chaleco de color distintivo, esto con el fin de evaluar la capacidad del algoritmo implementado para identificar y seguir a la persona de interés a pesar de que se presenten oclusiones durante el vuelo (véase la Sección 3.2.4.2.1 para más detalles). Para esta prueba se realizaron 5 vuelos o ensayos y se presentan las gráficas resultantes desde la Figura B.1 a la Figura B.10.

#### - Ensayo 1

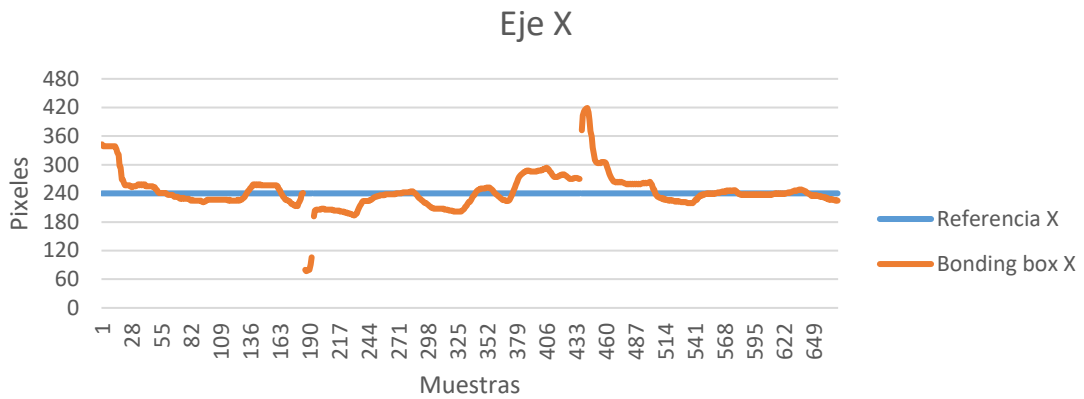


**Figura B.1** Respuesta del prototipo en el eje X durante el ensayo 1 (Prueba1)

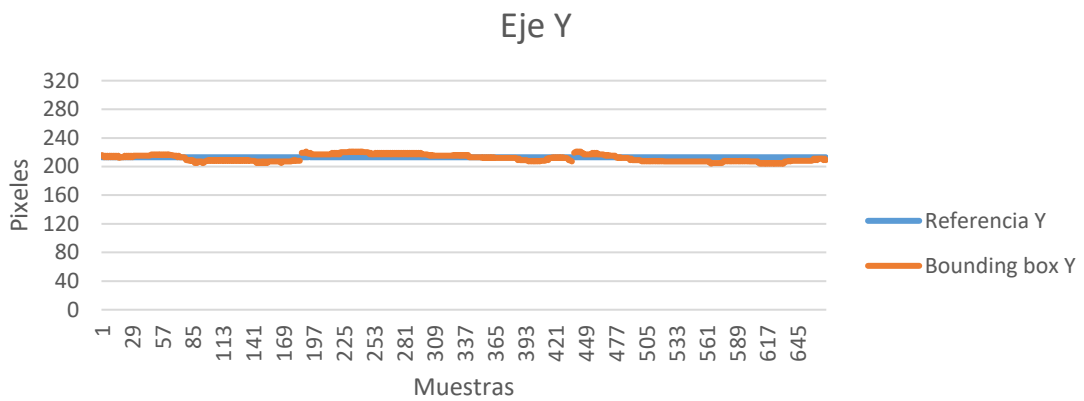


**Figura B.2** Respuesta del prototipo en el eje Y durante el ensayo 1 (Prueba 1)

- **Ensayo 2**

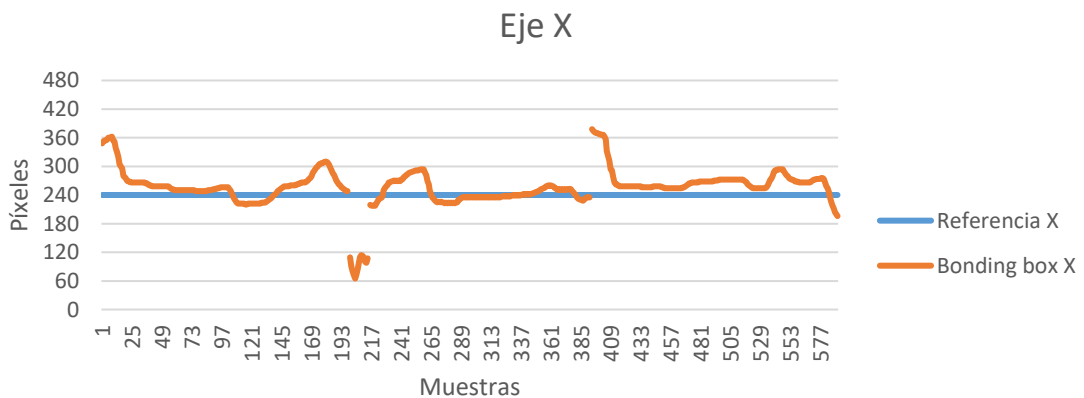


**Figura B.3** Respuesta del prototipo en el eje X durante el ensayo 2 (Prueba 1)



**Figura B.4** Respuesta del prototipo en el eje Y durante el ensayo 2 (Prueba 1)

- **Ensayo 3**



**Figura B.5** Respuesta del prototipo en el eje X durante el ensayo 3 (Prueba 1)

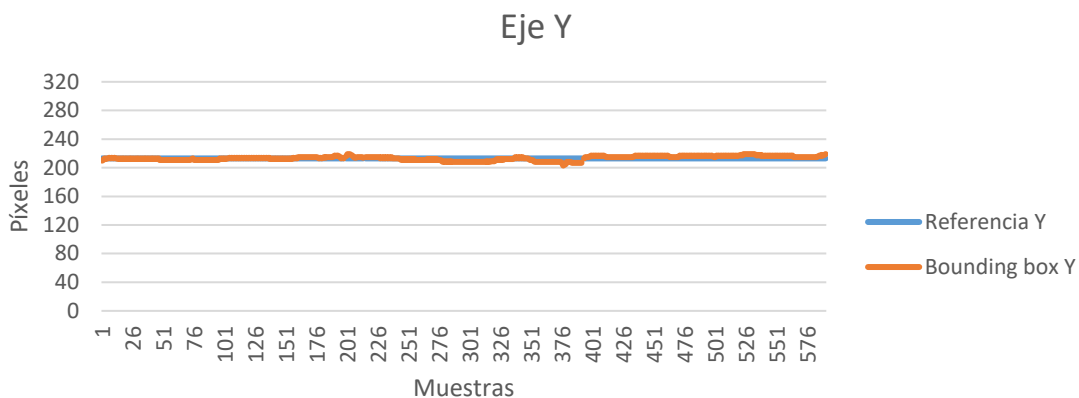


Figura B.6 Respuesta del prototipo en el eje Y durante el ensayo 3 (Prueba 1)

- **Ensayo 4**

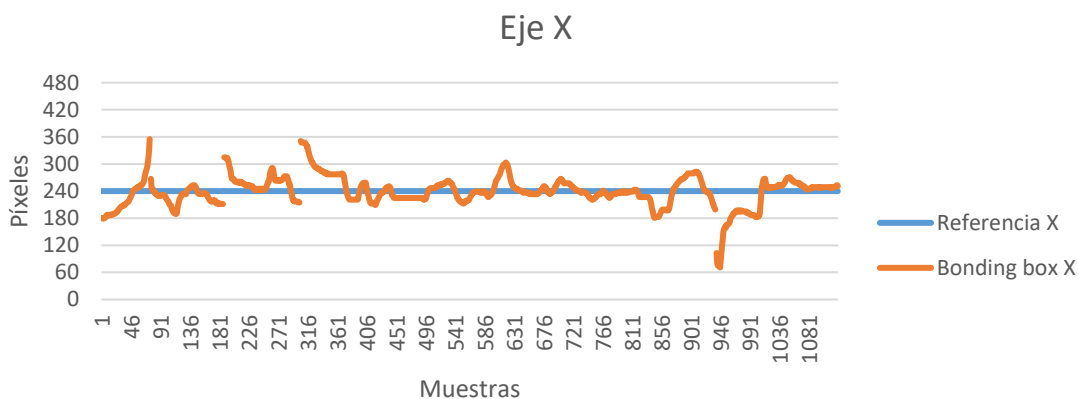


Figura B.7 Respuesta del prototipo en el eje X durante el ensayo 4 (Prueba 1)

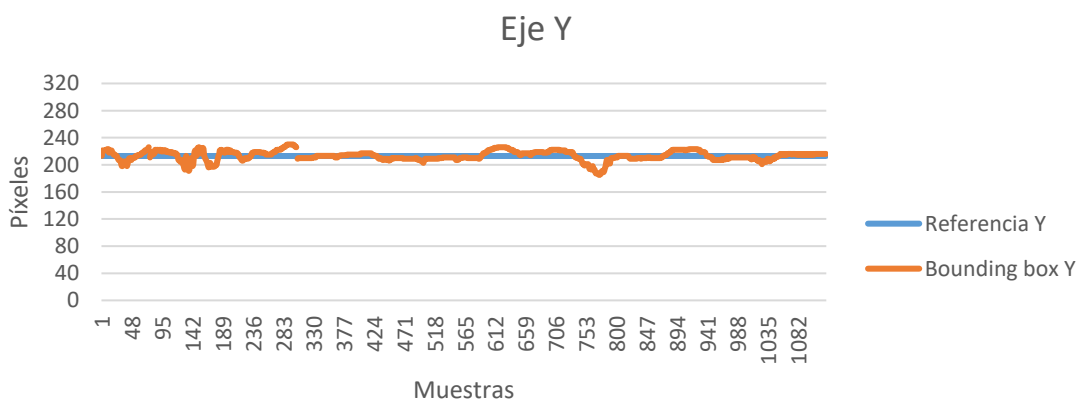
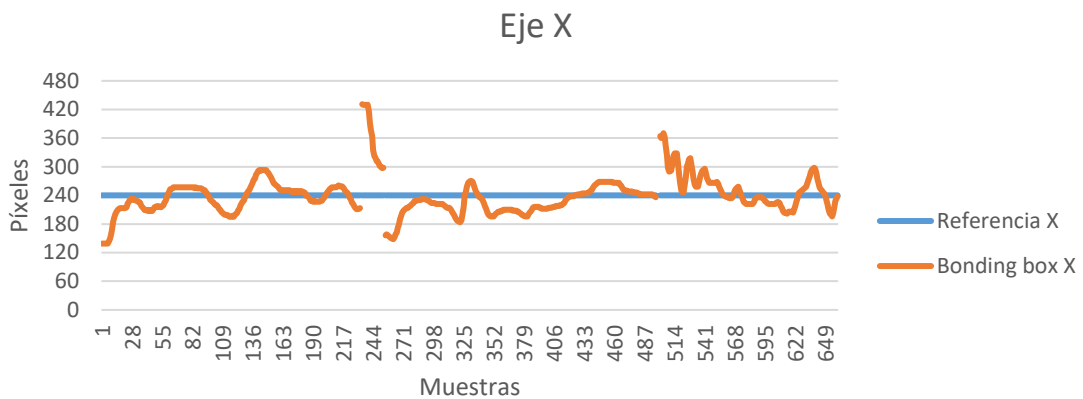
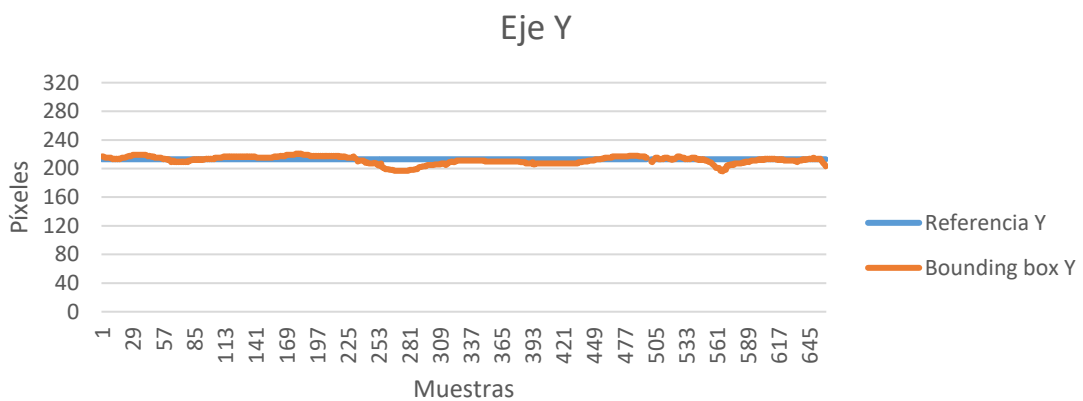


Figura B.8 Respuesta del prototipo en el eje Y durante el ensayo 4 (Prueba 1)

## - Ensayo 5



**Figura B.9** Respuesta del prototipo en el eje X durante el ensayo 5 (Prueba 1)



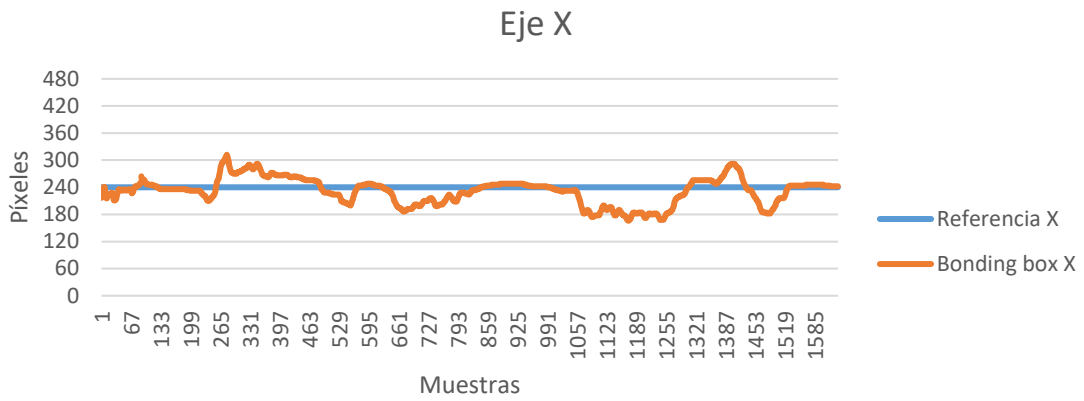
**Figura B.10** Respuesta del prototipo en el eje Y durante el ensayo 5 (Prueba 1)

## PRUEBA 2

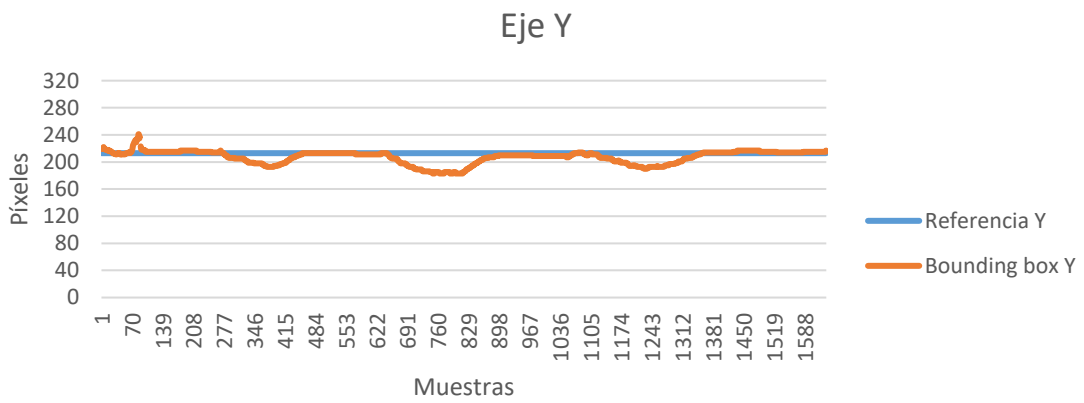
La segunda prueba permite observar la capacidad que tiene el dron para seguir a la persona con chaleco naranja por una trayectoria definida. Los movimientos realizados fueron: 3 pasos hacia adelante, 6 hacia la izquierda, 9 hacia atrás y 12 hacia la derecha (véase la Sección 3.2.4.2.2 para más detalles). Se realizaron 6 vuelos de prueba o ensayo y a continuación se presentan las gráficas resultantes desde la Figura B.11 a la Figura B.22.



- **Ensayo 1**

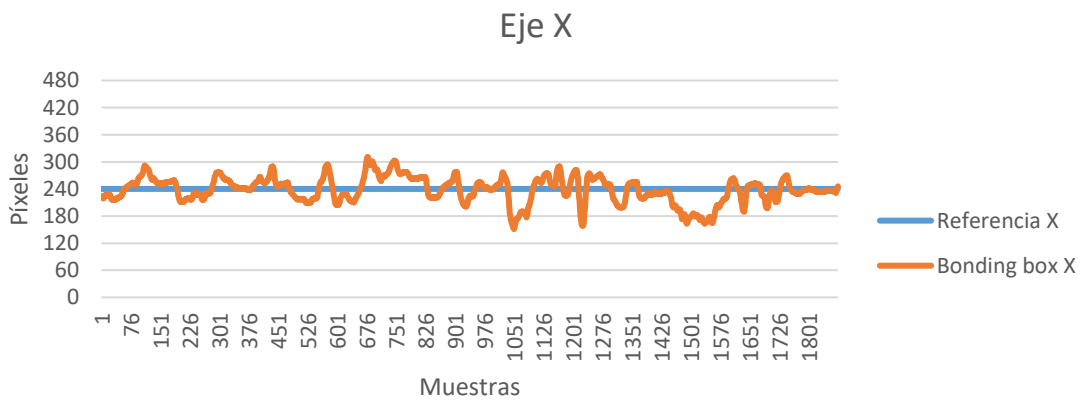


**Figura B.11** Respuesta del prototipo en el eje X durante el ensayo 1 (Prueba 2)

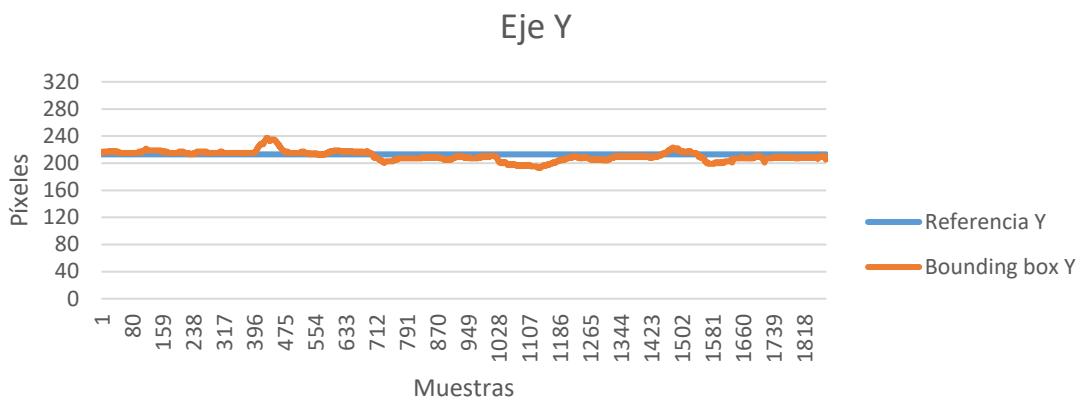


**Figura B.12** Respuesta del prototipo en el eje Y durante el ensayo 1 (Prueba 2)

- **Ensayo 2**

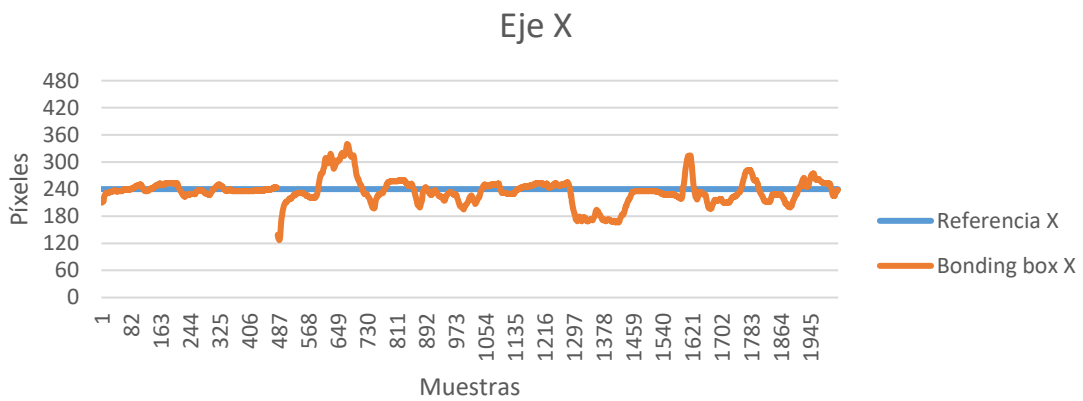


**Figura B.13** Respuesta del prototipo en el eje X durante el ensayo 2 (Prueba 2)

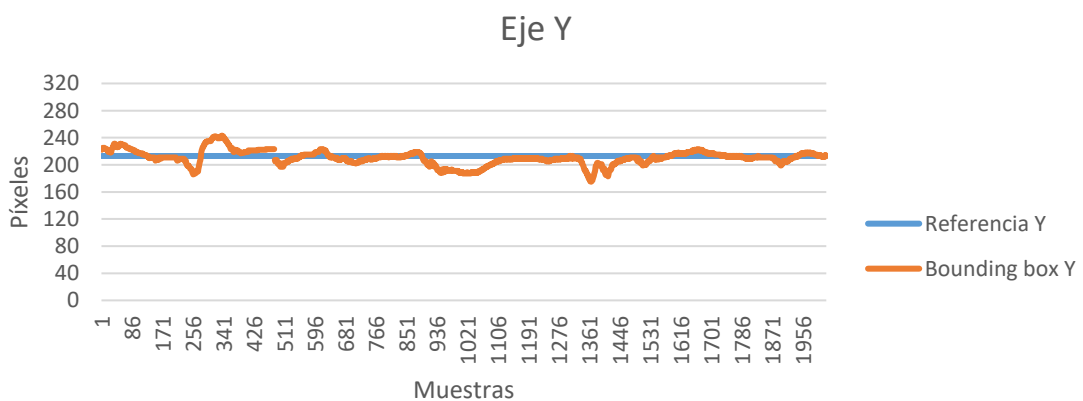


**Figura B.14** Respuesta del prototipo en el eje Y durante el ensayo 2 (Prueba 2)

**Ensayo 3**

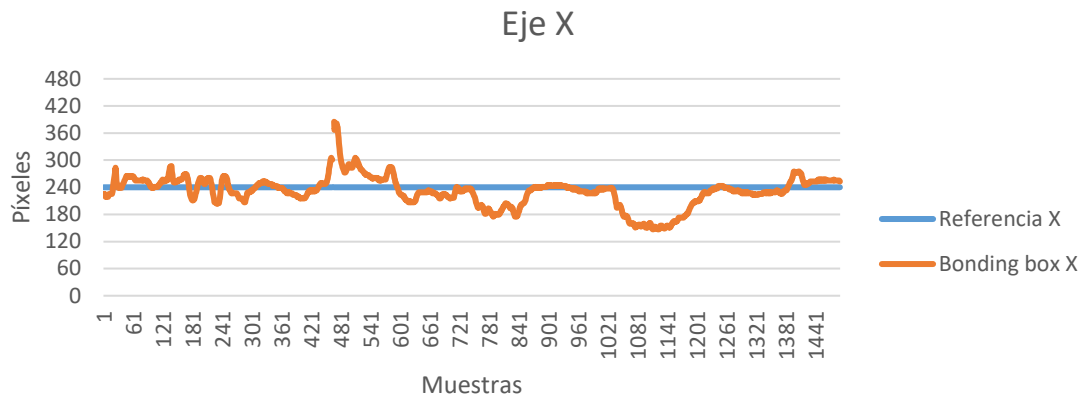


**Figura B.15** Respuesta del prototipo en el eje X durante el ensayo 3 (Prueba 2)

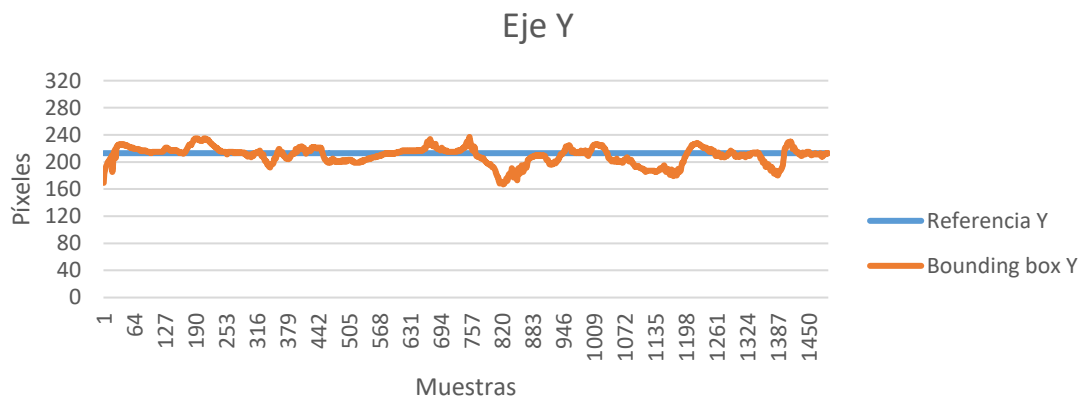


**Figura B.16** Respuesta del prototipo en el eje Y durante el ensayo 3 (Prueba 2)

- **Ensayo 4**

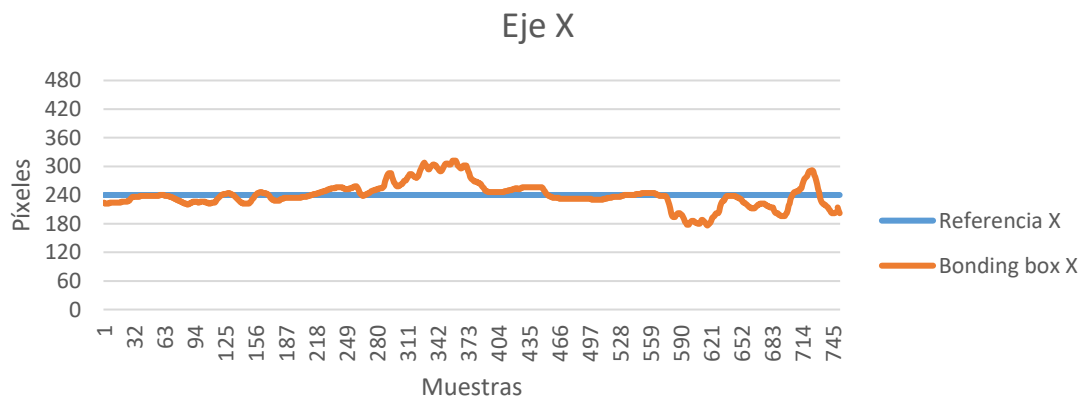


**Figura B.17** Respuesta del prototipo en el eje X durante el ensayo 4 (Prueba 2)

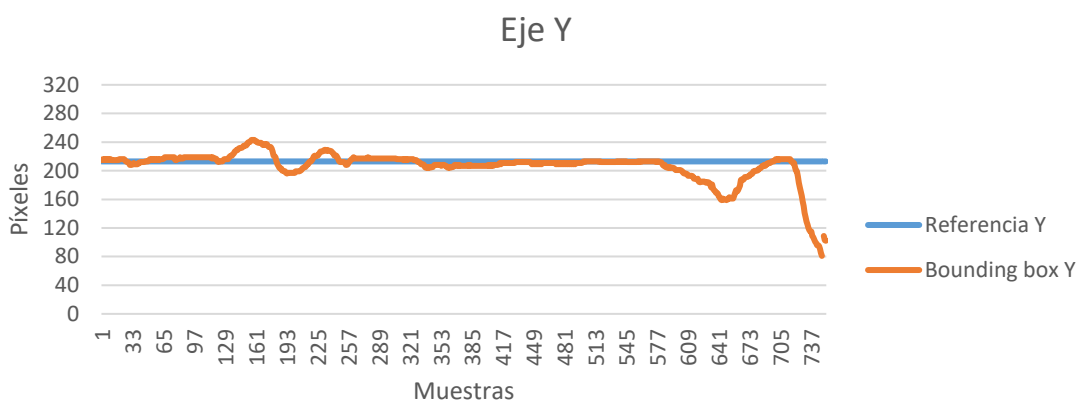


**Figura B.18** Respuesta del prototipo en el eje Y durante el ensayo 4 (Prueba 2)

- **Ensayo 5**

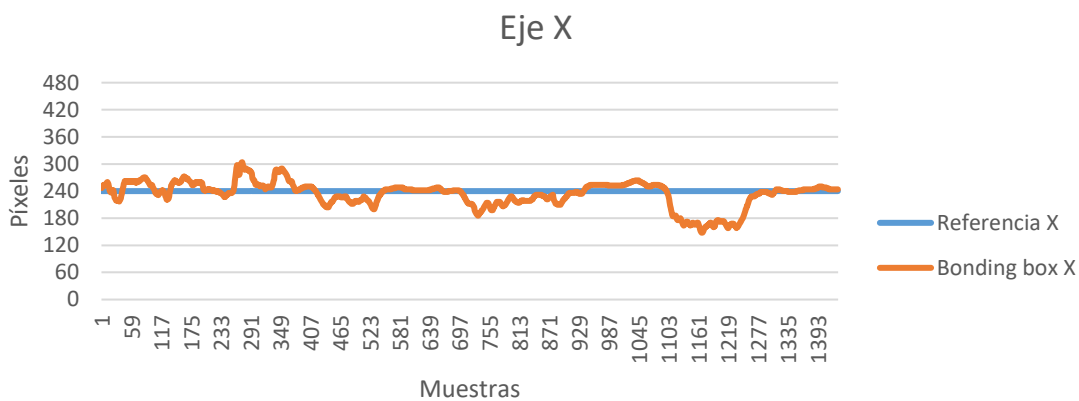


**Figura B.19** Respuesta del prototipo en el eje X durante el ensayo 5 (Prueba 2)

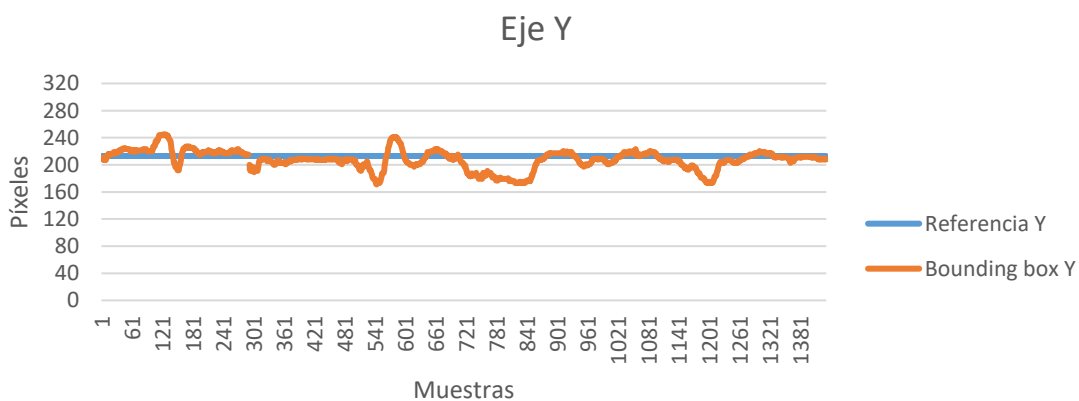


**Figura B.20** Respuesta del prototipo en el eje Y durante el ensayo 5 (Prueba 2)

**Ensayo 6**



**Figura B.21** Respuesta del prototipo en el eje X durante el ensayo 6 (Prueba 2)



**Figura B.22** Respuesta del prototipo en el eje Y durante el ensayo 6 (Prueba 2)

### PRUEBA 3

Finalmente, esta prueba consistió en seguir a la persona de interés en cualquier dirección, el único requerimiento fue que antes del despegue la persona se ubique frente al dron a una distancia de 3 metros aproximadamente para que al momento en que se levante el vuelo, el algoritmo sea capaz de reconocer correctamente a la persona con el chaleco naranja (véase la Sección 3.2.4.2.3 para más detalles). Para comprobar el correcto funcionamiento del prototipo se realizaron 6 vuelos o ensayos y se obtuvieron las gráficas desde la Figura B.23 a la Figura B.34.

#### - Ensayo 1

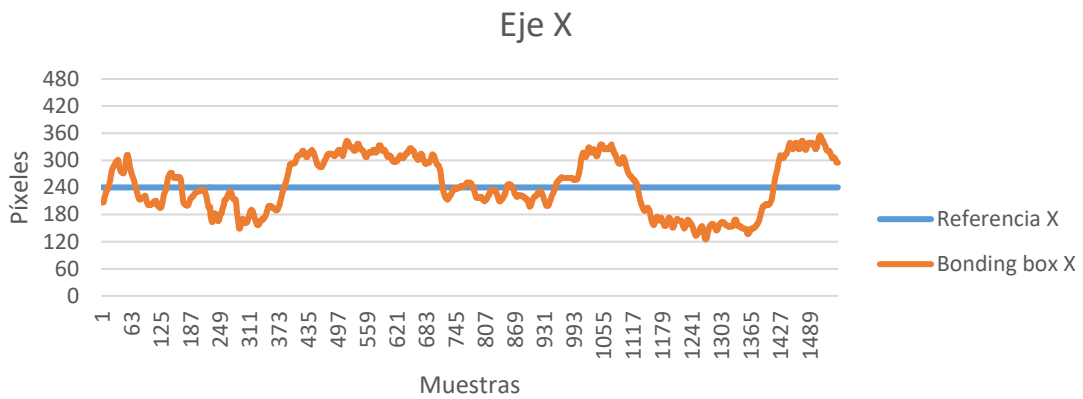


Figura B.23 Respuesta del prototipo en el eje X durante el ensayo 1 (Prueba 3)

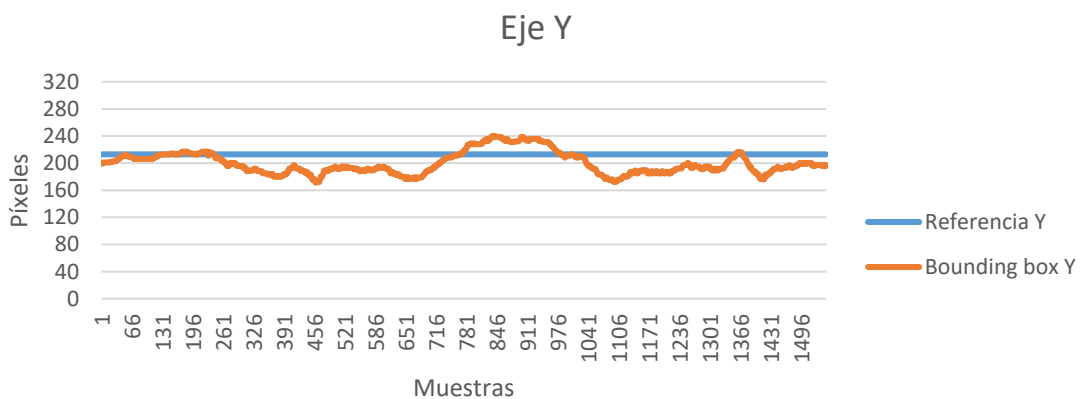
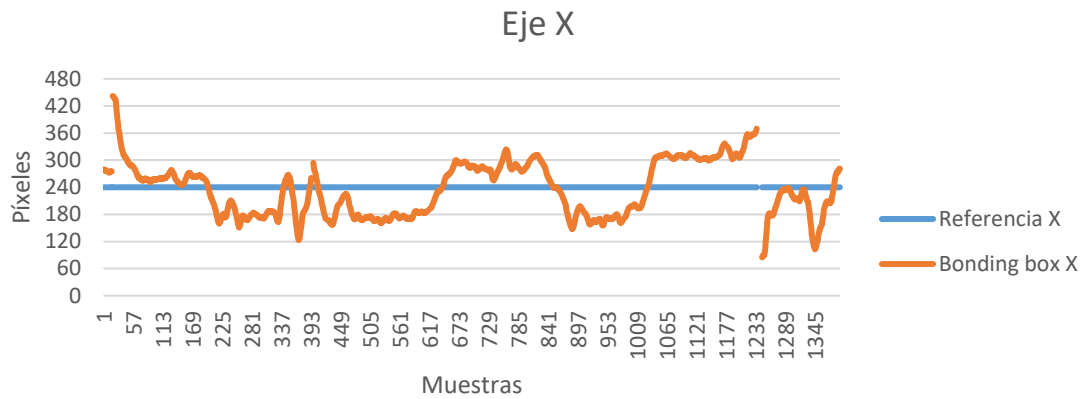
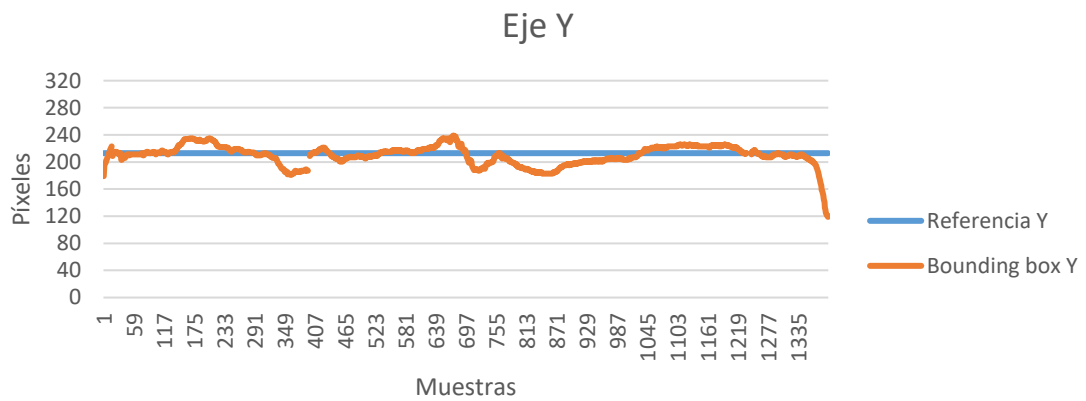


Figura B.24 Respuesta del prototipo en el eje Y durante el ensayo 1 (Prueba 3)

- **Ensayo 2**

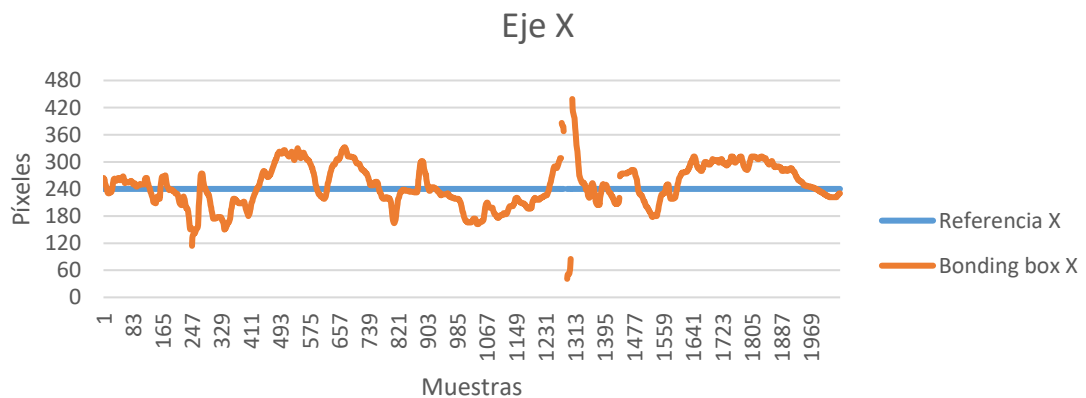


**Figura B.25** Respuesta del prototipo en el eje X durante el ensayo 2 (Prueba 3)

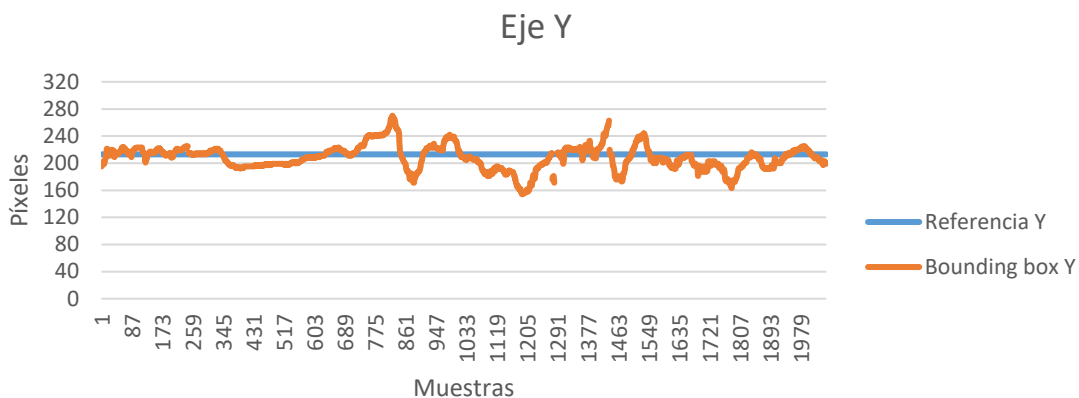


**Figura B.26** Respuesta del prototipo en el eje Y durante el ensayo 2 (Prueba 3)

- **Ensayo 3**

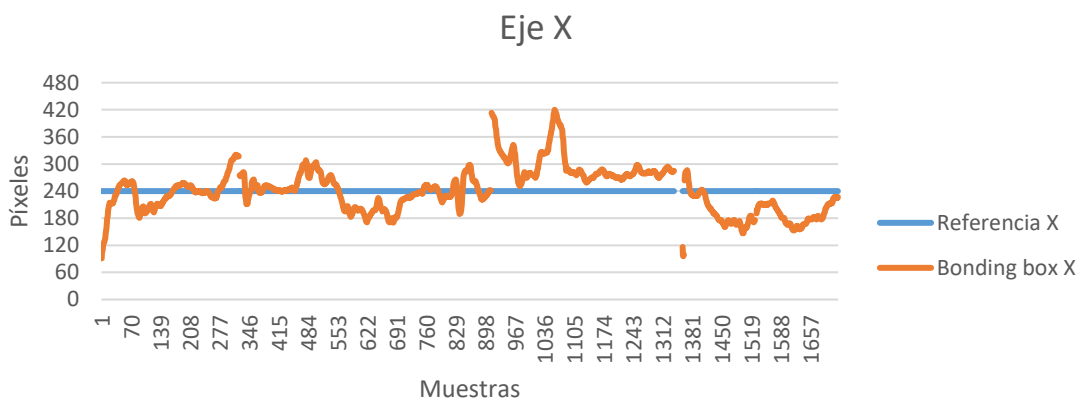


**Figura B.27** Respuesta del prototipo en el eje X durante el ensayo 3 (Prueba 3)

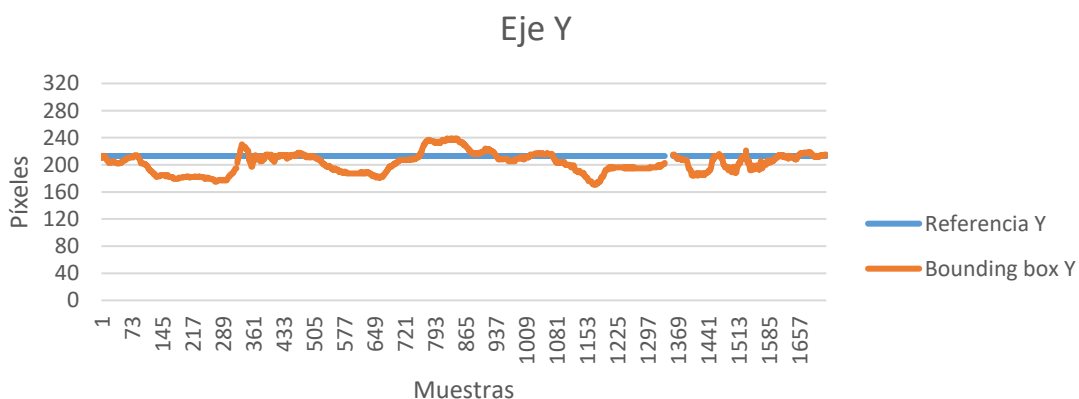


**Figura B.28** Respuesta del prototipo en el eje Y durante el ensayo 3 (Prueba 3)

- **Ensayo 4**

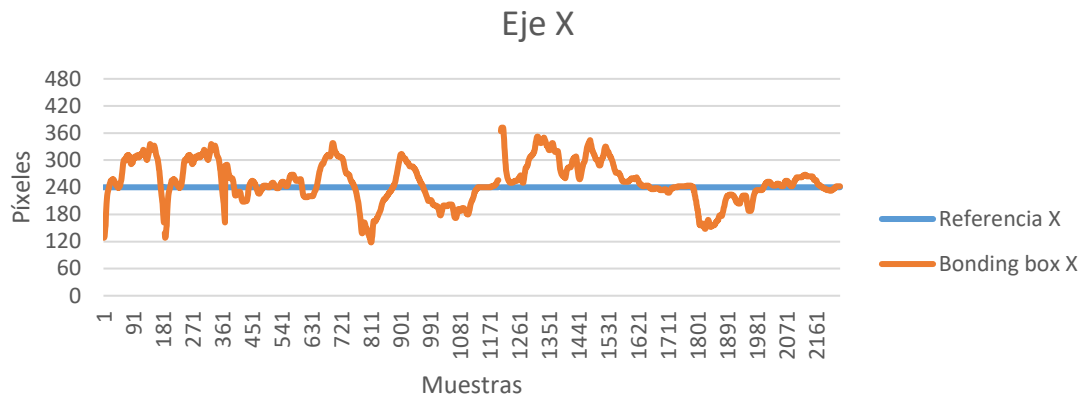


**Figura B.29** Respuesta del prototipo en el eje X durante el ensayo 4 (Prueba 3)

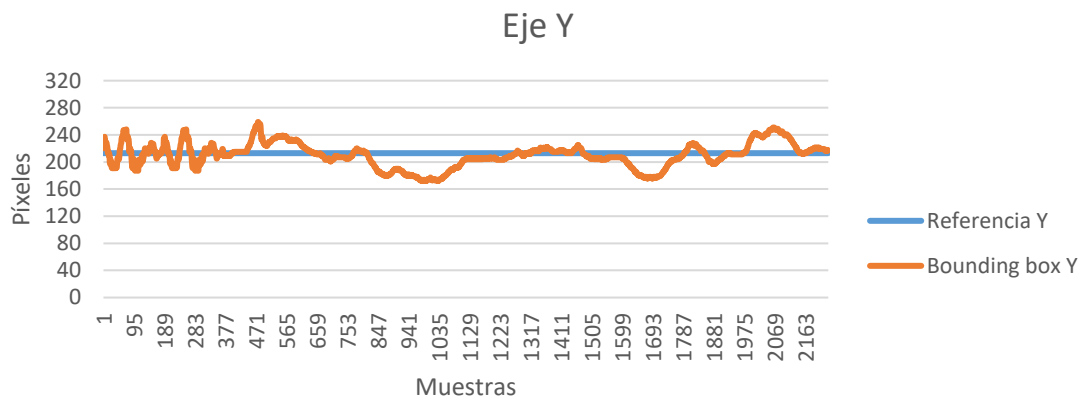


**Figura B.30** Respuesta del prototipo en el eje Y durante el ensayo 4 (Prueba 3)

- **Ensayo 5**

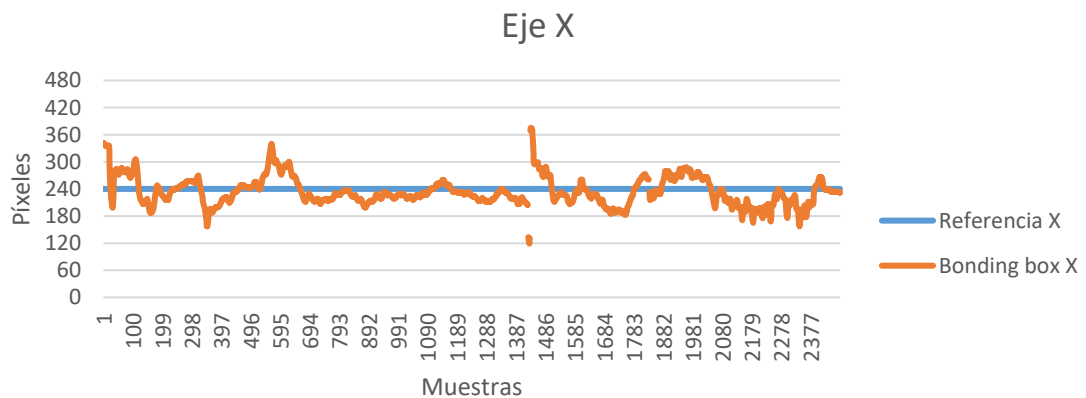


**Figura B.31** Respuesta del prototipo en el eje X durante el ensayo 5 (Prueba 3)



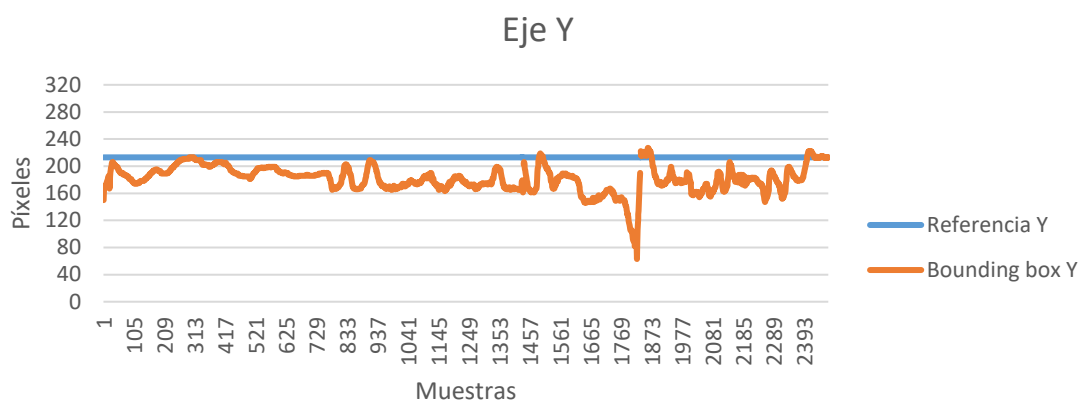
**Figura B.32** Respuesta del prototipo en el eje Y durante el ensayo 5 (Prueba 3)

- **Ensayo 6**



**Figura B.33** Respuesta del prototipo en el eje X durante el ensayo 6 (Prueba 3)





**Figura B.34** Respuesta del prototipo en el eje Y durante el ensayo 6 (Prueba 3)