UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACSISTEL

INGENIERÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

Implementación de un sistema inteligente de detección y seguimiento de objetos en movimiento a través de visión artificial, integrado en un vehículo aéreo no tripulado

Nicolás Alejandro Quirumbay Nieto

Dirigido por:
Ing. Junior Rafael Figueroa Olmedo, M.Sc.

La Libertad - 2025

DEDICATORIA

Dedico este título, fruto de esfuerzo y sacrificio, a mis padres, quienes me dieron la vida y siempre confiaron en mí. A mi padre, Joselito Quirumbay, quien se esforzó incansablemente para que nunca me faltara nada y fue un hombre comprometido con sus hijos, siempre presente en los momentos más difíciles. A mi madre, Dixy Nieto, que ha estado a mi lado en todo momento y hoy comparte conmigo esta felicidad; gracias por tu amor incondicional y por nunca soltar mi mano. A mi abuelita Norma, un pilar fundamental desde el inicio de mi vida, gracias por cuidarme con tanto amor y por mantenerme firme en los momentos más duros. Dedico también este logro a mi admirable Albita, esa flor que ha florecido a mi lado desde el inicio de mi carrera. Su belleza es fuente de inspiración, pero es su inteligencia la que guía y fortalece mi camino. Tu apoyo constante ha sido importante para llegar hasta aquí, gracias por estar siempre conmigo. A mis hermanos, quienes forman parte de mí, como los seis pétalos de una misma flor: Julián, Joseline, Verónica, Eduardo, Dominique y Necker. En cada uno de ustedes encuentro un motivo para seguir adelante, un vínculo que me sostiene con fuerza, una parte esencial de mi ser. Y finalmente, me lo dedico a mí mismo, porque he luchado, perseverado y hoy recojo el fruto de una de mis mayores metas; sé que me lo merezco y, sobre todo, sé que jamás les fallaré a las personas que amo. Mis inicios fueron humildes, pero sé que el futuro que me espera será tan grande como los sueños que llevo en el corazón.

AGRADECIMIENTO

Agradezco inmensamente a Dios por haberme guiado en este camino de la Ingeniería en Electrónica y Automatización, por darme la fuerza necesaria para continuar en los momentos difíciles, por transmitirme fe en mí mismo y por escuchar mis oraciones, en las que solo pedía que cada día fuera mejor. Agradezco profundamente a la Universidad Estatal Península de Santa Elena, el lugar donde me formé profesionalmente y que se convirtió en el escenario de muchos aprendizajes y vivencias que marcaron mi vida. A mi familia, especialmente a mi padre Joselito, gracias por cada sacrificio y por no dejar que me faltara ni un cuaderno en mi maleta; prometo no fallarte y convertirme en el profesional que siempre soñaste. A mi madre Dixy, por su cariño incondicional, su constante apoyo y por aquellos momentos en los que, con una canción entre sus labios, me regalaba paz sin saber que también estaba sembrando en mí la fortaleza para seguir adelante. A mi abuelita Norma, por ese amor inmenso que nunca nos faltó y por aquellos desayunos llenos de ternura que no solo alimentaban el cuerpo, sino que también alegraban el alma; gracias por estar siempre presente con tu calidez inigualable. A mis seis hermanos, quienes siempre estuvieron a mi lado, brindándome su amor, gracias por arrancarme sonrisas y regalarme momentos inolvidables llenos de alegría. A mi querida enamorada, Albita, gracias por estar conmigo desde el inicio de esta carrera; tu amor y apoyo constante me dieron la fortaleza necesaria para seguir adelante, por eso, este logro también es tuyo. Extiendo mi gratitud a tu familia, especialmente a tu madre y a tus hermanos, por el cariño con el que siempre me recibieron y por abrirme las puertas de su hogar, haciéndome sentir parte de él con cada gesto y palabra. A mi buen amigo Douglas Yagual, gracias por haber recorrido este largo camino a mi lado, así como lo empezamos juntos, lo terminamos juntos; has sido como un hermano, gracias por tu apoyo, tus palabras de ánimo y tu compañía en los momentos complicados. Gracias también a todos mis maestros de cada materia que me acompañaron a lo largo de estos años, por sus enseñanzas, dedicación y por aportar a mi crecimiento académico y personal. Sobre todo, agradezco al Ing. Junior Figueroa, tutor de este proyecto, por su guía, paciencia, enseñanzas y ser un gran maestro.

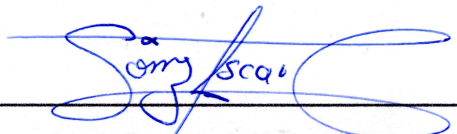
APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación denominado: **"Implementación de un sistema inteligente de detección y seguimiento de objetos en movimiento a través de visión artifical, integrado en un vehículo aéreo no tripulado"**, elaborado por el estudiante **Nicolás Alejandro Quirumbay Nieto**, de la carrera de **Ingeniería en Electrónica y Automatización** de la Universidad Estatal Península de Santa Elena, declaro que luego de haber orientado, estudiado y revisado, lo apruebo en todas sus partes y autorizo al estudiante que inicie los trámites legales correspondientes.

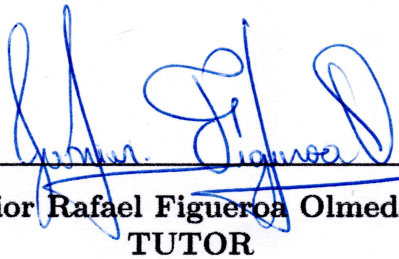
Ing. Junior Rafael Figueroa Olmedo, M.Sc.
Tutor

TRIBUNAL DE GRADO

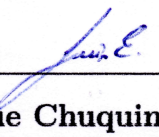
Ing. Ronald Humberto Rovira Jurado, Ph.D.
DIRECTOR DE CARRERA



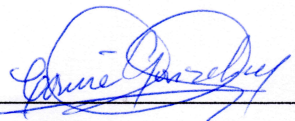
Ing. Oscar Gómez Morales , Mgtr.
DOCENTE ESPECIALISTA



Ing. Junior Rafael Figueroa Olmedo, Mgtr.
TUTOR



Ing. Luis Enrique Chuquimarca Jiménez, Mgtr.
DOCENTE GUÍA UIC



Ing. Corina Gonzabay De La A, Mgtr.
SECRETARIA DEL TRIBUNAL

DECLARACIÓN

Declaro que el contenido del presente trabajo de titulación es de mi entera responsabilidad, el patrimonio intelectual del mismo le pertenece a la Universidad Estatal Península de Santa Elena.

Nicolás.

Nicolás Alejandro Quirumbay Nieto.
Autor

Resumen

El presente trabajo desarrolla un sistema inteligente de detección y seguimiento de objetos implementado en un vehículo aéreo no tripulado (UAV). El sistema se basa en redes neuronales convolucionales (CNN), utilizando específicamente el modelo Yolov4 por su alta precisión y velocidad. Este modelo divide cada imagen capturada por la cámara del dron en una cuadrícula, y a cada celda le asigna la tarea de predecir cajas delimitadoras y sus probabilidades asociadas a determinadas clases de objetos. De esta forma, YOLOv4 logra identificar y localizar múltiples objetos simultáneamente en una sola pasada por la red neuronal. La implementación del algoritmo de detección y seguimiento se realizó en el entorno de desarrollo PyCharm, empleando el lenguaje de programación Python. El código incluye una interfaz gráfica que permite al usuario seleccionar uno de cuatro objetos desde un menú de opciones. Todo el sistema fue incorporado en un dron programable de la marca DJI Tello, el cual se conecta mediante su red wifi al computador. Una vez ejecutado el programa, el dron inicia un vuelo autónomo, realizando el seguimiento del objeto seleccionado. Durante el vuelo, el modelo YOLOv4 detecta en tiempo real la posición del objeto mediante coordenadas en la imagen, y con base en esta información, el sistema procesa las imágenes y envía comandos de movimiento al dron con el objetivo de mantener el objeto siempre centrado en el campo de visión, garantizando así un seguimiento continuo y preciso.

Palabras clave: Visión artificial, vehículo aéreo no tripulado, redes neuronales CNN, algoritmo de detección, procesamiento de imágenes.

Abstract

This work develops an intelligent object detection and tracking system implemented in an unmanned aerial vehicle (UAV). The system is based on convolutional neural networks (CNNs), specifically using the YOLOv4 model for its high accuracy and speed. This model divides each image captured by the drone's camera into a grid, and each cell is tasked with predicting bounding boxes and their associated probabilities for certain object classes. In this way, YOLOv4 is able to identify and locate multiple objects simultaneously in a single pass through the neural network. The detection and tracking algorithm was implemented in the PyCharm development environment, using the Python programming language. The code includes a graphical interface that allows the user to select one of four objects from a menu of options. The entire system was incorporated into a programmable DJI Tello drone, which connects to the computer via its Wi-Fi network. Once the program is executed, the drone begins an autonomous flight, tracking the selected object. During flight, the YOLOv4 model detects the object's position in real time using coordinates in the image. Based on this information, the system processes the images and sends movement commands to the drone with the aim of keeping the object always centered in the field of view, thus ensuring continuous and precise tracking.

Keywords: Computer vision, unmanned aerial vehicles, CNN neural networks, detection algorithm, image processing.

Índice

Índice de figuras **10**

Índice de cuadros **13**

1. Introducción **14**
1.1. Justificación . . . 14
1.2. Panorama actual . . . 15
1.3. Objetivos . . . 18
1.3.1. Objetivo General . . . 18
1.3.2. Objetivos específicos . . . 18
1.4. Fundamentos teóricos . . . 19
1.4.1. Robótica móvil aérea . . . 19
1.4.2. Vehículo aéreo no tripulado (UAV) . . . 19
1.4.2.1. Tipos de vehículos aéreos . . . 19
1.4.3. Principio de funcionamiento de un cuadricóptero . . . 22
1.4.3.1. Componentes básicos de un cuadricóptero . . . 24
1.4.4. Visión Artificial . . . 28
1.4.4.1. Componentes principales de la visión artificial . . . 28
1.4.4.2. Funciones principales en visión artificial . . . 34
1.4.5. Redes neuronales convolucionales (CNN) . . . 38
1.4.5.1. Funcionamiento de la red neuronal convolucional . . . 39
1.4.5.2. Red neuronal Yolo . . . 43
1.4.5.3. Métricas de detección de evaluación de objetos . . . 52
1.4.6. Lenguajes de programación para cuadricópteros . . . 53
1.4.7. Sistema de comunicación . . . 55
1.4.8. Protocolo de comunicación . . . 56
1.4.9. Normativas y reglamentos . . . 57
1.4.9.1. La Organización de Aviación Civil Internacional (OACI) . . . 57
1.4.9.2. La Agencia de la Unión Europea para la Seguridad Aérea (AESA) . . . 58
1.4.9.3. Reglamentación europea vigente en materia de drones 58
1.4.9.4. Normativas UAS – Regulaciones generales para el uso de drones en Ecuador . . . 59
1.4.9.5. Seguro de responsabilidad civil . . . 59
1.5. Marco contextual . . . 60

2. Métodos y diseño experimental **62**
2.1. Métodos . . . 62
2.1.1. Descripción del proyecto. . . . 62
2.2. Componentes de la propuesta . . . 63
2.2.1. Componentes físicos . . . 63
2.2.1.1. Dron Tello DJI . . . 63
2.2.1.2. Estructura y chasis . . . 64
2.2.1.3. Placa base . . . 65

2.2.1.4. Sistema de propulsión 65
2.2.1.5. Sensores 67
2.2.1.6. Sistema de alimentación 69
2.2.2. Componentes lógicos 70
2.2.2.1. Lenguaje de programación Python 70
2.2.2.2. Entorno de desarrollo integrado (IDE) 71
2.2.2.3. Librerías 72
2.3. Diseño experimental 74
2.3.1. Implementación del hardware 74
2.3.1.1. Adaptación del sistema de detección y seguimiento al hardware del dron 75
2.3.1.2. Procesamiento externo 76
2.3.1.3. Calibraciones del dron Tello DJI 76
2.3.2. Diseño e implementación del software 78
2.3.2.1. Modelo de detección YOLOv4 79
2.3.2.2. Configuraciones de los parámetros de la red Yolo . . 82
2.3.2.3. Funcionamiento interno del sistema inteligente aplicado al dron 87

3. Pruebas y Resultados **100**
3.1. Resultados 100
3.1.1. Evaluación del rango de detección del sistema 100
3.1.1.1. Evaluación de distancia para una botella 100
3.1.1.2. Evaluación de distancia para una taza 101
3.1.1.3. Evaluación de distancia para una tijera 103
3.1.1.4. Evaluación de distancia para un celular 104
3.1.2. Pruebas del sistema detección de objetos 105
3.1.2.1. Pruebas de detección de diferentes tipos de botellas . 105
3.1.2.2. Pruebas de detección de diferentes tipos de Tazas . . 106
3.1.2.3. Pruebas de detección de diferentes tipos de Tijeras . 107
3.1.2.4. Pruebas de detección de diferentes tipos de celulares 109

4. Conclusiones y recomendaciones **111**
4.1. Conclusiones 111
4.2. Recomendaciones 112

5. Bibliografía **114**

Referencias **114**

Referencias **120**

Índice de figuras

1. Vehículo aéreo no tripulado (UAV). [Autoría propia] 19
2. Aeronave con tres rotores. [Autoría propia] 20
3. Aeronave con cuatro rotores. [Autoría propia] 20
4. Aeronave con seis rotores. [Autoría propia] 21
5. Aeronave con ocho rotores. [Autoría propia] 21
6. : Fuerza de empuje. [Autoría propia] 22
7. Recta cuadrática . [Autoría propia] 22
8. Recta cuadrática. [Autoría propia] 23
9. Recta cuadrática. [Autoría propia] 23
10. fuerzas de impulsión del cuadricóptero. [Autoría propia] 23
11. Motor con escobillas y sin escobillas. 25
12. Tipos de hélices comunes. [Autoría propia] 25
13. Sentido de giro de las hélices. [Autoría propia] 26
14. Esquema del controlador eléctrico de velocidad. [Autoría propia] . . . 26
15. Esquema del Controlador de vuelo. [Autoría propia] 27
16. Transmisor y receptor de un cuadricóptero. [Autoría propia] 27
17. Tipos de baterías para cuadricópteros. [Autoría propia] 28
18. Visión artificial. [Autoría propia] 28
19. Iluminación frontal. 29
20. Iluminación en contra luz. 30
21. Iluminación difusa. 30
22. funcionamiento del lente de cámara de visión artificial. [Autoría propia] 31
23. Rango de espectro de luz en cámaras. 31
24. Rango de espectro de luz visible para el ojo humano. 32
25. Rango de espectro de luz termográfico. 32
26. Rango de espectro de luz hiperespectrales. 33
27. Comparación de resultados en distintas condiciones. [Autoría propia] 34
28. Etiqueta múltiple de un paisaje. [Autoría propia] 35
29. Segmentación umbral. 36
30. Segmentación por histograma. 36
31. Segmentación por detección de bordes. 37
32. Segmentación por cuencas hidrográficas. 37
33. Segmentación basada en clústeres. 38
34. Pixeles de entrada. 39
35. Matriz de pixeles normalizada en 0. 40
36. Matriz de pixeles normalizada en 1. [Autoría propia] 40
37. Estructura de las convoluciones. [Autoría propia] 40
38. Aplicación de filtro o kernel. 41
39. Aplicación de filtro o kernel. 41
40. Aplicación de subsamplingl con Max-Pooling 42
41. Arquitectura de una red neuronal. 43
42. Coordenadas de una imagen. 44
43. Arquitectura de la red original de YOLO. 44
44. Arquitectura de la red YOLOv4. 45
45. Convolución de la red yolov4. [Autoría propia] 46
46. Normalización de la red yolov4. [Autoría propia] 46

47. Activación Relu de la red yolov4. 47
48. Estructura CSP de la red yolov4. [Autoría propia] 48
49. Aplicación SPP de la red yolov4. [Autoría propia] 48
50. Aplicación Panet de la red yolov4. 49
51. Predicción de cuadros delimitador de la red yolov4. 50
52. Nivel de confianza de un objeto. [Autoría propia] 51
53. Objeto detectado (Persona). 51
54. Logo Python. 54
55. Logo C++. 54
56. Logo Java. 55
57. Diagrama general de bloques. [Autoría propia] 55
58. Diagrama general del sistema de detección y seguimiento de objetos. . 63
59. dron Tello DJI. 64
60. Estructura de dron Tello DJI. [Autoría propia] 65
61. Placa base del dron Tello DJI. [Autoría propia] 65
62. Motores sin escobillas del dron Tello DJI. [Autoría propia] 66
63. Hélices del dron Tello DJI. [Autoría propia] 67
64. Sensor de posicionamiento visual del dron Tello DJI. 68
65. Cámara frontal del dron Tello DJI. [Autoría propia] 68
66. Comparación de resultados en distintas condiciones. [Autoría propia] 69
67. Indicador de estado del dron Tello DJI. 70
68. Estructura interna del dron Tello DJI. [Autoría propia] 75
69. Detección del objeto dentro del encuadre. [Autoría propia] 76
70. Interconexiones del dron y el dispositivo de computo. [Autoría propia] 76
71. Calibración del IMU desde la aplicación. 77
72. Posiciónes secuenciales del dron para la calibración. 77
73. Calibración del centro de gravedad desde la aplicación. 78
74. Calibración del dron en curso. 78
75. División en cuadrícula del entorno. [Autoría propia] 79
76. Predicción de un objeto con una caja delimitadora. [Autoría propia] . 80
77. Predicción de clases de objetos. [Autoría propia] 80
78. Filtración de dos objetos. [Autoría propia] 82
79. División en 64 cuadros. [Autoría propia] 82
80. Procesamiento de 64 imágenes en un solo instante. [Autoría propia] . 83
81. Resolución de la imagen. [Autoría propia] 83
82. Canales de colores. [Autoría propia] 84
83. Red neuronal completa. [Autoría propia] 84
84. Penalización de los pesos de la red neuronal. [Autoría propia] 85
85. Rotación aleatoria. [Autoría propia] 85
86. Ajuste de saturación. [Autoría propia] 86
87. Fase del calentamiento del aprendizaje. [Autoría propia] 86
88. Diagrama de flujo del sistema. 88
89. Esquema de inicio del sistema. [Autoría propia] 89
90. Proceso de carga del modelo yolov4. [Autoría propia] 90
91. Interfaz gráfica del sistema. [Autoría propia] 90
92. Selección del objeto, botella. [Autoría propia] 91
93. Selección del objeto, taza. [Autoría propia] 91
94. Selección del objeto, tijera. [Autoría propia] 92

95. Selección del objeto, celular. [Autoría propia] 92
96. coordenadas X, Y y Z. 93
97. Objeto detectado en la parte lateral izquierda de la pantalla. 94
98. Objeto detectado en la parte lateral derecha de la pantalla. 94
99. Objeto detectado en la parte superior de la pantalla. 95
100. Objeto detectado en la parte inferior de la pantalla. 95
101. Objeto detectado lejos de la cámara del dron. 97
102. Objeto detectado cerca de la cámara del dron. 98
103. Aterrizaje del dron. 98
104. Análisis de distancia de una botella. 101
105. Análisis de distancia de una Taza. 102
106. Análisis de distancia de una Tijera. 103
107. Análisis de distancia de un celular. 104
108. Detección de 3 diferentes botellas. 105
109. Detección de 3 diferentes tazas. 106
110. Detección de 3 diferentes tijeras. 108
111. Detección de 3 diferentes celulares. 109

Índice de cuadros

1. Especificaciones técnicas del dron [1] 64
2. Especificaciones técnicas del motor sin núcleo [2] 66
3. Especificaciones técnicas de la hélice [1] 67
4. Especificaciones de captura de imagen y video [1] 69
5. Especificaciones de la batería [1] 70
6. Comandos básicos del dron DJI Tello usando Djitellopy [3] 72
7. Comandos de transmisión y captura de video con Djitellopy [3] 73
8. Comandos para obtener datos del dron con Djitellopy [3] 73
9. Parámetros evaluados en diferentes distancias 101
10. Parámetros evaluados en diferentes distancias 102
11. Parámetros evaluados en diferentes distancias 103
12. Parámetros evaluados en diferentes distancias 104
13. Resultados por tipo de botella 106
14. Resultados por tipo de taza 107
15. Resultados por tipo de tijera 108
16. Resultados por tipo de celular 109

1. Introducción

En la actualidad, los drones se han convertido en herramientas importantes para tareas de monitoreo y reconocimiento en distintos entornos, gracias a su capacidad para acceder a zonas de difícil alcance y recopilar información visual en tiempo real. Sin embargo, muchos de estos sistemas aún dependen del control manual mediante mandos a distancia, lo cual limita su autonomía y eficiencia operativa. Esta dependencia implica no solo la necesidad de supervisión constante por parte de un operador, sino también una menor capacidad de respuesta ante cambios dinámicos del entorno. Frente a estas limitaciones, surge la necesidad de desarrollar soluciones que incorporen inteligencia artificial y visión por computadora para permitir un comportamiento autónomo más avanzado. En este contexto, el presente proyecto plantea la implementación de un sistema de seguimiento de objetos en un vehículo aéreo no tripulado (UAV), mediante el uso de redes neuronales convolucionales (CNN) programadas en Python. Este sistema permitirá identificar, clasificar y seguir objetos de manera autónoma en tiempo real, utilizando una cámara frontal integrada y sensores de posicionamiento para ajustar su trayectoria según el movimiento del objetivo. Además de optimizar la precisión y velocidad del seguimiento, este desarrollo busca reducir la carga operativa humana y aumentar la fiabilidad del sistema. Su implementación no solo representa un avance técnico, sino también una oportunidad formativa significativa para los estudiantes de la UPSE, al integrar conocimientos en robótica, inteligencia artificial y control de sistemas, fomentando la investigación y el desarrollo en tecnologías emergentes con impacto académico e industrial.

1.1. Justificación

Hoy en día, el término robot.^{es}tá presente en todas partes, especialmente en el caso de los drones, que se han vuelto más comunes que antes. Actualmente, los drones se utilizan para monitorear diversas áreas, incluso para detectar animales o plantas. En este contexto, se utilizan sensores externos para detectar y seguir objetos, a veces incluso controlados manualmente con un mando a distancia. Esto presenta limitaciones, como en los sistemas de vigilancia, donde se requiere la presencia de un operador para controlar el cuadricóptero, lo que dificulta el cumplimiento de su función principal. Otra complicación frecuente es la falta de velocidad de respuesta, en comparación con los drones inteligentes actuales.

Con la ayuda de un cuadricóptero inteligente aumentaría la eficiencia de los procesos, ya que se reduciría la necesidad de emplear personal para realizar tareas de seguimiento manual. Además, mejorarían la calidad y rapidez de las operaciones realizadas por una persona, ya que el sistema automático puede seguir objetos de manera más constante y sin errores humanos. Por tal razón se busca implementar un sistema de seguimiento de objetos utilizando un cuadricóptero mediante visión artificial en tiempo real, con el fin de poder identificar y rastrear diferentes tipos de objetos para mejorar las funciones de un UAV de manera automática.

Esto no solo permitirá reducir la necesidad de personal para tareas manuales, sino que también garantizará una respuesta más rápida y precisa en comparación

con sistemas tradicionales, también permite explorar y desarrollar tecnologías avanzadas en el campo de la inteligencia artificial y la robótica y puede utilizarse para el desarrollo de sistemas autónomos, drones y vehículos autónomos.

El desarrollo de este proyecto en la UPSE ofrece a los estudiantes la oportunidad de avanzar en el campo de los drones y los algoritmos avanzados, proporcionándoles una base sólida para seguir el ritmo del progreso tecnológico, el cual es importante en diversas áreas de estudio y aplicación profesional. Este proyecto se concibe como el punto de partida para que los estudiantes puedan explorar y profundizar en tecnologías emergentes. A través de una analogía con la raíz de un árbol, se busca que este proyecto inicial actúe como fundamento para un crecimiento continuo. Al igual que una raíz, que necesita ser alimentada y regada para que el árbol crezca, este proyecto permitirá a los estudiantes seguir desarrollando sus habilidades y conocimientos, con la expectativa de que, en el futuro, puedan cosechar los frutos de este esfuerzo, obteniendo logros significativos en sus respectivas disciplinas.

1.2. Panorama actual

A lo largo de los años los drones han evolucionado conforme a los avances tecnológicos. Si bien es cierto, antes de que el inventor Abraham karem desarrolle los drones modernos que se conocen actualmente, se realizaron prototipos e intentos que datan de los años de 1849 en el mes de julio, donde por primera vez en la ciudad de Venecia se soltaron más de 200 globos aerostáticos no tripulados, que contenían bombas. Por consiguiente, los globos austriacos crean el primer precedente no como drones, sino los primeros inicios de vehículos aéreos no tripulados [4]. No obstante, en 1950 EE. UU. el uso de dispositivos no tripulados se empleó en misiones de vigilancia y reconocimiento en zonas de conflicto debido al gran beneficio que presentaba no poner en riesgo vidas humanas [5].

Es importante mencionar la IA (inteligencia artificial) principalmente el campo de visión artificial aplicado en cámaras, que se originó en la década de los sesenta mediante la realización de diferentes experimentos uno de ellos ejecutado por el científico estadounidense Larry Roberts que desarrolló un programa denominado "Mundo de Micro bloques" que consistía en el que un robot podía ver una estructura de bloques encima de una mesa, al verificar el contenido y reproducirlo desde otra perspectiva, consiguió que la imagen obtenida de la cámara se traslade al ordenador y fuera procesada debidamente [6].

De igual manera un grupo de neurofisiólogos efectuó un experimento que se basaba en mostrarle a un gato una serie de imágenes con el objetivo de comprender como las entendía y procesaba. Los resultados demostraban el procesamiento de imágenes comenzaba con formas simples como los bordes o líneas solidas [7]. Dentro de la década de los 2000 los drones con visión artificial empezaron a utilizarse en diferentes campos que brindan mejorar procesos civiles y comerciales en zonas industriales, como servicios públicos y energía, construcción, infraestructura, minería, gas petróleo, agricultura, militar y seguridad pública, investigación e inteligencia, también puede ser empleado como hobbie [8].

Empresas industriales presenta una mayor ventaja debido a la gran eficacia y calidad de realizar trabajos complicados como su capacidad de acceder a lugares estrechos o de difícil acceso, asimismo previniendo accidentes laborales realizados por el ser humano, aumentando la probabilidad de realizar inspecciones con más frecuencia y reduciendo costes incensarios en la implementación de otro equipo [9].

La evolución de la tecnología en los drones con visión artificial se aplicó por primera vez en usos militares que fueron utilizados para la vigilancia y reconocimiento, con el pasar de los años los UAV (Vehículo aéreo no tripulado) se han ido modificando, como sus componentes eléctricos, ahora son más pequeños y gracias a esto los drones son más accesibles en la civilización dando paso a diversas aplicaciones que hoy en día es utilizado con más frecuencia. La aplicación de IA en los drones ha mejorado sus capacidades como imágenes de gran resolución, transmisión de datos en tiempo real y sobre todo en su resistencia, permitiendo adentrarse en entornos más complejos y realizar estrategias para llevar misiones con un propósito [10].

Industrias agrícolas como Croplife Latin America aplican el manejo de drones que tienen componentes mejorados que les permiten rociar aguas, insecticidas o herbicidas a gran escala, además de visualizar si existe alguna problemática correspondiente a los cultivos, aportando un impacto positivo en la producción de alimentos. E incluso se realizan proyectos financiados en 2020 por parte de la Universidad de Murcia y Centro de Desarrollo Tecnológico e Industrial (CDTI) que ayudan a mejorar la calidad de los drones con la implementación de un sistema artificial y que cada algoritmo planteado debe realizarse para diferentes tipos de cultivo, cuyo objetivo es minimizar las problemáticas (plagas) existentes y optimizar un mejor proceso de producción [11].

También, las industrias de energía y recursos llevan a cabo inspecciones periódicas en plataformas marinas y líneas eléctricas utilizando drones, que alertan a los operadores sobre posibles catástrofes o peligros, como la rotura de una tubería. Estos dispositivos no solo resultan más económicos, sino que además son más respetuosos con el medio ambiente [12]. En este contexto, los robots UAV se han consolidado como una herramienta esencial en diversas industrias modernas, destacándose en sectores como la agricultura, la construcción, la minería y la seguridad. La incorporación de estas tecnologías ha transformado la forma en que se ejecutan múltiples tareas, permitiendo la automatización de procesos y la optimización de operaciones. Es importante señalar que las características y funcionalidades de los drones varían significativamente dependiendo del tipo de industria en que se apliquen [13].

En esta línea, el documento titulado “Monitoreo de Detección con Drones del Estado de Plantas de Arándanos en Invernaderos aplicando algoritmos de Deep Learning” describe la creación de un sistema de inspección autónomo mediante drones (UAV), cuyo objetivo es supervisar el estado de salud de plantas de arándano cultivadas en invernaderos. El propósito principal consiste en identificar señales tempranas de estrés, tales como manchas rojizas en las hojas o tallos más delgados, a través del uso conjunto de inteligencia artificial y robótica. El desarrollo del proyecto se estructura en cuatro fases: inicialmente, se lleva a cabo la recolección y el etiquetado de imágenes de plantas tanto sanas como afectadas, lo que da origen al conjunto

de datos. En una segunda etapa, se entrena una red neuronal convolucional (CNN) para clasificar dichas imágenes, y se integra un sistema de lógica difusa para mejorar la precisión de los resultados. Posteriormente, se incorporan marcadores ArUco y algoritmos de seguimiento de objetos para posicionar el dron de manera precisa, permitiendo la captura de imágenes desde distintos ángulos. Finalmente, se enfatiza el desarrollo de una solución económica y accesible que permita a los agricultores realizar monitoreos precisos, favoreciendo la detección oportuna de problemas y la adopción de medidas correctivas para minimizar pérdidas en la producción [14]. Aunque este enfoque comparte similitudes con el presente trabajo, como el uso de redes neuronales convolucionales, existen diferencias clave: mientras el sistema anterior utiliza marcadores ArUco como referencia visual, el sistema actual implementa un modelo especializado en la detección de formas, como podría ser el algoritmo YOLO.

Por su parte, el proyecto "Detección e identificación de formas utilizando imágenes tomadas de un UAV", desarrollado por Alejandro Ernesto Gómez Tamm, presenta un sistema basado en un dron equipado con una cámara integrada por el fabricante, que permite reconocer formas a partir de las imágenes obtenidas durante el vuelo. Una vez detectadas, el dron puede estimar la posición y distancia del objeto, mostrándolo en pantalla, lo cual abre posibilidades para aplicaciones como la localización de personas. Para validar el sistema, el autor emplea una simulación en ROS y Gazebo, donde se representa un entorno virtual en el que se identifican figuras geométricas básicas. Se utilizan OpenCV para el análisis de imágenes y RViz para la visualización de datos sensoriales. Este trabajo pone en evidencia las capacidades de los UAV en entornos simulados, así como las ventajas que ofrecen las RPAS frente a otras tecnologías en tareas de reconocimiento visual. Además, se describen los recursos utilizados y se evalúa el comportamiento del dron en vuelo autónomo, proponiendo mejoras futuras para ampliar su aplicabilidad en escenarios reales [15].

Finalmente, el proyecto "Sistema de visión artificial integrado a plataforma aérea para la detección de personas en tiempo real", desarrollado por Susana Andrea Parra Cabrera y William Ricardo Cuervo Lote, plantea la implementación de técnicas avanzadas de visión artificial aplicadas a un UAV con el fin de detectar personas en tiempo real. En este trabajo se combinan sensores y cámaras con distintos algoritmos, tales como la pirámide de imágenes, la ventana deslizante, Mean Shift, Histogram of Oriented Gradients (HOG) y You Only Look Once (YOLO), explicando sus fundamentos, ventajas y limitaciones. El uso de OpenCV resulta clave en este contexto, ya que permite implementar algoritmos como HOG, LBP y SVM de manera versátil en diversos lenguajes y plataformas. Asimismo, se recurre a CUDA para acelerar el procesamiento mediante GPU, lo cual incrementa significativamente la velocidad del análisis. Se destaca también la importancia de disponer de bases de datos específicas desde la perspectiva aérea para un entrenamiento adecuado del sistema. La interfaz fue desarrollada en Matlab, permitiendo generar un ejecutable accesible para su uso por parte de cualquier usuario interesado [16].

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar e implementar un sistema de detección y seguimiento de objetos utilizando un vehículo aéreo no tripulado de modo que permita realizar aplicaciones de visión artificial con aprendizaje automático.

1.3.2. Objetivos específicos

- Analizar un conjunto de datos del proyecto para determinar los equipos y algoritmos necesarios que se utilizarán en su desarrollo.
- Dimensionar el prototipo robótico móvil que presente las mejores características tanto en software como en hardware y que permita realizar la aplicación de detección y seguimiento de objetos.
- Analizar el algoritmo de visión artificial que cumpla con las características necesarias para llevar a cabo el desarrollo del proyecto.
- Implementar el algoritmo de visión artificial en un software que permita procesar y analizar datos en tiempo real.
- Realizar pruebas experimentales y físicas del funcionamiento del dron, llevando a cabo un análisis de errores para identificar y corregir posibles fallos en su desempeño.

1.4. Fundamentos teóricos

1.4.1. Robótica móvil aérea

La robótica aérea, representada principalmente por UAV (vehículos aéreos no tripulados), han cambiado la forma en que interactuamos con el espacio aéreo. Están disponibles en diferentes tamaños y diseños, lo que les permite realizar tareas diversas, desde capturar imágenes aéreas hasta transportar mercancías. Su diseño compacto y adaptable los convierte en herramientas clave en varios sectores [17].

Los robots aéreos tienen un amplio rango de aplicaciones debido a su capacidad de vuelo. En el ámbito militar, han sido utilizados desde los inicios de la aviación para vigilancia, selección de objetivos y ataques, logrando resultados significativos. En el ámbito civil, su uso es aún más variado, incluyendo teledetección, respuesta ante desastres, adquisición de imágenes, vigilancia, transporte y entregas. La robótica aérea sigue evolucionando y enfrentando desafíos para ampliar su integración en estos sectores [18].

1.4.2. Vehículo aéreo no tripulado (UAV)

Un UAV o dron es una aeronave capaz de operar de manera autónoma sin la necesidad de un piloto humano en su interior, estás aeronaves tiene dos tipos de variantes controlables: pueden operar de manera completamente autónoma utilizando algoritmos de vuelo automáticos, o pueden ser controladas de manera remota mediante un control a distancia [19].

Figura 1: Vehículo aéreo no tripulado (UAV). [Autoría propia]

1.4.2.1 Tipos de vehículos aéreos

No todos los drones son iguales. Sus diferencias radican en la estructura, el número de rotores y la tecnología que incorporan, lo que influye en su estabilidad, autonomía y rendimiento en vuelo. A continuación, se describirán los tipos más utilizados, clasificándolos según el número de hélices.

Tricópteros: El tricóptero es un tipo de dron que posee tres brazos (Figura 2), cada uno con un motor. Los motores en los brazos delanteros giran en direcciones contrarias para generar el empuje necesario y mantener el equilibrio del aparato. En el caso del motor trasero, este funciona como un servomotor, permitiendo ajustes en su inclinación para garantizar mayor estabilidad y control durante el vuelo. A diferencia de los cuadricópteros, los tricópteros tienen menos puntos de apoyo, lo que requiere un sistema de control más delicado y preciso. Esto les otorga mayor agilidad, pero a su vez presenta un desafío adicional para su estabilización [20].

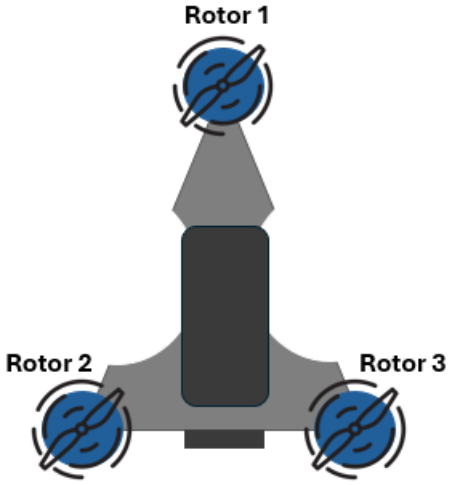


Figura 2: Aeronave con tres rotores. [Autoría propia]

Cuadricópteros: El cuadricóptero es uno de los tipos de drones más comunes y utilizados en la actualidad. Su diseño incluye cuatro brazos (Figura 3), cada uno con un motor, lo que le permite generar el empuje necesario para volar de manera estable. Gracias a la disposición equidistante de sus motores, este tipo de dron logra un equilibrio óptimo, lo que le confiere una gran estabilidad en el aire. Esta configuración hace que sea más fácil de controlar y maniobrar, lo que lo convierte en una opción ideal tanto para principiantes como para aplicaciones profesionales [20].

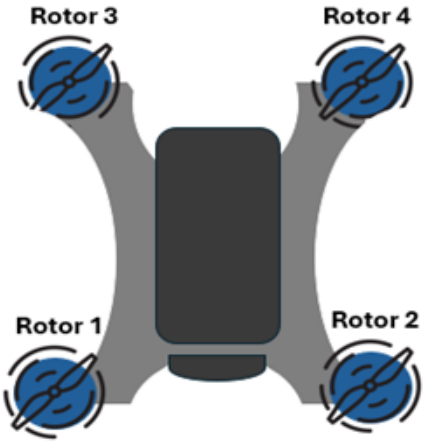


Figura 3: Aeronave con cuatro rotores. [Autoría propia]

Héxacópteros: El hexacóptero es un tipo de dron que se caracteriza por tener

seis brazos (Figura 4), cada uno con su respectivo motor y hélice. Esta configuración multirrotor le permite tener una mayor estabilidad y capacidad de carga. Además, al contar con más motores, el hexacóptero ofrece redundancia, lo que significa que puede seguir volando incluso si uno de sus motores falla, lo que aumenta la seguridad en vuelos críticos. Su mayor capacidad de elevación también les permite llevar más equipos adicionales, como cámaras de mayor tamaño o sensores avanzados [21].

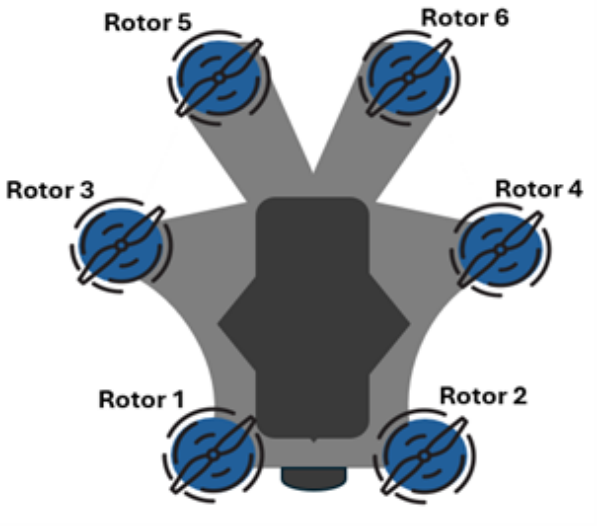


Figura 4: Aeronave con seis rotores. [Autoría propia]

Octocópteros: El octocóptero se distingue por tener ocho rotores (Figura 5), lo que le brinda una estabilidad y potencia superiores en comparación con otros drones como los cuadricópteros o hexacópteros. Esta configuración le permite soportar cargas más pesadas y resistir mejor las condiciones de viento. Su diseño avanzado lo hace ideal para tareas profesionales. Sin embargo, debido al mayor número de motores, el octocóptero consume más energía y tiene un tamaño más grande, lo que puede limitar su tiempo de vuelo y su maniobrabilidad en espacios pequeños. Pese a estas desventajas, su capacidad para cargar equipos pesados y su estabilidad en vuelos complejos por lo que es una opción preferida en aplicaciones que exigen altos estándares de rendimiento [22].

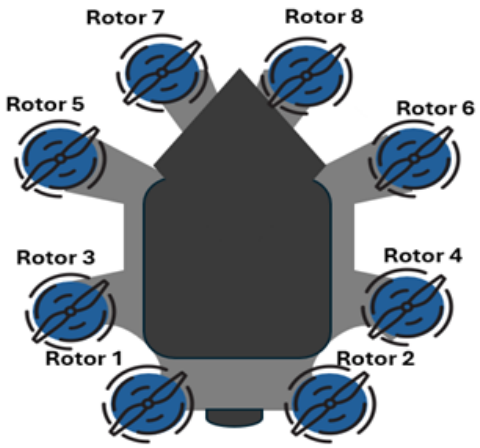


Figura 5: Aeronave con ocho rotores. [Autoría propia]

1.4.3. Principio de funcionamiento de un cuadricóptero

El cuadricóptero tiene cuatro rotores que permiten sostener su peso. Cada uno de estos rotores gira para producir una fuerza de empuje, que es importante para que el dron se mantenga en el aire. Esta fuerza, será representar con la letra "F"[23].

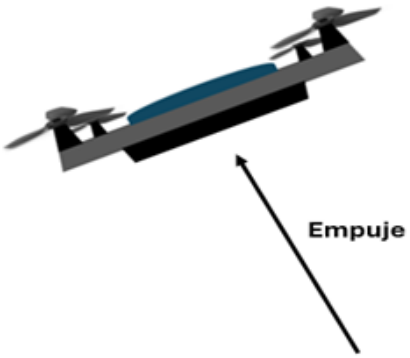


Figura 6: : Fuerza de empuje. [Autoría propia]

La relación entre el empuje producido por los motores de un dron y la velocidad a la que giran A_V, sigue un patrón cuadrático. Esto significa que a medida que aumenta la velocidad de los motores, el empuje también se incrementa, pero de manera más pronunciada, ya que el empuje es proporcional al cuadrado de la velocidad angular. Matemáticamente, esta relación se puede expresar como $F = K \cdot A_V^2$, en donde K es una constante [23].

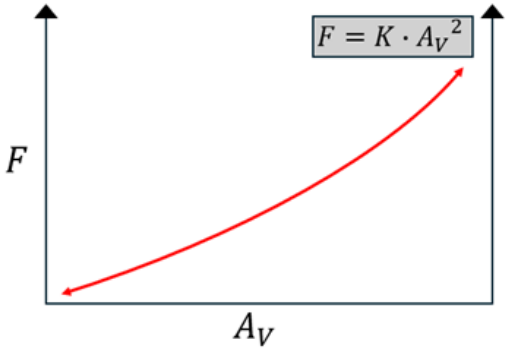


Figura 7: Recta cuadrática . [Autoría propia]

Cada vez que un rotor gira, debe superar la resistencia del aire. El momento, que es la capacidad para hacer girar un objeto, está relacionado de forma cuadrática con las revoluciones por minuto (RPM). Esto implica que el momento de resistencia T sigue la fórmula $T = K \cdot A_V^2$ [23].

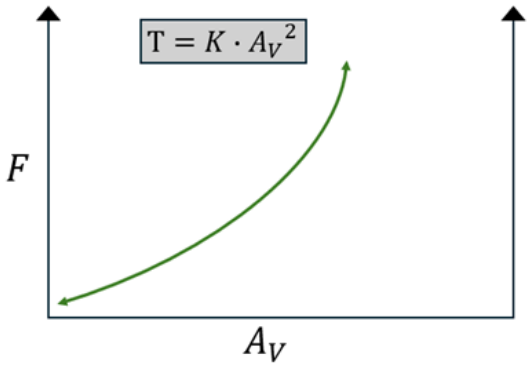


Figura 8: Recta cuadrática. [Autoría propia]

Como la aeronave tiene cuatro rotores, cada uno tiene un peso de 1/4 de la totalidad del cuadricóptero manteniendo su vuelo estable. Entonces, si volvemos al gráfico 8 (empuje vs. RPM), obtenemos la velocidad de giro de los cuatro motores ($S = \frac{1}{4}m \cdot g$), en donde m es la masa del cuadricóptero y g es la aceleración de la gravedad [23].

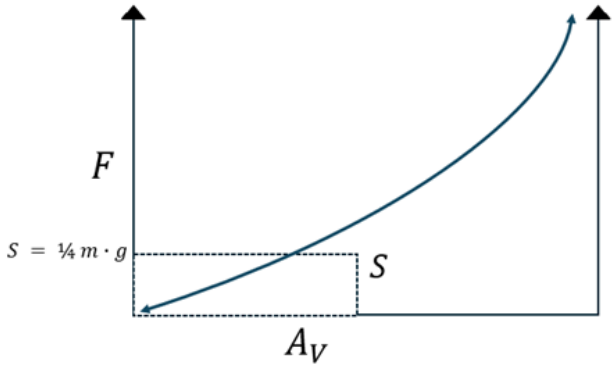


Figura 9: Recta cuadrática. [Autoría propia]

El motor necesita producir la fuerza necesaria para vencer la resistencia al giro. Cuando el cuadricóptero se mantiene en el aire sin moverse, la rotación de los cuatro rotores contrarresta su peso. Si se determina la velocidad (S) de cada uno de los rotores, es posible calcular la fuerza de torsión que producen [23].

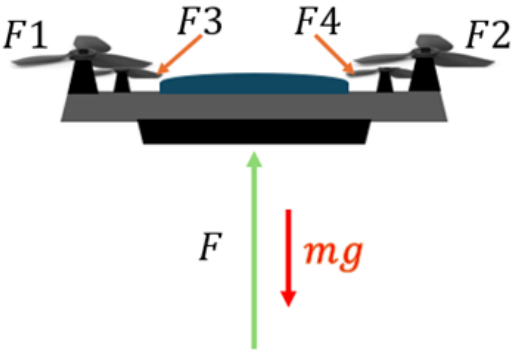


Figura 10: fuerzas de impulsión del cuadricóptero. [Autoría propia]

La fuerza total que impulsa al cuadricóptero es el resultado de combinar las fuerzas generadas por los cuatro rotores, restando el efecto de la gravedad. Es decir, para que el cuadricóptero se eleve, la suma del empuje de los rotores debe ser mayor que el peso del aparato (Fg = mg). En vuelo estacionario, ambas fuerzas se igualan, manteniendo el cuadricóptero en el aire sin ascender ni descender [23].

$$F = F_1 + F_2 + F_3 + F_4 - mg \tag{1}$$

Una vez que se conoce el centro de masa del cuadricóptero, es más fácil realizar el cálculo del momento total (par) Alrededor del punto donde se equilibra la suma de las fuerzas de empuje [23].

$$M = r_1 \cdot F_1 + r_2 \cdot F_2 + r_3 \cdot F_3 + r_4 \cdot F_4 + M_1 + M_2 + M_3 + M_4 \tag{2}$$

Los términos $M_1 + M_2 + M_3 + M_4$ corresponden a los efectos de giro producidos por cada rotor, mientras que $r_1 \cdot F_1 + r_2 \cdot F_2 + r_3 \cdot F_3 + r_4 \cdot F_4$ representan las fuerzas de reacción generadas por la rotación de los rotores en direcciones opuestas [23].

1.4.3.1 Componentes básicos de un cuadricóptero

Los componentes de un cuadricóptero son importantes para su funcionamiento y rendimiento, ya que permiten que el vehículo aéreo mantenga el vuelo, se estabilice y ejecute una variedad de tareas. La interacción de estos elementos asegura que el cuadricóptero pueda volar de manera controlada y segura en diferentes condiciones. Conocer el papel que juega cada uno de estos componentes es fundamental para entender su operación. A continuación, exploraremos los componentes básicos que conforman un cuadricóptero y cómo contribuyen a su rendimiento general.

Chasis: El chasis de un UAV actúa como el cuerpo que sostiene todos sus componentes. Es el elemento encargado de conectar y fijar las partes del dron en una disposición que favorezca su rendimiento en el aire. Además, el diseño del chasis influye en el tamaño total del dron, asegurando que cada pieza esté bien colocada y obtener un buen funcionamiento aerodinámico Los brazos, que cumplen la función de mantener los motores sujetos a la estructura principal, juegan un papel clave en el comportamiento del dron. La longitud de los brazos afecta a este comportamiento, ya que los brazos largos proporcionan mayor estabilidad, mientras que los cortos mejoran la maniobrabilidad [24].

Motor: El motor genera el empuje necesario para elevar y mantener un buen balance entre el empuje y el peso ya que es importante para la estabilidad y maniobrabilidad. Para que el dron ascienda, necesita producir suficiente empuje para superar tanto la resistencia del aire como la gravedad. Los motores utilizados comúnmente se dividen en dos tipos principales: los motores con escobillas (Figura 11, A) y los sin escobillas (Figura 11, B). Ambos funcionan al utilizar corriente eléctrica. La diferencia entre los dos tipos es que los motores sin escobillas utilizan un sistema electrónico para la conmutación de la corriente, lo que les otorga mayor eficiencia y una vida útil más larga. En cambio, los motores con escobillas dependen de componentes mecánicos que al paso del tiempo puede traer problemas, lo que requiere de un mantenimiento frecuente [25].

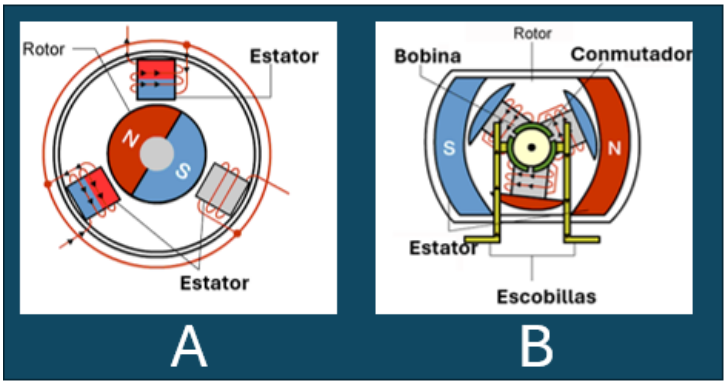


Figura 11: Motor con escobillas y sin escobillas.
[Autoría propia]

Hélices: Las hélices de los drones son partes giratorias que crean la fuerza necesaria para que el cuadricóptero se eleve al mover el aire con su diseño aerodinámico. En la Figura 12 se observan distintos tipos de hélices, tales como los de dos palas, que son livianas y adecuadas para recorridos cortos, los de tres palas, que ayudan a mantener mejor el control en condiciones de viento, y los de cuatro palas, que permiten más fuerza, aunque requieren más energía. El tamaño de la hélice también influye en el desempeño, ya que las más grandes permiten mayor elevación, mientras que las más pequeñas hacen que el dron sea más veloz [26].

Figura 12: Tipos de hélices comunes. [Autoría propia]

En un cuadricóptero, las hélices delanteras y traseras giran en sentidos contrarios como se muestra en la Figura 13, esto se elaboró para evitar que el dron rote sin control y mantener su estabilidad. Este principio también se aplica en hexacópteros y octocópteros, donde la disposición de las hélices sigue un patrón similar [27].

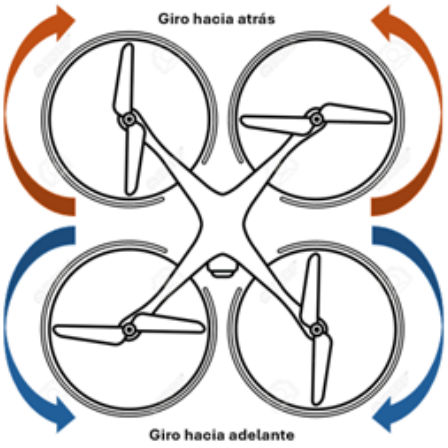


Figura 13: Sentido de giro de las hélices. [Autoría propia]

Controlador eléctrico de velocidad (ESC): El (ESC) está encargado de regular la velocidad de los motores, que permite un control de vuelo preciso. En los cuadricópteros, cada motor está conectado a su propio ESC, que ajusta la velocidad de rotación según las señales del controlador de vuelo, para obtener una estabilidad y respuesta rápida. Además de controlar la velocidad, el ESC también gestiona la dirección de rotación el cuál protege contra sobrecalentamientos, cortocircuitos, y contribuye a un vuelo controlado al mantener el balance general del cuadricóptero [23].

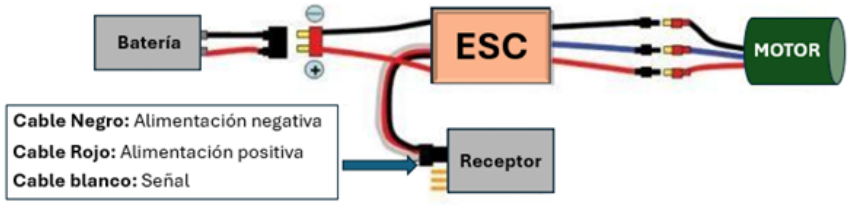


Figura 14: Esquema del controlador eléctrico de velocidad. [Autoría propia]

Controlador de vuelo: El controlador de vuelo es el componente principal de un dron, actúa como su çerebro.ªl procesar información de diversos sensores y convertir las órdenes del piloto en movimientos precisos. Esta unidad central combina sensores, software y comunicación para mantener la estabilidad y respuesta del dron. Su función principal es mantener el equilibrio del vuelo mediante giroscopios y acelerómetros, interpreta comandos de control remoto o de un sistema automático para ajustar los motores, permite la navegación con GPS en algunos casos y comunicarse con otros sistemas como cámaras o módulos de telemetría. Además, existen diferencias clave entre los distintos controladores de vuelo y sus componentes, lo que influye en el desempeño y control de las aeronaves no tripuladas [28]. La Figura 15 muestra el esquema del controlador de vuelo de un dron.

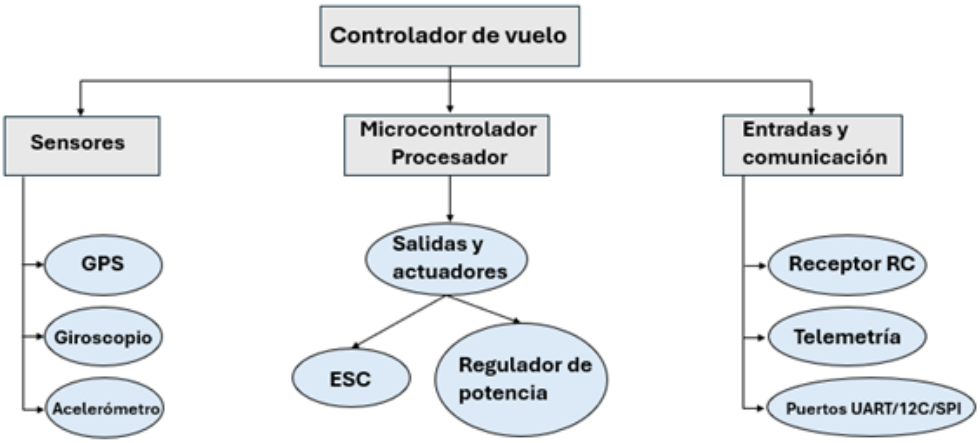


Figura 15: Esquema del Controlador de vuelo. [Autoría propia]

Transmisor y Receptor: El transmisor es el centro de control principal del dron, operado por el piloto desde el suelo. Utiliza ondas de radio para enviar señales codificadas que corresponden a los comandos específicos del piloto como se muestra en la Figura 16, tales como cambios en la velocidad, altitud, dirección y maniobras complejas. Estos comandos se transmiten generalmente a través de joysticks o controles remotos, que ofrece al piloto un control preciso sobre el movimiento del cuadricóptero. El receptor, que se encuentra montado en el dron, actúa como un ïntérprete"de estas señales de radio. Al recibir las ondas de radio entrantes, las decodifica y las convierte en instrucciones procesables para el controlador de vuelo del dron. Además, el transmisor y receptor trabajan dentro de frecuencias específicas, lo que permite la comunicación estable incluso en condiciones de interferencia, gracias a esto, los datos se transmiten de manera correcta y sin retrasos. Este sistema de comunicación bidireccional no solo permite controlar el dron, sino también recibir retroalimentación sobre su estado [29].

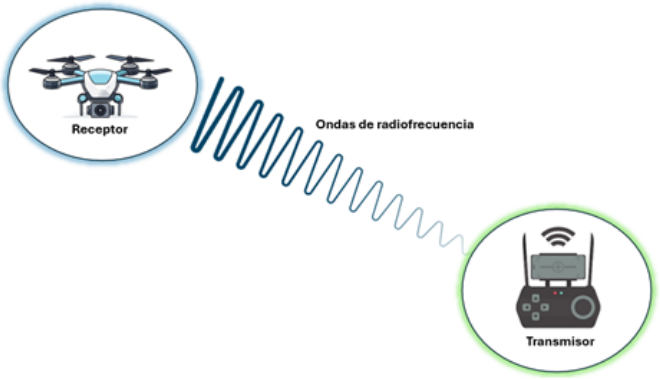


Figura 16: Transmisor y receptor de un cuadricóptero. [Autoría propia]

Baterías: Una batería de cuadricóptero es un dispositivo indispensable que convierte la energía química en eléctrica, proporcionando la potencia necesaria para que el dron funcione, alimentando motores, luces y sistemas electrónicos. Existen varios tipos de baterías como se muestra en la Figura 17, cada una con características particulares que se ajustan a diferentes necesidades. Las baterías de polímero de litio (LiPo) son las más populares debido a su excelente relación entre peso y energía, lo que permite un alto rendimiento en el vuelo. Las baterías de iones de litio (Li-ion),

aunque tienen una densidad energética ligeramente inferior, destacan por su mayor vida útil, lo que las hace ideales para vuelos largos. Finalmente, las baterías de níquel-metal hidruro (NiMH), a pesar de ser más pesadas y tener menos potencia, ofrecen una gran durabilidad y la capacidad de soportar múltiples ciclos de carga, lo que las hace apropiadas para ciertos usos específicos [30].

Figura 17: Tipos de baterías para cuadricópteros. [Autoría propia]

1.4.4. Visión Artificial

La visión artificial es una disciplina tecnológica que proporcionan a las máquinas de la capacidad de ver y analizar imágenes digitales. Esta tecnología permite a los dispositivos no solo extraer información visual, sino también interpretar escenas y realizar tareas específicas basadas en lo que observan. Utilizando diversos tipos de sensores, la visión artificial puede captar detalles con gran precisión, lo que la hace indispensable en la industria. Sin embargo, no actúa como una herramienta aislada, sino que se integra en toda la cadena de producción, facilitando la comunicación y el intercambio de datos. Esta capacidad para procesar imágenes de manera similar a la visión humana, aunque con características únicas, está estrechamente vinculada al desarrollo de la Inteligencia Artificial [31].

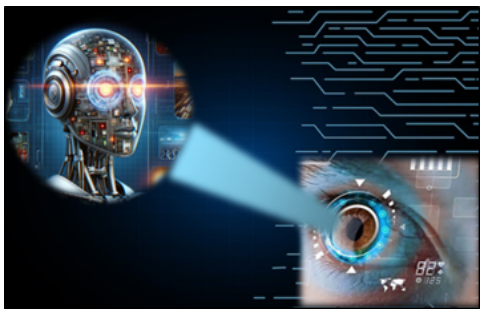


Figura 18: Visión artificial. [Autoría propia]

1.4.4.1 Componentes principales de la visión artificial

Los componentes fundamentales de un sistema de visión artificial trabajan en conjunto para capturar, procesar y analizar la información visual, permitiendo que el sistema extraiga características clave de las imágenes y tome decisiones informadas. Aunque la cámara es el componente principal de cualquier sistema de visión artificial, su funcionamiento completo depende de la integración de otros componentes

importantes que apoyan la adquisición y el análisis de las imágenes. A continuación, se describirán los componentes que estructuran la visión artificial.

Iluminación

Cuando un sistema de visión artificial observa un objeto, la luz interactúa con él de diversas maneras, como reflexión, absorción, transmisión, difusión y refracción. La luz que se refleja en el objeto es capturada por la cámara, y dependiendo de las propiedades del objeto, como su color y textura, se pueden generar diferentes intensidades y patrones de luz. Los objetos oscuros absorben más luz, creando sombras, mientras que los objetos brillantes reflejan más, destacando detalles. Los objetos transparentes o translúcidos permiten que la luz los atraviese, lo que puede dificultar su identificación. Además, la luz puede dispersarse al chocar con superficies rugosas, o cambiar de dirección al pasar por medios de diferente densidad. Estas interacciones son procesadas por el sistema de visión artificial para interpretar la forma, tamaño, contorno y otras características del objeto [32]. Existen diversos tipos de iluminación utilizados en visión artificial, cada uno adaptado a necesidades específicas dependiendo de la aplicación y las características del objeto a analizar como, por ejemplo:

- **Iluminación frontal:** La iluminación frontal en visión artificial se refiere a la luz que se dirige directamente hacia el objeto desde una posición frontal, es decir, desde el mismo ángulo en que se observa el objeto. Este tipo de iluminación es comúnmente utilizada para iluminar de manera uniforme las superficies de los objetos, destacando sus detalles y características más visibles. Al iluminar de frente, se pueden minimizar las sombras profundas que podrían interferir con la visión del objeto como se ilustra en la Figura 19, pero, en ocasiones, puede generar reflejos o destellos, especialmente en superficies brillantes. Esta técnica es útil para aplicaciones en las que se necesita una imagen clara y detallada de la superficie del objeto [33].

Figura 19: Iluminación frontal.

Fuente: [33].

- **Iluminación en contraluz:** La iluminación en contra luz en visión artificial ocurre cuando la fuente de luz se coloca detrás del objeto que se está observando, lo que crea un contraste fuerte entre el objeto y su fondo. Este tipo de iluminación es útil para resaltar los contornos y bordes de un objeto, ya que genera una silueta que permite visualizar claramente su forma como se observa en la Figura 20. La luz que pasa a través de objetos translúcidos o

semitransparentes también puede resaltar detalles internos, lo que es especialmente valioso en aplicaciones como el escaneo de estructuras complejas o la detección de defectos invisibles a simple vista [33].

Figura 20: Iluminación en contra luz.

Fuente: [33].

- **Iluminación difusa:** La iluminación difusa en visión artificial se refiere a una luz que se dispersa de manera uniforme sobre un objeto, en lugar de proyectarse directamente desde una fuente puntual. Este tipo de iluminación suaviza las sombras y reduce los reflejos brillantes, proporcionando una imagen más clara y equilibrada del objeto como se muestra en la Figura 21. Al eliminar las sombras intensas, la luz difusa ayuda a resaltar detalles y características más finas de la superficie del objeto sin generar áreas demasiado iluminadas u oscuras. Además, este tipo de iluminación minimiza las alteraciones en la imagen causadas por fuentes de luz directas, Sin embargo, a veces puede perderse la profundidad o los detalles tridimensionales, ya que no se generan sombras marcadas que ayuden a definir la forma del objeto [33].

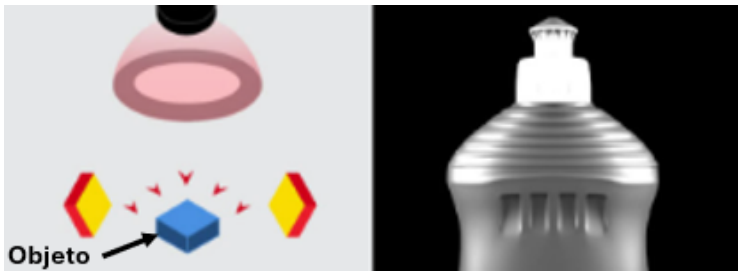


Figura 21: Iluminación difusa.

Fuente: [33].

Lente

La lente cumple una función esencial al enfocar la luz que llega del objeto hacia el sensor de la cámara, permitiendo que la imagen capturada sea clara y detallada. Su tarea principal es asegurar que los elementos del objeto estén correctamente enfocados, por lo que es importante para obtener una imagen nítida. Además, la lente debe de estar a una distancia de trabajo adecuada, la resolución requerida y el aumento adecuado [34]. En los sistemas de visión artificial, las lentes suelen contar con un enfoque fijo o ajustable, a diferencia de las cámaras comunes, que generalmente

incluyen enfoque automático. Al clasificar las lentes, se emplean diversos términos como ángulo de visión (AoV), campo de visión (FoV), distancia del objeto, distancia focal, apertura y F-Stop, los cuales son cruciales para entender sus características y capacidades [35]. En la Figura 22 se ilustra el funcionamiento de la lente la cámara de visión artificial.

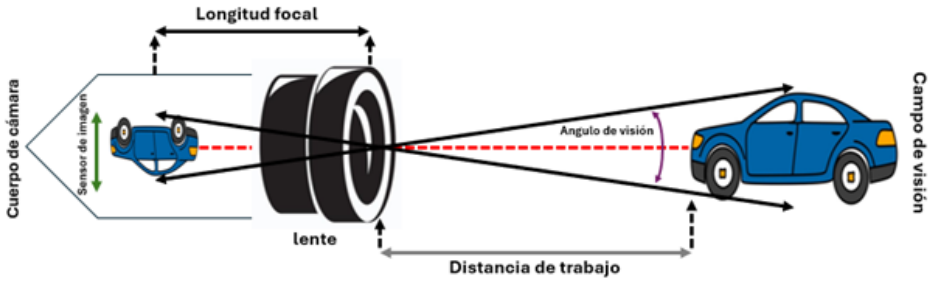


Figura 22: funcionamiento del lente de cámara de visión artificial. [Autoría propia]

Cámara

Una cámara de visión artificial es un dispositivo con una carcasa protectora que alberga diversos componentes importantes, como sensores de imagen, lentes especializadas y procesadores de alto rendimiento. Su propósito es capturar y analizar imágenes de manera precisa, emulando la percepción visual humana. También conocidas como çámaras inteligentes", estas unidades emplean algoritmos avanzados y software especializado para interpretar la información visual, lo que permite detectar defectos, identificar objetos, medir dimensiones y reconocer patrones. Gracias a sus componentes diseñados para minimizar distorsiones y mejorar la nitidez, estos sistemas garantizan resultados confiables en entornos industriales y automatizados [36].

La luz se divide en distintas longitudes de onda, y el rango que nuestros ojos pueden captar se denomina espectro visible. Dependiendo de su uso, En la Figura 23 se puede apreciar que las cámaras pueden detectar tanto estas longitudes de onda perceptibles como otras que están fuera del alcance de la visión humana [37].

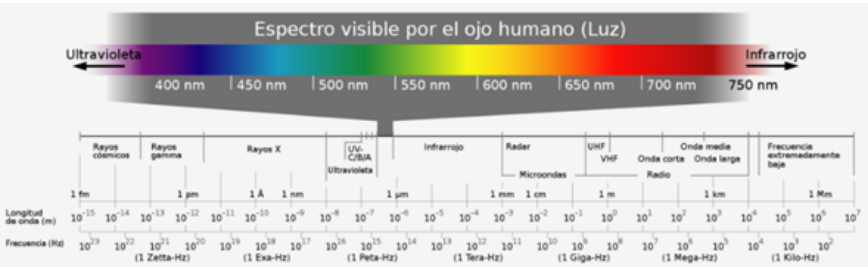


Figura 23: Rango de espectro de luz en cámaras.

Fuente: [37].

En los sistemas de visión artificial, las cámaras más utilizadas trabajan con longitudes de onda que el ojo humano puede percibir. Dentro de este grupo, existen dos categorías principales:

- **Monocromo:** Estás cámaras se enfocan en capturar solo la intensidad de la luz, registrando las variaciones en los niveles de brillo en una imagen, no procesan información cromática, lo que significa que solo producen imágenes en escala de grises. Esto les permite obtener detalles precisos sobre el contraste y la textura de los objetos, ya que cada píxel representa la cantidad de luz que incide sobre él, sin interferencias de colores [38].

- **Color:**A diferencia de las monocromáticas, estas cámaras pueden captar y diferenciar los colores dentro del espectro visible. Utilizan filtros específicos, como el sistema RGB (rojo, verde y azul), para interpretar la información cromática de la escena [38].

Figura 24: Rango de espectro de luz visible para el ojo humano.

Fuente: [37].

De igual manera, existen cámaras que trabajan con longitudes de onda fuera del rango visible. Dentro de este grupo, se pueden identificar dos categorías principales:

- **Termográfica:** Las cámaras termográficas en visión artificial son dispositivos especializados que tienen la capacidad de detectar la radiación infrarroja (LWIR) que emiten los objetos debido a su temperatura. Esta radiación, invisible para el ojo humano, se convierte en una imagen visible que muestra las variaciones térmicas de la superficie de los objetos. Gracias a esto, las cámaras pueden medir y visualizar la temperatura de los objetos en tiempo real [38].

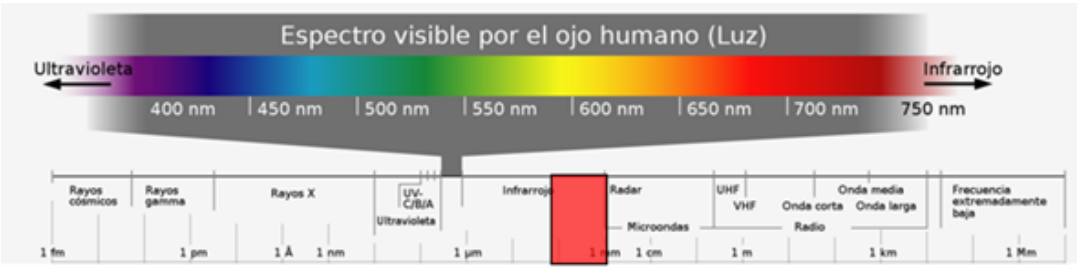


Figura 25: Rango de espectro de luz termográfico.

Fuente: [37].

- **Hiperespectrales:**Las cámaras hiperespectrales son dispositivos avanzados que capturan información más allá de lo que podemos ver con nuestros ojos. Estas cámaras descomponen la luz en un gran número de segmentos estrechos de diferentes longitudes de onda, lo que les permite identificar detalles que otras cámaras no pueden detectar. Al capturar múltiples bandas espectrales, estas cámaras pueden analizar cómo interactúa la luz con los materiales, lo que les permite reconocer características únicas de cada uno. Esto hace posible identificar con precisión objetos que, a simple vista, parecen iguales, todo sin necesidad de tocarlos ni dañarlos [39]. En la Figura 26 se muestra una representación gráfica del rango de espectro de luz hiperespectrales

Figura 26: Rango de espectro de luz hiperespectrales.

Fuente: [37].

Software

El software de imágenes en visión artificial es fundamental para procesar y analizar las capturas realizadas por la cámara. Mientras la cámara se encarga de obtener y enviar las imágenes a la computadora principal, el software gestiona la comunicación con el hardware, controla las funciones de la cámara, recibe y procesa los datos visuales, toma decisiones basadas en los resultados obtenidos y se encarga de la integración con otros dispositivos o sistemas que requieran esta información [35]. En este campo, el software de imágenes se divide en tres categorías principales:

- **Visor de cámara:** El visor de cámara representa la parte más intuitiva y fácil de manejar dentro del software de imágenes, lo que permite iniciar su uso sin complicaciones. No solo ayuda a establecer la conexión con la red y configurar los parámetros de la cámara, sino que también permite capturar, mejorar, visualizar y almacenar imágenes. Además, algunos programas incorporan herramientas avanzadas, entre ellas histogramas y gráficos de líneas, que brindan un análisis detallado de las imágenes procesadas [35].

- **Software Integral:** Un software integral de visión artificial incorpora las funciones importantes de un visor de cámara, también cuenta con herramientas avanzadas para un análisis más riguroso. Entre sus características se incluyen la gestión de proyectos, la configuración de múltiples cámaras y servidores, y el uso de unidades de procesamiento gráfico (GPU) para acelerar el procesamiento de datos. Además, tiene capacidades para la reconstrucción de imágenes en tres dimensiones y la visualización de objetos en 3D, lo que es fundamental en aplicaciones que requieren precisión y análisis espacial [35].

- **Kit de desarrollo de software (SDK):** Es un conjunto de herramientas que permite crear aplicaciones, estos kits incluyen controladores compatibles con

distintas interfaces de cámara, compatibilidad con capturadores de fotogramas si el sistema los requiere y una API especializada en procesamiento de imágenes industriales. Además, suelen ofrecer compatibilidad con varios lenguajes de programación, como C y C++ por su capacidad para manejar el hardware de manera directa, Python por su facilidad de uso y bibliotecas como OpenCV, C en entornos industriales, Java para aplicaciones multiplataforma, Matlab en análisis científico y LabVIEW para integración con hardware industrial. También incorporan ejemplos de código, documentación detallada y herramientas de configuración que permiten ajustar parámetros de la cámara y visualizar los efectos en un tiempo definido [40].

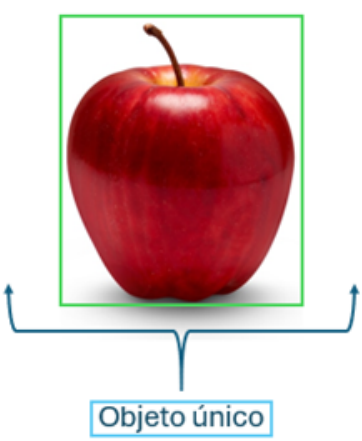
1.4.4.2 Funciones principales en visión artificial

Como ya se ha mencionado en el campo de la visión artificial, los sistemas computacionales realizan diversos procesos que les permiten interpretar y analizar imágenes de manera correcta. Estos procesos son de suma importancia en este sistema para que las máquinas puedan identificar objetos, reconocer patrones y comprender el entorno visual. A continuación, se describirán algunas de las tareas que pueden realizar estos sistemas.

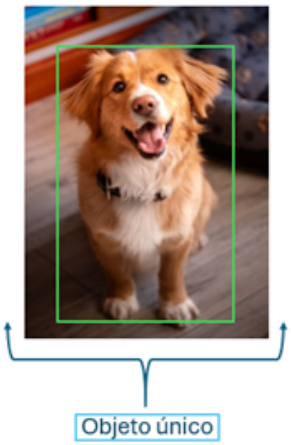
1. **Clasificación de imágenes**

   La clasificación de imágenes consiste en agrupar y etiquetar imágenes o regiones específicas dentro de ellas según características predefinidas. Dependiendo del contexto, las etiquetas pueden asignarse basándose en uno o varios factores [41]. Existen dos tipos principales de clasificación de imágenes:

   - **Clasificación de etiqueta única:** En este tipo, cada imagen recibe solo una etiqueta. El modelo analiza la imagen y la clasifica bajo un único criterio [41]. Por ejemplo, se puede entrenar un modelo para identificar fotos de frutas y asignar la etiqueta (Figura 27, A) o animales (Figura 27, B), sin permitir que una imagen reciba más de una etiqueta



(A) Etiqueta única de una fruta



(B) Etiqueta única de un animal

Figura 27: Comparación de resultados en distintas condiciones. [Autoría propia]

- **Clasificación multietiqueta** A diferencia del caso anterior, las imágenes pueden recibir múltiples etiquetas simultáneamente. Este enfoque es útil cuando una imagen puede pertenecer a varias categorías al mismo tiempo [41]. Un ejemplo sería la clasificación de fotos de paisajes naturales, donde una misma imagen puede ser etiquetada como "montañas", "bosquez río"debido a que muestra todas esas características en un solo escenario como se muestra en la Figura 28.

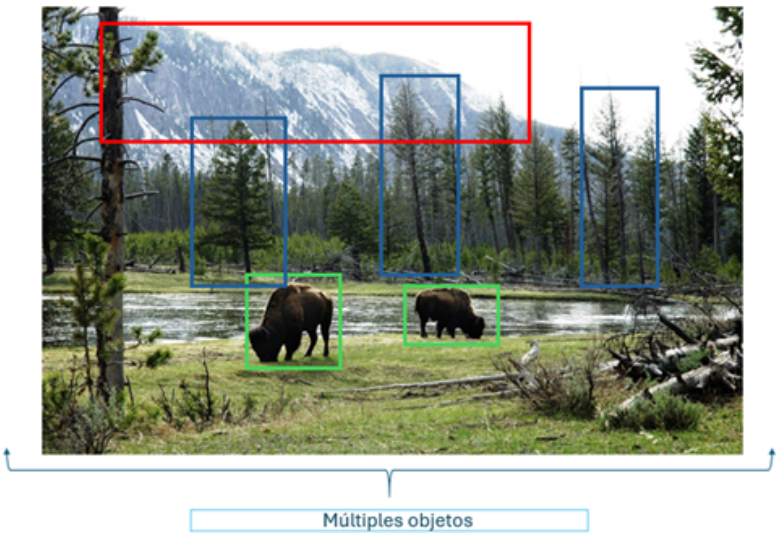


Figura 28: Etiqueta múltiple de un paisaje. [Autoría propia]

2. Detección y seguimiento de objetos

La detección y el seguimiento de objetos son procesos fundamentales en visión artificial que permiten no solo identificar y localizar elementos dentro de una imagen o video, sino también monitorear su movimiento a lo largo del tiempo. Mientras que la detección analiza cada cuadro de manera independiente para encontrar y delimitar los objetos presentes en una escena, el seguimiento permite asociar esos mismos objetos en secuencias consecutivas, manteniendo su identidad incluso ante cambios de posición, iluminación o perspectiva. Para lograrlo, se emplean técnicas avanzadas que combinan modelos de clasificación con algoritmos de predicción de movimiento [42].

3. Segmentación

Las técnicas tradicionales de segmentación de imágenes funcionan analizando los colores de los píxeles y otras características como el brillo, el contraste y la intensidad para distinguir y separar diferentes elementos dentro de una imagen. Una de sus ventajas es que pueden configurarse y entrenarse de manera rápida con algoritmos simples de aprendizaje automático, lo que las hace muy útiles para tareas como la clasificación semántica. Aunque los métodos más avanzados basados en aprendizaje profundo permiten un análisis más detallado y preciso, especialmente en casos complejos como la segmentación panóptica, que requiere entender mejor el contexto de la imagen, las técnicas clásicas siguen siendo una

gran alternativa. A continuación, se describen las técnicas más utilizadas a lo largo del tiempo [43].

- **Umbrales:** Los métodos de umbral se emplean para convertir imágenes a una forma binaria, clasificando los píxeles según si sus niveles de intensidad superan o no un valor umbral determinado. Este valor puede calcularse utilizando técnicas como el método de Otsu, que optimiza el umbral para maximizar la diferencia entre las clases de la imagen [43]. Aunque es una técnica sencilla, depende de la distribución de los valores de intensidad en la imagen. Es aplicable en donde los objetos tienen un contraste claro respecto al fondo. Sin embargo, puede ser poco eficiente en imágenes con variaciones de iluminación o ruido, lo que requiere enfoques más refinados o preprocesamiento adicional. En la Figura 29 se presenta una segmentación utilizando la técnica de umbral.

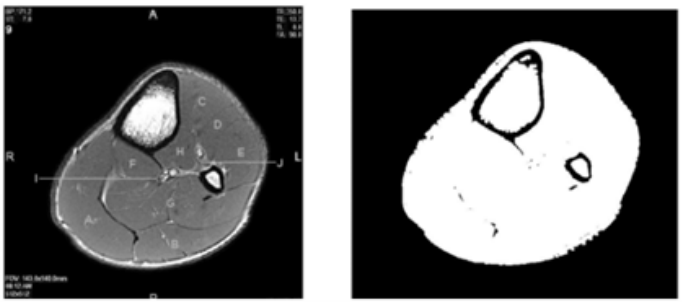


Figura 29: Segmentación umbral.

Fuente: [44].

- **Histogramas:** Los histogramas se utilizan para visualizar la frecuencia de diferentes valores de píxel dentro de una imagen. Esta técnica es útil para determinar los límites de intensidad que separan diferentes áreas de la imagen, permitiendo la segmentación a través de un análisis detallado de la distribución de intensidades. Los histogramas ayudan a definir de manera eficiente los umbrales para segmentar el fondo de los objetos, especialmente cuando la variación en los valores de píxel es limitada [43]. En la Figura 30 se presenta una segmentación utilizando la técnica por histograma.

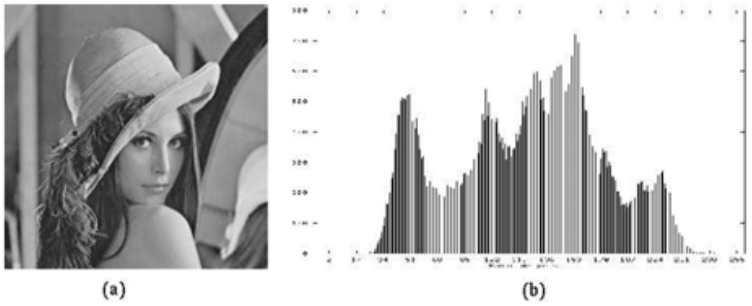


Figura 30: Segmentación por histograma.

Fuente: [45].

- **Detección de bordes:** La detección de bordes es una de las técnicas más utilizadas en visión artificial que permite identificar las zonas donde se producen transiciones significativas en la intensidad de píxeles, lo que indica el

límite entre objetos o regiones. Los algoritmos de detección de bordes, como el operador Sobel o el de Canny, buscan puntos donde el contraste cambia bruscamente, facilitando la localización precisa de objetos en una imagen [46]. Esta técnica es importante para tareas que requieren la identificación de formas o la definición clara de límites, y suele complementarse con técnicas de suavizado para reducir el ruido. Aunque eficaz, puede verse limitada por el ruido en las imágenes, por lo que es importante aplicar filtros previos. En la Figura 31 se presenta una segmentación utilizando la técnica por detección de bordes.

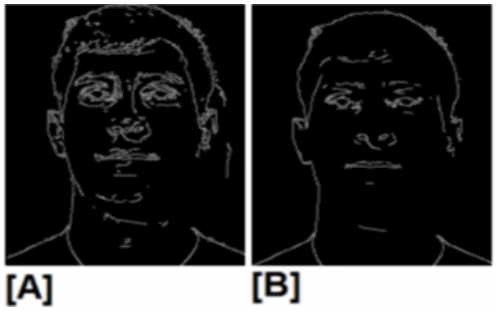


Figura 31: Segmentación por detección de bordes.

Fuente: [47].

- **Cuencas hidrográficas:** Los algoritmos de cuencas hidrográficas convierten la imagen en escala de grises, luego la procesan como un mapa topográfico en el que el brillo de cada píxel actúa como elevación. A partir de esta representación, el algoritmo puede identificar valles y crestas que corresponden a regiones de bajo y alto brillo, respectivamente, lo que permite segmentar áreas de interés. Este enfoque es útil para entender cómo se distribuyen las intensidades de los píxeles en una imagen, especialmente cuando se busca identificar zonas homogéneas o patrones topográficos. Sin embargo, su complejidad computacional puede ser un reto en imágenes grandes o en tiempo real, lo que limita su aplicabilidad en algunos contextos [43]. En la Figura 32 se presenta una segmentación utilizando la técnica por cuencas hidrográficas.

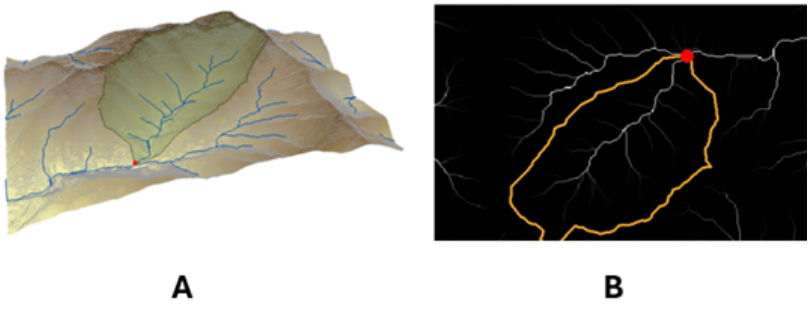


Figura 32: Segmentación por cuencas hidrográficas.

Fuente: [48].

- **Segmentación basada en regiones:** La segmentación basada en regiones se enfoca en dividir una imagen en áreas homogéneas, agrupando píxeles que comparten características visuales similares, como color, textura e intensidad.

Para lograrlo, se emplean diversos algoritmos de agrupación, entre los que destacan K-medias, C-medias difusas y agrupación jerárquica, los cuales permiten organizar los píxeles en segmentos que reflejan las propiedades comunes de los mismos. Esta técnica es particularmente útil en imágenes con variaciones en iluminación o contraste, ya que tiene la capacidad de adaptarse a estas diferencias y generar regiones que sean más representativas de la imagen original. Sin embargo, un desafío común es la segmentación excesiva, que produce demasiados segmentos pequeños, o la segmentación insuficiente, que puede dar como resultado regiones demasiado amplias, lo que puede restar detalles importantes en la imagen [49].

- **Segmentación basada en clústeres:** La segmentación por agrupamiento, como la técnica de K-medias, divide una imagen en grupos o clústeres de píxeles con características similares, utilizando un algoritmo no supervisado. El proceso comienza seleccionando centroides aleatorios y asignando cada píxel al centroide más cercano, luego se recalculan los centroides en función de los píxeles asignados. Este proceso se repite hasta que los centroides ya no cambian significativamente. Este método es eficaz para segmentar imágenes donde los objetos presentan características similares entre sí, pero puede tener limitaciones en cuanto a la elección del número de clústeres y la inicialización de los centroides. A pesar de estos desafíos, es ampliamente utilizado por su capacidad para segmentar grandes volúmenes de datos visuales sin necesidad de un etiquetado previo [50]. En la Figura 33 se presenta una segmentación utilizando la técnica basada en clústeres.

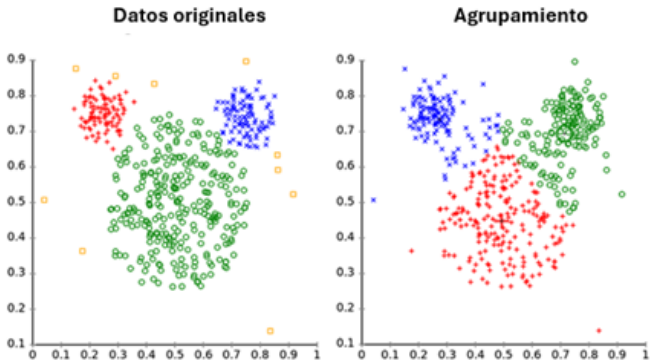


Figura 33: Segmentación basada en clústeres.

Fuente: [51].

1.4.5. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (CNN, por sus siglas en inglés), son modelos de inteligencia artificial diseñados para analizar datos organizados en forma de matriz, como ocurre con las imágenes. Su funcionamiento se basa en aplicar filtros que permiten identificar automáticamente rasgos importantes dentro de una imagen, como bordes, texturas o figuras, sin necesidad de intervención manual. Esta capacidad de reconocimiento progresivo se asemeja al procesamiento visual del cerebro humano. Por ello, las CNN se han convertido en una herramienta fundamental en

tareas visuales complejas como la clasificación y detección de objetos, impulsando grandes avances en el campo de la visión por computadora[52].

1.4.5.1 Funcionamiento de la red neuronal convolucional

La idea principal es que una CNN aprende progresivamente a identificar elementos dentro de una imagen, empezando por los detalles más simples y terminando con estructuras completas. Este aprendizaje ocurre gracias a muchas imágenes de ejemplo, lo que permite que la red generalice y reconozca un objeto en distintas condiciones visuales [52].

Para que una red neuronal convolucional pueda identificar correctamente distintos objetos, es necesario entrenarla con miles de imágenes que representen una amplia variedad de casos. Por ejemplo, para reconocer bicicletas, debe analizar imágenes con distintos modelos, colores, posiciones y entornos; en el caso de las flores, debe ser capaz de identificar una rosa sin importar si está en un jardín, en un florero o incluso si es un dibujo de la misma manera funciona para detectar automóviles, debe reconocer un auto ya sea de frente, desde arriba, al costado y diferentes ángulos incluso si el auto está con diferentes tonos, o sombras [52].

Pixeles y neuronas

Una red neuronal convolucional recibe como entrada los pixeles de una imagen, y cómo el tamaño de la imagen y la cantidad de canales de color afectan el número total de neuronas de entrada. En la Figura 34 se muestra una imagen en escala de grises de 32×32 píxeles, cada píxel representa una sola intensidad de color, por lo que cada píxel se asocia con una neurona. En imágenes a color, se utilizan tres canales (rojo, verde y azul), lo que multiplica por tres la cantidad de datos de entrada en este caso se multiplica 32×32×3 que es igual a 3.072 neuronas de entradas. Este concepto es fundamental para entender cómo las CNN procesan la información visual desde el nivel más básico: los valores numéricos de cada píxel [52].

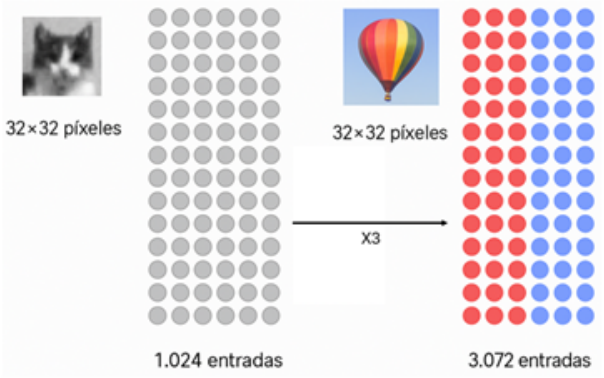


Figura 34: Pixeles de entrada.
[Autoría propia]

En las redes CNN, los valores de los píxeles, que originalmente van de 0 a 255 en imágenes de 8 bits por canal, suelen ser normalizados al rango de 0 a 1 antes de ser introducidos a la red, lo cual se hace por razones técnicas y de rendimiento [52]. En la Figura 35 se muestra una imagen que pasó a la normalización de 0.

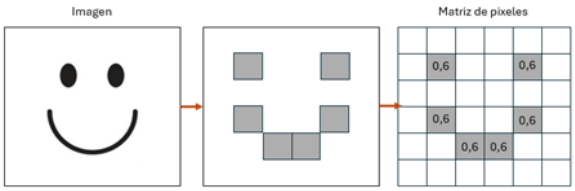


Figura 35: Matriz de pixeles normalizada en 0.
[Autoría propia]

Sin embargo, si se trabajara con imágenes a color, asignar un valor de 1 a cualquiera de los canales RGB (Rojo, Verde o Azul) indicaría la máxima intensidad de ese color específico [52], como se muestra en la Figura 36.

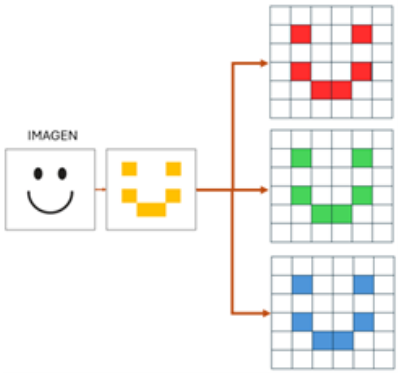


Figura 36: Matriz de pixeles normalizada en 1. [Autoría propia]

Convolución

En esta primera etapa de convolución se toma la imagen de entrada y se le aplican varios filtros para generar mapas de características iniciales. Estos mapas capturan patrones visuales básicos como líneas, bordes o curvas, que son consideradas características primitivas. Luego, se aplica un proceso de subsampling, como el Max-Pooling, para reducir el tamaño de estos mapas conservando la información más relevante. Esta etapa marca el primer nivel de análisis visual, y a medida que se agregan más capas de convolución, la red podrá identificar formas más complejas y representaciones más abstractas de la imagen [52].

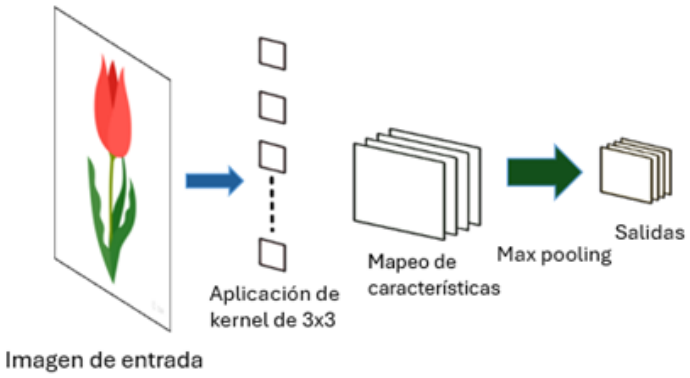


Figura 37: Estructura de las convoluciones. [Autoría propia]

En esta etapa se inicia el paso clave de las redes neuronales convolucionales: la operación de convolución. Esta técnica consiste en aplicar un pequeño filtro o "kernel"sobre grupos de píxeles cercanos en la imagen original. El kernel, por ejemplo, de tamaño 3×3, se desplaza por toda la imagen fila por fila y columna por columna, realizando un producto escalar en cada posición. El resultado de estas operaciones forma una nueva matriz, que se convierte en una capa de neuronas ocultas. Si la imagen es a color, el proceso se extiende a los tres canales (Rojo, Verde y Azul), por lo que se utilizan tres filtros 3×3, cuyos resultados se combinan y se les añade un término de sesgo (bias) para generar una única salida [52].

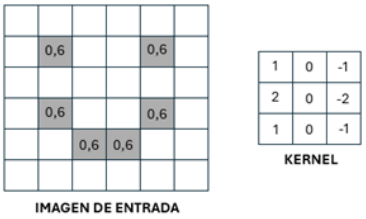


Figura 38: Aplicación de filtro o kernel.
[Autoría propia]

El kernel se inicializa con valores aleatorios, aunque a veces se utilizan distribuciones normales que respetan ciertas simetrías para mejorar el rendimiento inicial. A medida que la red aprende, estos valores se van ajustando automáticamente mediante el proceso de retropropagación (backpropagation), lo que permite que el kernel se especialice en detectar características específicas de las imágenes. Como ejemplo, se realiza la operación utilizando un kernel de 3×3, el cual se coloca sobre los primeros 3×3 píxeles de la imagen de entrada. Dado que el kernel es más pequeño que la imagen completa, se desplaza progresivamente hacia la derecha y hacia abajo. En cada posición, se multiplica cada valor del kernel por su píxel correspondiente en la imagen, y luego se suman todos esos productos. El resultado de esta suma se almacena en una nueva matriz, que constituye la imagen de salida después del filtrado [53].

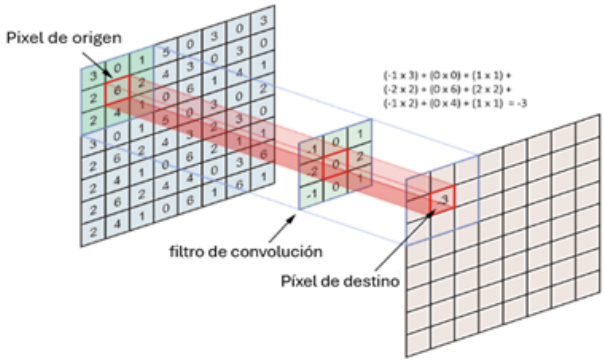


Figura 39: Aplicación de filtro o kernel.
Fuente: [53]

Ahora observamos una imagen en escala de grises de 32×32 píxeles, donde cada píxel representa una intensidad y, por lo tanto, se asocia a una neurona de entrada, dando un total de 1.024 neuronas. Pero si trabajamos con una imagen a color,

debemos considerar tres canales RBG, lo que eleva la entrada a 32×32×3, es decir, 3.072 neuronas. Este gran volumen de datos permite a la red analizar la información visual desde sus niveles más básicos. Sin embargo, al avanzar en la red y aplicar convoluciones, la cantidad de neuronas puede crecer rápidamente. Para evitar este crecimiento excesivo, aplicamos un proceso de reducción llamado subsampling, siendo el más utilizado el Max-Pooling. Este método reduce el tamaño de las representaciones intermedias conservando las características más relevantes detectadas por los filtros, lo que ayuda a disminuir la complejidad computacional sin perder información clave [52].

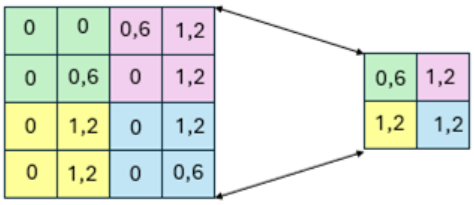


Figura 40: Aplicación de subsamplingl con Max-Pooling .
[Autoría propia]

El Max-pooling de tamaño 2×2 se aplica después de una capa convolucional para reducir el tamaño de las imágenes de características sin perder la información más relevante [52]. Por ejemplo, si tras la convolución obtenemos una imagen de características de 4×4 píxeles con los valores:

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 5 & 6 & 1 & 2 \\ 4 & 3 & 9 & 8 \\ 7 & 2 & 5 & 6 \end{bmatrix} \tag{3}$$

Aplicando Max-pooling de 2×2 con paso 2, dividimos la imagen en bloques de 2×2 y seleccionamos el valor máximo de cada uno. Así, de los bloques se obtienen los valores máximos: 6, 2, 7 y 9, reduciendo la imagen a 2×2: 6,2;7,96, 2; 7, 96,2;7,9. Este proceso disminuye la cantidad de datos a procesar, hace más eficiente la red y permite que se mantenga la información más representativa de cada región analizada, ayudando a enfocarse en las características esenciales de la imagen [52].

En una red neuronal convolucional, el proceso de reducción de una imagen no ocurre una sola vez, sino que se repite a lo largo de varias capas. Primero, se aplica una convolución con distintos kernels que extraen características específicas como bordes, texturas o formas simples. Luego, se realiza un Max-pooling por ejemplo, de tamaño 2×2 que reduce las dimensiones de la imagen y conserva la información más relevante. Esta salida reducida pasa a una nueva etapa donde se repite el mismo proceso: nuevas convoluciones con más kernels, seguidas de otro Max-pooling, y así sucesivamente. Con cada iteración, la imagen se hace más pequeña, pero la red va capturando representaciones más complejas y abstractas de la información. Este encadenamiento de convoluciones y subsampling permite que la red aprenda jerárquicamente, desde características simples hasta patrones de alto nivel, lo cual es esencial para tareas como clasificación, reconocimiento o segmentación de objetos en imágenes [52].

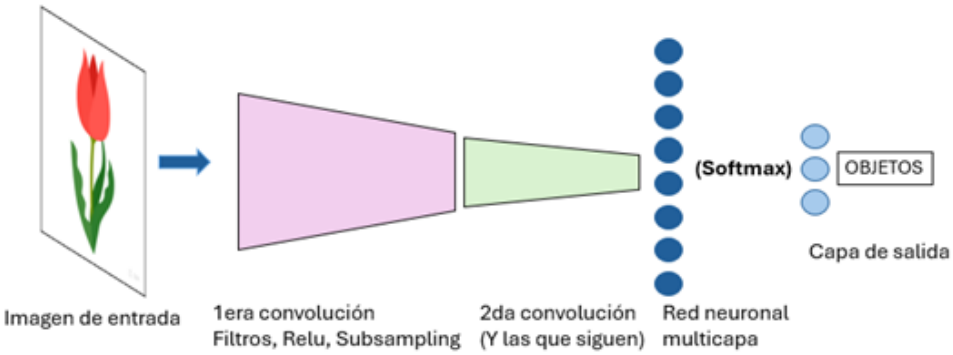


Figura 41: Arquitectura de una red neuronal.
[Autoría propia]

1.4.5.2 Red neuronal Yolo

YOLO (You Only Look Once) es un modelo para la detección de objetos en imágenes en tiempo real, creado por Joseph Redmon y Ali Farhadi en 2015. A diferencia de otros enfoques más complejos, como R-CNN, YOLO realiza todo el proceso de detección en una sola etapa mediante una red neuronal convolucional (CNN), lo que le permite identificar objetos de forma rápida y precisa. El método consiste en dividir la imagen en una cuadrícula, y cada celda de esa cuadrícula se encarga de predecir si hay un objeto, sus coordenadas dentro de la imagen y su categoría. En sus primeras versiones, el modelo fue desarrollado utilizando el framework Darknet, optimizado para velocidad y rendimiento. Esta arquitectura integral y directa le otorga una ventaja significativa en aplicaciones que requieren respuestas en tiempo real [54].

Funcionamiento

El modelo utiliza una red neuronal convolucional que divide la imagen en una cuadrícula, donde cada celda es responsable de detectar objetos que caen dentro de su área. Cada celda predice varios cuadros delimitadores junto con una puntuación de confianza que indica la probabilidad de que haya un objeto y qué tan precisa es la predicción, basada en la intersección sobre unión (IOU). Las coordenadas del cuadro delimitador se expresan en relación con la celda para el centro (x, y), mientras que el ancho y la altura se normalizan respecto al tamaño total de la imagen, permitiendo así una detección precisa y localizada de los objetos en la imagen completa [55].

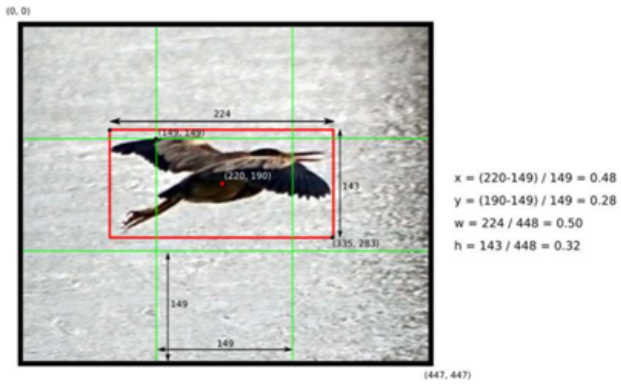


Figura 42: Coordenadas de una imagen.
Fuente: [55]

El modelo YOLO se basa en una red neuronal convolucional, la cual está compuesta principalmente por capas convolucionales para la extracción de características visuales a partir de la imagen de entrada. Aunque su arquitectura está formada mayormente por este tipo de capas, también incluye capas completamente conectadas (densas) en las etapas finales, especialmente en las primeras versiones del modelo, para generar las predicciones sobre las clases y las ubicaciones de los objetos. Esta arquitectura toma como referencia el diseño de GoogleNet, utilizado comúnmente en tareas de clasificación de imágenes. Específicamente, la red original de YOLOv1 estaba formada por 24 capas convolucionales y 2 capas completamente conectadas. Para optimizar el procesamiento y reducir la complejidad del modelo, se implementan convoluciones de tamaño 1x1 que disminuyen la profundidad de los mapas de características, seguidas de convoluciones de 3x3 que permiten captar patrones más complejos. Estas dos configuraciones se combinan de forma alternada a lo largo de la red. La última capa convolucional produce un tensor de forma (7,7,1024), que luego se reduce mediante las capas totalmente conectadas para generar una salida final de tamaño 7x7x30, donde se codifican las predicciones de clase y las coordenadas de los objetos detectados [55].

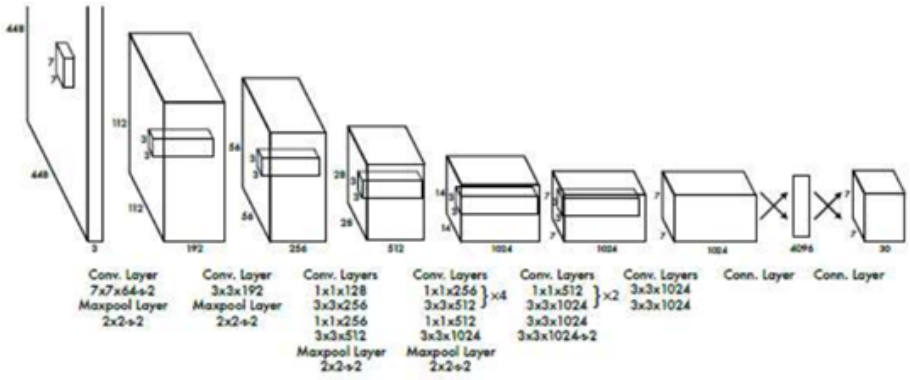


Figura 43: Arquitectura de la red original de YOLO.
Fuente: [55]

Dentro del desarrollo de yolo han surgido múltiples versiones, cada una con mejoras y características específicas que responden a diferentes necesidades en las aplicaciones de detección de objetos. Estas versiones permiten adaptar el modelo a

distintos entornos, desde sistemas avanzados con gran capacidad de cómputo hasta dispositivos con recursos limitados. Una de las versiones más destacadas para este último caso es Yolov4, ya que fue diseñada para ofrecer un alto rendimiento incluso en CPU con bajo poder de procesamiento. Esta versión optimiza la precisión y la velocidad sin necesidad de utilizar GPU de alto costo, lo que la convierte en una opción ideal para sistemas embebidos, cámaras inteligentes, dispositivos IoT o aplicaciones en tiempo real que requieren eficiencia con pocos recursos. Por estas razones, es fundamental analizar más a fondo esta red, que representa una solución potente y accesible dentro del ecosistema de detección inteligente.

YOLOv4

Yolov4 es un modelo avanzado de detección de objetos en tiempo real que mejora el rendimiento de sus versiones anteriores al combinar velocidad y precisión en una sola arquitectura. Su diseño está compuesto por tres partes principales: la columna vertebral, que se encarga de extraer las características importantes de la imagen; el cuello, que refuerza y fusiona esas características para facilitar la detección; y la cabeza, que finalmente realiza la predicción de las clases y las posiciones de los objetos. Estas partes están conectadas mediante capas específicas que permiten una detección óptima, eficiente y robusta [54].

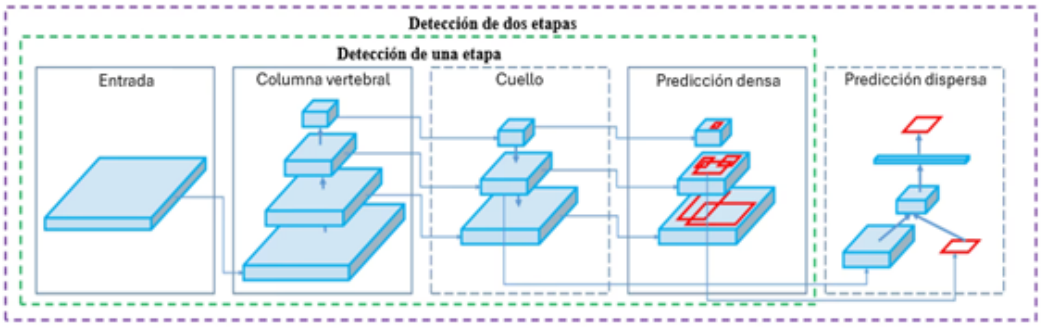


Figura 44: Arquitectura de la red YOLOv4.
Fuente: [54]

Se analizará detalladamente cada una de las estructuras que conforman la arquitectura completa del modelo yolov4, con el fin de comprender con mayor claridad qué ocurre en cada etapa del proceso de detección. Este desglose permitirá entender cómo se extraen las características visuales, cómo se refinan y fusionan a diferentes escalas, y finalmente, cómo se generan las predicciones de las cajas delimitadoras y las clases de los objetos detectados en la imagen.

A. Columna vertebral (Backbone)

En el modelo YOLOv4, la columna vertebral o backbone cumple la función esencial de extraer las características más relevantes de la imagen de entrada, tales como bordes, texturas, formas y patrones, sin identificar aún objetos específicos. Para esta tarea, se emplea comúnmente CSPDarknet53, una versión mejorada del Darknet-53 que incorpora una estructura llamada Cross Stage Partial (CSP), la cual divide y fusiona caminos de información para optimizar el flujo del gradiente durante el entrenamiento, reduciendo la redundancia computacional sin sacrificar precisión. Esta arquitectura también utiliza bloques residuales que permiten el aprendizaje profun-

do más estable y eficiente. Por ejemplo, al ingresar una imagen de un gato en una calle, el backbone no clasifica al gato, pero sí extrae detalles visuales como el contorno de su cuerpo, las formas de las orejas o la textura de su pelaje. Dentro de este proceso de la columna vertebral existen operaciones importantes como las capas convolucionales, la normalización por lotes (Batch Normalization), la activación Leaky ReLU y la estructura CSP [56].

- **Capas convolucionales:** Las capas convolucionales son la base de la red neuronal y se encargan de extraer características visuales importantes de las imágenes, como bordes, formas y texturas. Esto se logra aplicando filtros o kernels que se deslizan sobre la imagen o sobre un mapa de características anterior, realizando operaciones de producto punto entre los valores del filtro y los valores de la imagen. Cada filtro aprende a reconocer un patrón específico y, al pasar por varias capas, la red es capaz de identificar objetos cada vez más complejos [57]. Por ejemplo, en una imagen de un gato, las primeras capas podrían detectar los bordes de sus orejas o bigotes como se muestra en la Figura 45.

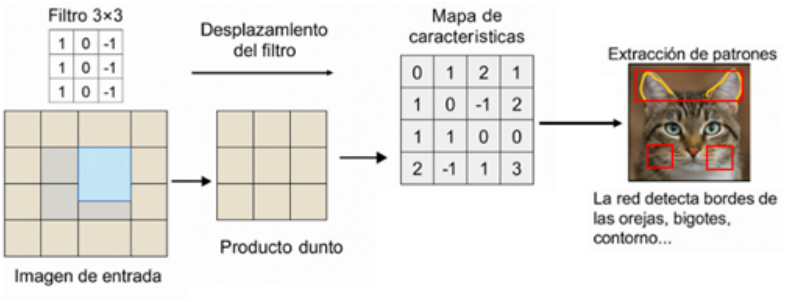


Figura 45: Convolución de la red yolov4. [Autoría propia]

- **Batch normalización:** La normalización por lotes es una técnica aplicada después de las capas convolucionales para estabilizar y acelerar el proceso de entrenamiento. Su objetivo es mantener la salida con una media cercana a cero y una varianza cercana a uno, lo cual ayuda a que la red aprenda de forma más eficiente y evita problemas como gradientes explosivos [58]. Por ejemplo, si la red ha detectado patrones como los ojos de un gato, la BN ajusta esos valores para que las siguientes capas los interpreten de manera uniforme. Además, esta operación incluye parámetros que permiten a la red aprender cómo escalar y desplazar estos valores normalizados durante el entrenamiento.

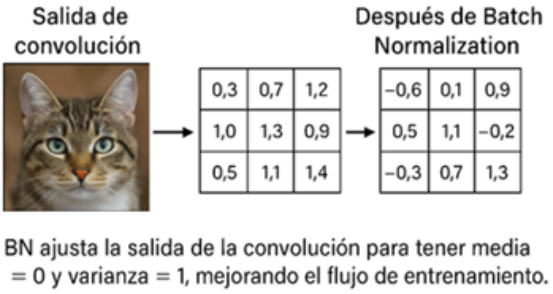


Figura 46: Normalización de la red yolov4. [Autoría propia]

- **Activación Leaky ReLU:** Se aplica para introducir no linealidad en la red, permitiendo que esta aprenda relaciones más complejas entre las características extraídas. A diferencia de la Relu clásica, que elimina completamente los valores negativos, la versión "leaky"deja pasar una pequeña proporción de estos valores, lo que evita que ciertas neuronas se apaguen completamente durante el entrenamiento [59]. Esto es útil, por ejemplo, en la Figura 47 se observa que, cuando algunas características detectadas del gato tienen valores negativos que aún contienen información útil. De este modo, se mantiene un flujo de información más constante en toda la red.

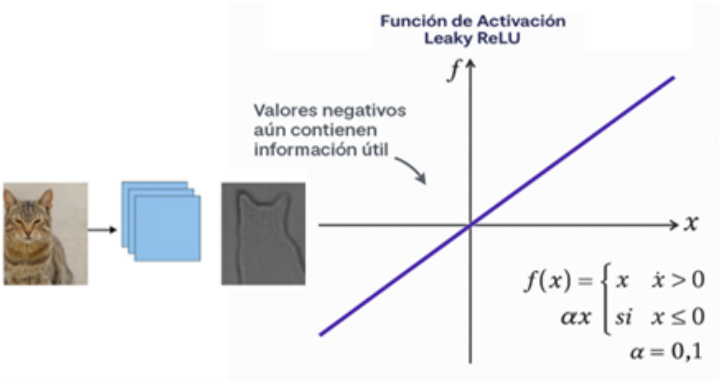


Figura 47: Activación Relu de la red yolov4.
[Autoría propia]

Se aplica como función de activación para transformar la salida lineal de una neurona, permitiendo que los valores positivos pasen sin cambio y los valores negativos pasen atenuados, multiplicados por un pequeño factor ($\alpha = 0{,}1$); esto evita que las neuronas que reciben entradas negativas queden inactivas al producir una salida diferente de cero, lo que mantiene el flujo de gradientes durante el entrenamiento y mejora la capacidad de aprendizaje de la red al evitar que neuronas "mueran"debido a salidas nulas constantes.

- **Estructura CSP:** La arquitectura CSP, implementada en CSPDarknet53, mejora significativamente la eficiencia del modelo al dividir el flujo de datos en dos rutas: una que se procesa con bloques convolucionales y otra que pasa directamente sin modificación. Al final, ambas rutas se fusionan, lo que permite reducir la redundancia de información y mantener un equilibrio entre capacidad de aprendizaje y velocidad de procesamiento [60]. Por ejemplo, En la Figura 48 se muestra que, si se tiene un mapa de características que representa un objeto en este caso un animal y su fondo, CSP puede procesar solo los detalles del gato con mayor profundidad mientras conserva intacta parte de la información global, lo que mejora la detección final sin aumentar excesivamente el costo computacional.

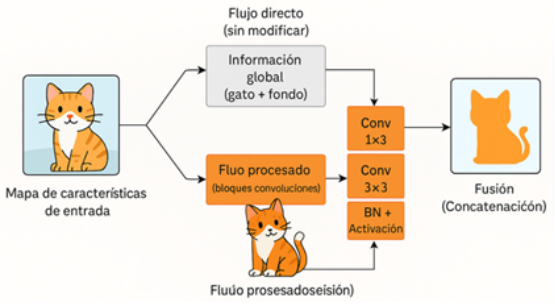


Figura 48: Estructura CSP de la red yolov4. [Autoría propia]

B. Cuello (Neck)

Tiene como función principal reforzar y combinar las características extraídas por la columna vertebral, preparándolas para una detección más precisa en la etapa final. Este componente actúa como un puente entre las capas profundas del modelo (que contienen información rica en semántica) y las capas más superficiales (que conservan detalles espaciales), permitiendo una fusión efectiva entre ambas. De esta manera, se logra que las características relevantes de distintos niveles de abstracción se combinen de forma jerárquica. Dentro de esta sección se aplican dos estructuras importantes como SPP (Spatial Pyramid Pooling), que permite capturar información a diferentes escalas espaciales sin necesidad de redimensionar la entrada, y PANet (Path Aggregation Network), que mejora la propagación de información desde capas inferiores hacia capas superiores para refinar aún más las predicciones [56].

- **Spatial Pyramid Pooling (SPP):** El Spatial Pyramid Pooling (SPP) es una técnica que permite a las redes neuronales procesar imágenes de distintos tamaños sin necesidad de redimensionarlas previamente, al dividir la imagen en varias regiones a diferentes escalas (por ejemplo, 1x1, 2x2, 3x3) y aplicar pooling en cada una para extraer características fijas [60]. En la Figura 49, se muestra que la entrada de una imagen en este caso una manzana, puede ser de cualquier tamaño, el SPP la divide en distintas secciones, extrae lo más relevante de cada una y genera un vector de características de tamaño fijo, lo que ayuda a la red a reconocer la manzana sin importar el tamaño o la forma de la imagen original.

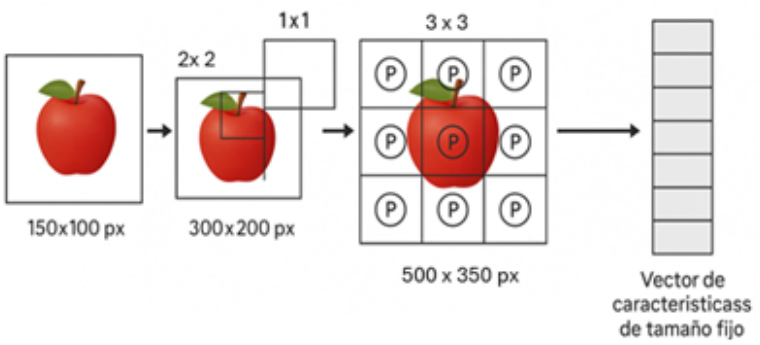


Figura 49: Aplicación SPP de la red yolov4. [Autoría propia]

- **PANet (Path Aggregation Network):** Es una arquitectura diseñada para mejorar la capacidad de una red neuronal para combinar información de diferentes niveles o escalas dentro de una imagen, facilitando una mejor detección y segmentación de objetos. En términos simples, recoge características detalladas tanto de capas profundas como superficiales, fusionándolas eficazmente para que la red tenga una visión más completa y precisa del contenido visual [61]. Por ejemplo, En la Figura 50 se muestra una imagen de entrada que contiene una manzana, PANet procesa la imagen integrando detalles finos, como el contorno y la textura de la piel, junto con características más generales, como la forma y el color. Gracias a esta combinación de información, la red puede identificar correctamente la manzana, incluso si está parcialmente oculta o en un fondo complejo.

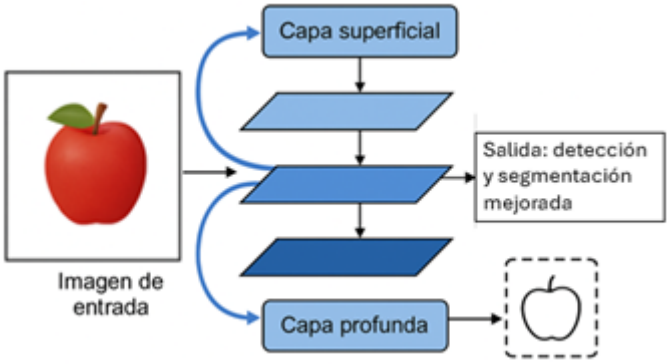


Figura 50: Aplicación Panet de la red yolov4.
[Autoría propia]

Realiza su proceso integrando características extraídas de distintas capas de la red neuronal a través de caminos que van en ambas direcciones: de arriba hacia abajo y de abajo hacia arriba. Primero, recoge las características más generales y abstractas de las capas profundas (que capturan información global, como la forma del objeto), y las combina con las características detalladas y localizadas de las capas superficiales (que capturan texturas y bordes).

C. Cabeza (Dense prediction)

Es la etapa final en una red yolo y se encarga de realizar funciones importantes para la detección de objetos. Esta sección toma la información procesada por las capas anteriores y transforma las características aprendidas en resultados comprensibles al generar, para cada objeto, una predicción de su clase, su ubicación exacta en la imagen mediante una caja delimitadora, y el nivel de confianza asociado a dicha predicción. Para lograrlo, realiza tres procesos en la fase 1, predice las coordenadas de las cajas que indican dónde se encuentra un posible objeto. En la fase 2, estima un valor de confianza que representa la probabilidad de que realmente exista un objeto en esa ubicación. En la fase 3, clasifica el contenido de cada caja según las clases entrenadas, eligiendo la que tenga mayor probabilidad. El resultado de estas operaciones es un tensor con las coordenadas, la puntuación de confianza y las probabilidades por clase, el cual es filtrado en pasos posteriores para mostrar únicamente las detecciones más precisas [56].

- **Predicción de cajas delimitadoras (Bounding boxes):** La predicción de cajas delimitadoras (Bounding Boxes) es el proceso mediante el cual la red determina con precisión dónde se encuentra un objeto dentro de una imagen. Esta predicción no se hace sobre la imagen completa de forma directa, sino que la imagen es dividida en una rejilla (grid), y cada celda de esa rejilla se encarga de predecir si hay un objeto en su área de cobertura. Cada celda no solo intenta detectar objetos, sino que también predice un conjunto de valores que definen la caja: (x, y, w, h) [62].

  En la Figura 51 se muestra una imagen dividida en una rejilla de 6x6, donde cada celda representa una región de la imagen que puede detectar objetos dentro de su área de cobertura. Una de las celdas, seleccionada aleatoriamente, contiene un punto que indica el centro del objeto detectado, definido por las coordenadas relativas (x, y) dentro de esa celda. A partir de ese punto, se dibuja un rectángulo delimitador que puede extenderse fuera de la celda, mostrando visualmente el tamaño estimado del objeto mediante los valores de ancho (w) y altura (h) en relación con el tamaño completo de la imagen.

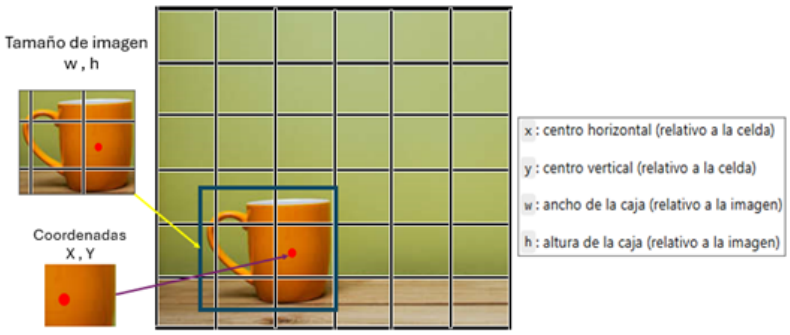


Figura 51: Predicción de cuadros delimitador de la red yolov4.
[Autoría propia]

- **Estimación de confianza:** La estimación de confianza, también conocida como objectness score, es un proceso donde la red evalúa cuán probable es que una caja delimitadora propuesta contenga realmente un objeto, sin importar de qué clase sea. Este valor no indica qué tipo de objeto es, sino simplemente si hay algo presente que vale la pena analizar. Para calcularlo, la red utiliza las características extraídas en cada celda de la imagen y, mediante funciones de activación como la sigmoide, produce un número entre 0 y 1 que representa esa certeza [63].

  En la Figura 52 se muestra una imagen en la que hay varios objetos distribuidos sobre una escena, y se observa cómo el modelo genera múltiples cajas delimitadoras (bounding boxes) alrededor de cada uno de ellos. Cada una de estas cajas está asociada a un valor numérico que representa la estimación de confianza, es decir, la probabilidad de que efectivamente haya un objeto dentro de esa región. Entre todas las predicciones, se destaca una barra roja que indica la caja con mayor nivel de confianza, lo que sugiere que ese objeto es el que más probablemente ha sido detectado correctamente. Por ello, el sistema

selecciona esta caja como el objeto legítimamente reconocido, descartando las demás predicciones de menor confianza.

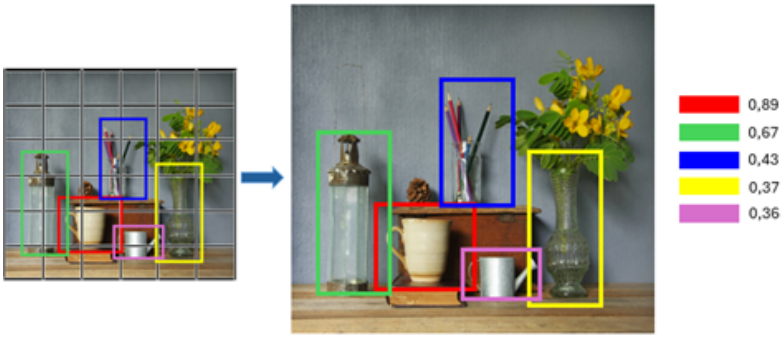


Figura 52: Nivel de confianza de un objeto. [Autoría propia]

- **Clasificación y detección del objeto:** Una vez que la red ha localizado un posible objeto dentro de una caja delimitadora y ha calculado su confianza de que hay algo, pasa a la etapa de clasificación, en la que determina qué tipo de objeto es (por ejemplo: perro, bicicleta, semáforo, etc.). Para lograrlo, la red utiliza las características profundas que ha extraído a lo largo de múltiples capas convolucionales, donde se han detectado patrones como bordes, texturas, formas, colores y relaciones espaciales, los cuales son transformados en vectores numéricos [63].

  Cada celda de la rejilla que analiza la imagen puede generar múltiples cajas delimitadoras, y para cada una de ellas la red proporciona sus coordenadas, un valor de objectness score y un vector de probabilidades para cada clase posible (por ejemplo, 80 clases en el conjunto COCO). Estos vectores son procesados mediante funciones de activación como la softmax, que convierte los valores en probabilidades normalizadas entre 0 y 1, asignando a la caja la clase con mayor probabilidad. Por ejemplo, si una celda detecta un objeto con un objectness score de 0.92, y la distribución de clases después de softmax es: automóvil 0.05, persona 0.83 y semáforo 0.00, entonces el modelo concluye que el objeto es una persona.

Figura 53: Objeto detectado (Persona).
[Autoría propia]

Para obtener la confianza final de la detección, se multiplica el *objectness score* por la probabilidad de la clase elegida: $0{,}92 \times 0{,}83 = 0{,}7636$, lo que indica un 76.36 % de certeza de que esa caja efectivamente contiene una persona.

1.4.5.3 Métricas de detección de evaluación de objetos

Existen diversas métricas ampliamente empleadas para medir cuán efectivo es un modelo de detección de objetos, ya que permiten cuantificar la precisión con la que identifica y localiza correctamente los elementos dentro de una imagen. Estas métricas ayudan a comparar el rendimiento entre distintos modelos, analizar sus puntos fuertes y débiles, y optimizar su comportamiento en función del contexto o aplicación específica.

IOU (Intersección sobre unión)

La métrica de Intersección sobre Unión evalúa qué tan bien coincide una predicción con la realidad al comparar dos cuadros delimitadores: el que predice el modelo y el que corresponde al objeto real. Esta comparación se basa en cuánto se superponen ambos cuadros, dividiendo el área común entre ellos por el total del área que abarcan juntos. Mientras mayor sea el valor de IoU, mejor es la precisión espacial del modelo, ya que indica que el cuadro predicho se ajusta de manera más exacta al objeto real [64].

$$\text{IoU} = \frac{\text{Área de intersección}}{\text{Área de unión}} \tag{4}$$

Por ejemplo, si el cuadro real tiene un área de 50 unidades cuadradas, el cuadro predicho tiene 60, y la zona donde se superponen mide 30, entonces la unión sería 50 + 60 - 30 = 80, y el IoU se obtiene como 30 dividido entre 80, dando un valor de 0.375. Este resultado indica que hay una coincidencia parcial entre la predicción y la realidad, y mientras mayor sea este valor, mejor será el ajuste entre ambos cuadros.

Precisión

Para calcular la precisión respecto al tiempo en la detección de un solo objeto, se analiza el comportamiento del sistema durante un periodo determinado, considerando cuántas veces detectó correctamente el objeto y cuántas veces lo detectó erróneamente. La precisión se obtiene dividiendo el número de verdaderos positivos (TP), que son las veces que el sistema detectó correctamente la presencia del objeto cuando realmente estaba, entre la suma de verdaderos positivos y falsos positivos (FP), donde los falsos positivos representan las veces que el sistema indicó que el objeto estaba presente cuando en realidad no lo estaba. La ecuación para calcular la precisión es:

$$\text{Precisión} = \frac{\text{Número de detecciones correctas (TP)}}{\text{Número total de detecciones (TP + FP)}} \tag{5}$$

Por ejemplo, si en un intervalo de 10 segundos el modelo detectó correctamente el objeto 8 veces (TP = 8) y lo detectó por error 2 veces (FP = 2), la precisión sería

$8/8 + 2 = 0.8$, lo que da como resultado un 80 % de precisión durante ese tiempo.

Recall

Para calcular el recall con respecto al tiempo, se analiza cuántas veces el sistema logró detectar correctamente un objeto en comparación con la cantidad total de veces que ese objeto realmente estuvo presente durante un periodo determinado. El recall se obtiene dividiendo el número de verdaderos positivos (TP), que son las detecciones correctas, entre la suma de verdaderos positivos más falsos negativos (FN), donde los falsos negativos representan las veces que el objeto estuvo presente pero no fue detectado.

$$\text{Recall} = \frac{\text{Verdaderos Positivos (TP)}}{\text{Verdaderos Positivos (TP)} + \text{Falsos Negativos (FN)}} \tag{6}$$

Por ejemplo, si en un lapso de 12 segundos un objeto apareció 10 veces y el sistema lo detectó correctamente en 7 ocasiones, fallando en 3, el *recall* sería $\frac{7}{10} = 0{,}7$, lo que representa un 70 %, indicando que el sistema fue capaz de identificar el objeto el 70 % de las veces que realmente estuvo presente.

1.4.6. Lenguajes de programación para cuadricópteros

Los cuadricópteros operan mediante instrucciones programadas que controlan su vuelo, procesan datos de sensores y ejecutan tareas de manera autónoma o remota. Para lograr esto, se utilizan lenguajes de programación que determinan su funcionamiento, optimizan su rendimiento y facilitan su integración con otros sistemas. Según el objetivo del dron, pueden emplearse lenguajes de bajo nivel, que brindan un control preciso sobre el hardware, o de alto nivel, que agilizan el desarrollo y mejoran la escalabilidad. La elección del lenguaje de programación impacta directamente en factores como el desempeño, la flexibilidad y la facilidad de implementación [65]. Al decidir cuál utilizar, es importante considerar los siguientes aspectos:

- **Facilidad de uso:** Algunos lenguajes son intuitivos y permiten desarrollar prototipos rápidamente, lo que resulta ideal para principiantes.
- **Eficiencia y velocidad:** En aplicaciones que requieren procesamiento en tiempo real y respuestas inmediatas, es fundamental emplear lenguajes optimizados para alto rendimiento.
- **Compatibilidad e integración:** La capacidad de conectar el dron con sensores, cámaras y otros periféricos es esencial para ampliar sus funciones.
- **Comunidad y soporte:** Un ecosistema de desarrolladores activo y bibliotecas bien documentadas pueden acelerar el desarrollo y la resolución de problemas.

Teniendo en cuenta estos aspectos, exploremos los principales lenguajes utilizados en la programación de drones.

Python

Es uno de los lenguajes de programación más utilizado a nivel mundial, una excelente opción para la programación de cuadricópteros, especialmente para quienes buscan desarrollar modelos de manera rápida y sencilla. Su sintaxis clara y fácil de comprender permite que tanto principiantes como programadores experimentados puedan utilizarlo sin complicaciones. Además, cuenta con una gran variedad de herramientas y recursos que facilitan su integración con los sistemas de control de vuelo, el procesamiento de datos y la incorporación de sensores [65].

Entre sus principales ventajas, Python destaca por su facilidad de aprendizaje, apoyada por una amplia cantidad de documentación y tutoriales en línea. También dispone de numerosas bibliotecas, como DroneKit y PyDrone, que ayudan en tareas relacionadas con el manejo de comandos de vuelo y la gestión de información proveniente de los sensores. Otra de sus cualidades es su compatibilidad con múltiples plataformas, lo que permite desarrollar aplicaciones que funcionan en distintos sistemas y son capaces de interactuar con una variedad de hardware para drones [65].

Figura 54: Logo Python.

Fuente: [66].

C++ con gran velocidad de procesamiento

En el desarrollo de software para drones, C++ es uno de los lenguajes más empleados debido a su capacidad para administrar los recursos del sistema con precisión. Su uso es fundamental en aplicaciones que requieren procesamiento en tiempo real y un control detallado del hardware, lo que lo convierte en una herramienta clave para garantizar estabilidad y respuesta rápida en tareas críticas. Dentro de sus principales cualidades, C++ permite un manejo preciso de los recursos del sistema, lo que mejora el desempeño de los drones. Su enfoque en la programación orientada a objetos facilita la organización de estructuras complejas y favorece la escalabilidad del software. Además, dispone de una amplia variedad de bibliotecas especializadas en robótica y sistemas de control, como ROS (Robot Operating System) y PX4, que ayudan en la creación de sistemas avanzados de navegación y operación [65].

Figura 55: Logo C++.

Fuente: [67].

Java en diferentes sistemas

Java es un lenguaje de programación que no está restringido a una sola plataforma y

mantiene un equilibrio entre rendimiento y facilidad de uso. Su principio de escribir una vez, ejecutar en cualquier lugar, permite que el mismo código funcione en distintos sistemas sin necesidad de ajustes, lo que lo convierte en una opción frecuente en el desarrollo de aplicaciones para drones [65].

Entre sus características más importantes, Java tiene compatibilidad multiplataforma gracias a la Java Virtual Machine (JVM), lo que facilita su funcionamiento en diferentes dispositivos. También cuenta con muchas bibliotecas y frameworks que ayudan en tareas como la gestión de redes y la creación de interfaces gráficas. Además, su comunidad activa y la extensa documentación disponible proporcionan recursos útiles para solucionar problemas y mejorar el desarrollo de software en proyectos de drones [65].

Figura 56: Logo Java.

Fuente: [68].

1.4.7. Sistema de comunicación

Para que un mensaje sea transmitido, es necesario contar con un sistema de comunicación que facilite el traslado de la información desde su punto de origen, denominado fuente, hasta el destino, a través del espacio y el tiempo. Esto puede hacerse utilizando cables, como en el caso de los teléfonos, o mediante ondas, como ocurre con las radios. Un sistema de comunicación implica la transferencia de información con un propósito claro desde un lugar (emisor, origen, transmisor) hacia otro lugar (destino, receptor). Los mensajes pueden adoptar diversas formas, como secuencias de símbolos, variaciones en la intensidad de luz y colores en una transmisión televisiva, o incluso la presión acústica generada por la voz. Los sistemas de comunicación eléctrica proporcionan los medios necesarios para que la información, convertida en señales, sea enviada o intercambiada. Un sistema de comunicación se compone principalmente de tres elementos fundamentales: el transmisor, el canal de transmisión y el receptor [69].

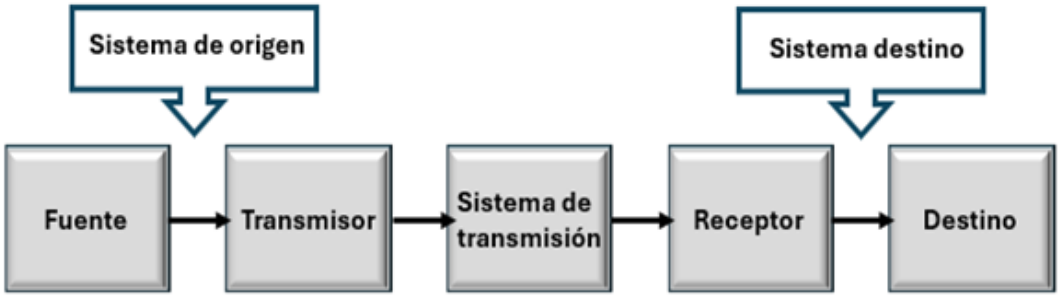


Figura 57: Diagrama general de bloques. [Autoría propia]

A continuación, se detallan los tipos de comunicación más utilizados:

- **Conexión USB:** El USB es una interfaz universal que emplea un sistema de bus en serie, permitiendo la transferencia de datos entre dispositivos externos y una computadora u otros terminales. Este sistema permite la transmisión de datos bidireccional, y el cable USB también tiene la capacidad de suministrar energía a los dispositivos conectados. Este sistema emplea la transmisión de datos de forma secuencial, enviando bits de manera continua a través de un solo canal de datos, lo que optimiza la velocidad de transferencia. A diferencia de las conexiones paralelas, que requieren sincronización entre varias líneas de datos, la transmisión en serie no presenta este desafío. Además, este método de transmisión permite el uso de múltiples líneas de datos simultáneamente, lo que posibilita la transferencia y recepción de datos al mismo tiempo con un solo cable USB. Estándares como Ethernet, HDMI y SATA también utilizan esta forma de transmisión serial [70].

- **Conexión Wifi:** El Wifi es una tecnología que posibilita la conexión sin cables de dispositivos electrónicos a redes de área local (LAN) y, por extensión, a Internet. Funciona mediante la transmisión de datos a través de ondas de radio, estableciendo una comunicación directa entre los dispositivos y un punto de acceso, habitualmente denominado router. Este sistema facilita la movilidad y flexibilidad al permitir que múltiples dispositivos se conecten simultáneamente a la red, sin depender de conexiones físicas, mejorando así la accesibilidad y la experiencia del usuario en entornos domésticos, empresariales y públicos [71].

- **Conexión Bluetooth:** Bluetooth es un estándar de comunicación inalámbrica diseñado para eliminar los cables y facilitar la transferencia de datos entre dispositivos digitales. Su alcance es limitado, por lo que se usa principalmente en conexiones cercanas, como entre teléfonos, tabletas y otros accesorios [72]. En el caso de los drones, se emplea para comunicarse con controles remotos y teléfonos inteligentes, permitiendo configuraciones rápidas y actualizaciones de software sin necesidad de conexiones físicas [73].

1.4.8. Protocolo de comunicación

El sistema de comunicación de un dron es un componente esencial para su desempeño y operación. Permite que los operadores tengan un control preciso sobre sus movimientos, ajustando la altitud, la velocidad y la trayectoria del dron en tiempo real, lo que es crucial para misiones que requieren alta precisión. Además del control básico, estos sistemas facilitan la transmisión de datos vitales, como las coordenadas GPS, la telemetría del dron (que incluye información sobre el estado de los sistemas del dron), y las señales de video, las cuales son necesarias para realizar tareas de monitoreo o inspección. El flujo constante y confiable de estos datos también mejora la seguridad durante la operación, ya que los operadores pueden tener una visión completa de la situación en todo momento [74].

- **Control y Comando:** La comunicación de control es fundamental para que el operador pueda dirigir el dron con precisión. Sin esta conexión, el dron no podría recibir instrucciones, lo que lo haría incapaz de llevar a cabo tareas útiles. Esta comunicación permite que el operador ajuste en tiempo real la

dirección, velocidad y altitud del dron, asegurando un manejo adecuado y adaptado a las necesidades de la misión. Gracias a ella, se puede modificar el comportamiento del dron de manera constante, lo que facilita un control exacto en todo momento [74].

- **Telemetría y Monitoreo:** Mientras el dron está en vuelo, es importante que el operador esté al tanto de su estado en todo momento. La telemetría y el monitoreo permiten que el dron envíe datos sobre su nivel de batería, altitud y velocidad. Estos datos constantes permiten que el operador se mantenga informado sobre el rendimiento del dron. Con esta información, el operador puede tomar decisiones rápidas y solucionar cualquier inconveniente antes de que se convierta en un problema. Si la batería está baja o el dron no está funcionando como debería, el operador puede intervenir a tiempo y mantener la misión segura [74].

- **Transmisión de Datos:** Los drones suelen estar equipados con diversos sensores que recopilan información durante el vuelo. A través del sistema de comunicación, estos datos se transmiten al operador en tiempo real, en lugar de quedar almacenados solo en el dispositivo. Esta capacidad de enviar información de manera continua permite al operador monitorear y ajustar la operación de inmediato. Al recibir estos datos en tiempo real, el operador puede tomar decisiones rápidas y modificar la misión si es necesario, mejorando así la efectividad y adaptabilidad de la operación [74].

1.4.9. Normativas y reglamentos

En la siguiente sección se detallan las normativas y reglamentos, tanto a nivel nacional como internacional, que rigen el uso de cuadricópteros. Se proporciona un resumen de estos reglamentos con el fin de asegurar una operación segura de aeronaves no tripuladas.

1.4.9.1 La Organización de Aviación Civil Internacional (OACI)

La Organización de Aviación Civil Internacional (OACI) es el organismo responsable de definir los lineamientos globales en materia de aviación civil, incluyendo el uso de aeronaves no tripuladas (drones). Esta entidad, adscrita a la Organización de las Naciones Unidas (ONU), fue creada en 1944 con el fin de afrontar los retos inherentes a la aviación internacional. Su función principal consiste en desarrollar estándares y recomendaciones que promuevan operaciones aéreas seguras, eficientes y armonizadas a nivel mundial [75]. Entre sus objetivos más relevantes se encuentran:

- Formulación de normativas aeronáuticas internacionales que garanticen la seguridad operacional en el espacio aéreo [75].

- Fomento del desarrollo tecnológico para la optimización de infraestructuras como rutas aéreas, aeropuertos y sistemas de navegación aérea [75].

- Promoción de la equidad competitiva entre Estados, impulsando la eficiencia en el uso de recursos y asegurando imparcialidad en la aplicación de regulaciones [75].

1.4.9.2 La Agencia de la Unión Europea para la Seguridad Aérea (AESA)

La Agencia de la Unión Europea para la Seguridad Aérea (AESA) fue constituida el 4 de julio de 2018 con el propósito de uniformar la normativa en el ámbito de la aviación civil dentro de la Unión Europea [75]. Desde el 31 de diciembre de 2020, está en vigor una legislación específica para la operación de sistemas de aeronaves no tripuladas (UAS), también conocidos como drones [75]. Esta normativa contempla los siguientes elementos clave:

A. **Categoría Abierta:** Aplicable a vuelos de bajo riesgo, que pueden realizarse sin autorización previa, siempre que se cumplan ciertas condiciones básicas [75].

B. **Categoría Específica:** Dirigida a operaciones de riesgo medio, las cuales requieren evaluación y aprobación por parte de la autoridad competente [75].

C. **Categoría Certificada:** Reservada para misiones de alto riesgo, con exigencias similares a las de la aviación tripulada en cuanto a certificación del operador y de la aeronave [75].

En el marco de la normativa vigente sobre el uso de drones, se han establecido una serie de requisitos fundamentales para garantizar la seguridad, el control y la responsabilidad en las operaciones. Entre las principales disposiciones se incluyen el registro obligatorio de operadores, la regulación de las zonas de vuelo y la exigencia de un seguro de responsabilidad civil.

- Registro obligatorio de operadores: Todos los operadores deben registrarse ante la AESA y colocar visiblemente en sus drones el número de registro asignado [75].
- Zonificación del espacio aéreo: Se establecen áreas específicas con condiciones de operación diferenciadas permitidas, restringidas o prohibidas, considerando aspectos de seguridad, privacidad y control del tráfico aéreo [75].
- Seguro de responsabilidad civil obligatorio: Se exige una póliza que cubra los daños a terceros durante la operación de drones. Esta medida aplica tanto para actividades recreativas como comerciales, siendo obligatoria para aeronaves con un peso máximo al despegue (MTOM) igual o superior a 20 kg [75].

1.4.9.3 Reglamentación europea vigente en materia de drones

Actualmente, el marco normativo europeo para el uso de drones está compuesto por los siguientes reglamentos:

- Reglamento de Ejecución (UE) 2019/947 (24 de mayo de 2019): Establece las condiciones generales para la utilización de aeronaves no tripuladas en el espacio aéreo de la UE [76].
- Reglamento de Ejecución (UE) 2020/639 (12 de mayo de 2020): Modifica el Reglamento 2019/947, introduciendo ajustes en los escenarios estándar de operación, ya sea dentro o fuera del alcance visual del operador [76].

- Reglamento de Ejecución (UE) 2020/746 (4 de junio de 2020): También modifica el Reglamento 2019/947, incorporando prórrogas en los plazos de implementación debido al impacto de la pandemia por COVID-19 [76].
- Reglamento Delegado (UE) 2019/945 (12 de marzo de 2019): Regula los requisitos técnicos de los sistemas UAS y las condiciones de operación para operadores provenientes de países no pertenecientes a la Unión Europea [76].
- Reglamento Delegado (UE) 2020/1058 (27 de abril de 2020): Introduce enmiendas al Reglamento 2019/945, añadiendo dos nuevas categorías de sistemas de aeronaves no tripuladas [76].

1.4.9.4 Normativas UAS – Regulaciones generales para el uso de drones en Ecuador

La **Gestión de Aviación Civil de Ecuador (GDCAE)**, entidad encargada de la seguridad en la operación de drones, ha puesto a disposición información en línea sobre los requisitos para vuelos recreativos y comerciales. A continuación, se presentan los puntos clave. Para obtener detalles adicionales, se recomienda consultar la fuente oficial [77].

Regulaciones sobre el Uso de Drones en Ecuador

El uso de drones en Ecuador está autorizado bajo el cumplimiento de las normativas establecidas por la **GDCAE** [77]. Las principales reglas que deben seguirse al operar un dron en el país son las siguientes:

- Los vuelos deben realizarse únicamente durante el día y en condiciones meteorológicas favorables con visibilidad clara [77].
- La altitud máxima permitida es de 400 pies.
- Es obligatorio mantener el dron dentro del campo de visión directa del operador en todo momento [77].
- Se prohíbe volar a menos de 9 kilómetros (5,5 millas) de cualquier aeropuerto o aeródromo.
- No está permitido operar drones en áreas sensibles, como instalaciones gubernamentales o militares [77].
- Se pueden emplear maniobras de vuelo automatizadas siempre que el piloto tenga la capacidad de intervenir en cualquier momento durante la operación [77].

1.4.9.5 Seguro de responsabilidad civil

El propietario u operador de un sistema de aeronave pilotada a distancia (RPAS/UAS) es responsable de los posibles daños ocasionados a terceros durante la operación del dron [77]. Por ello, se exige la contratación de un seguro de responsabilidad civil con coberturas mínimas establecidas de la siguiente manera:

- Para drones con un peso de despegue entre 2 kg y 25 kg: cobertura mínima de USD 3.000,00 [77].
- Para drones con un peso superior a 25 kg: cobertura mínima de USD 5.000,00 [77].

Restricciones en las Galápagos

El uso de drones en las Islas Galápagos está prohibido por el Parque Nacional Galápagos, con el objetivo de proteger su ecosistema. La normativa busca evitar posibles afectaciones a la fauna local, ya que las aves y otros animales pueden verse alterados o colisionar con estas aeronaves [77].

Solo se permite el uso de drones para actividades de monitoreo e investigación, o en casos específicos con una autorización especial para fines comerciales. Para más información, se recomienda contactar con la Dirección del Parque Nacional Galápagos (GNPD) [77].

1.5. Marco contextual

El proyecto se centra en la necesidad de desarrollar un sistema inteligente de detección y seguimiento de objetos en movimiento. Este sistema busca tener una base hacia futuros proyectos y también mejorar la capacidad de los UAV (vehículos aéreos no tripulados) en la provincia de Santa Elena, en la que pueden responder a diferentes áreas de emergencias, tales como monitoreos, vigilancia y rescates en tiempo real, etc.

El proyecto será desarrollado en un laboratorio de la Universidad Estatal Península de Santa Elena (UPSE), un espacio con dimensiones adecuadas y condiciones apropiadas para llevar a cabo las pruebas necesarias. Se garantizará una correcta distribución de los objetos a detectar, de modo que su identificación por parte del dron sea clara y sin inconvenientes. Además, se procurará que el área de trabajo cuente con una iluminación suficiente para evitar interferencias en los sensores del UAV, asegurando así que su sistema de visión funcione correctamente.

Cabe destacar que los objetos serán detectados principalmente por su forma; sin embargo, el color también influye en este proceso. Por ejemplo, si un objeto es de color blanco y el entorno presenta tonalidades similares o muy claras, su detección puede dificultarse, ya que tiende a perderse visualmente en el fondo. Lo mismo ocurre con objetos de color negro u oscuros si el entorno también presenta poca iluminación o colores similares. Por esta razón, se recomienda trabajar en un entorno controlado, donde se pueda asegurar un contraste adecuado entre los objetos y el fondo, facilitando así su correcta identificación. Es importante mencionar que, además del laboratorio, este proyecto podrá ser probado en otras áreas dentro de la universidad, como los patios, los corredores o incluso algunas aulas, siempre y cuando se cumplan las medidas de seguridad requeridas.

La elección de estos entornos no es arbitraria, sino que responde a la necesidad de que el sistema mantenga una distancia adecuada con el objeto a seguir, evitando pérdidas de señal o errores en la detección. Si se intentara aplicar este proyecto

en espacios significativamente más grandes, como un estadio de fútbol, se requeriría una reconfiguración del sistema para adaptarse a las nuevas condiciones, lo que podría aumentar el riesgo de fallos en el reconocimiento del objeto. Por esta razón, se han seleccionado entornos controlados dentro de la universidad, donde es posible minimizar estos inconvenientes y garantizar un ambiente propicio para el desarrollo y prueba del UAV, optimizando su desempeño antes de considerar aplicaciones en escenarios más complejos.

Este sistema podría ser implementado en diversas situaciones donde la vigilancia y el seguimiento automático sean fundamentales para mejorar la seguridad y la capacidad de respuesta. Un dron seguidor de objetos podría ser de gran utilidad tanto en seguridad y vigilancia como en búsqueda y rescate. En el ámbito de la seguridad, podría monitorear carreteras y zonas urbanas para detectar y seguir vehículos sospechosos, como aquellos que circulan sin placas o han sido reportados como robados, transmitiendo su ubicación a las autoridades para facilitar una intervención precisa sin recurrir a persecuciones peligrosas. Por otro lado, en misiones de rescate, podría localizar excursionistas perdidos en montañas mediante cámaras térmicas o señales de emergencia, seguir sus movimientos y guiar a los rescatistas hasta su ubicación, aumentando las probabilidades de supervivencia en situaciones extremas.

2. Métodos y diseño experimental

2.1. Métodos

Para llevar a cabo el proyecto de desarrollo de un dron detector y seguidor de objetos, se realizará un análisis detallado de los objetivos del sistema de detección y seguimiento autónomo. Se establecerán las especificaciones necesarias, como el tamaño, la velocidad del dron, el tipo de cámara y el nivel de precisión requerido. Con estos criterios definidos, se seleccionarán los componentes adecuados y se procederá a capturar imágenes en diversas condiciones para su posterior procesamiento. Se entrenará un modelo de visión artificial, utilizando técnicas de aprendizaje profundo como las redes neuronales convolucionales (CNN), para detectar con precisión el objeto a seguir. Este modelo será probado y ajustado hasta alcanzar un desempeño aceptable. Luego, se integrará en el sistema del dron para que pueda operar de forma autónoma.

Durante el desarrollo, se evaluarán las entradas y salidas del sistema, asegurando una correcta interpretación de los datos. El dron capturará imágenes en tiempo real, las procesará y analizará para identificar la posición del objeto, generando comandos que ajusten su movimiento con el fin de mantener una distancia adecuada. Se implementarán mecanismos de control para evitar obstáculos y se harán ajustes en tiempo real para mejorar el rendimiento del seguimiento. Asimismo, se desarrollará una interfaz gráfica que permitirá visualizar la detección y controlar parámetros relevantes. El proyecto se enfocará en entornos cerrados y bien iluminados, lo que permitirá obtener mejores resultados con los recursos disponibles. El desarrollo se llevará a cabo utilizando Python como lenguaje principal y PyCharm como entorno de trabajo. Finalmente, se documentará todo el proceso y se establecerá un plan de mantenimiento que garantice la continuidad y fiabilidad del sistema.

2.1.1. Descripción del proyecto.

Se implementará un sistema de detección y seguimiento de objetos en un vehículo aéreo no tripulado (UAV), específicamente en un cuadricóptero. Esta aeronave estará equipada con una cámara frontal, que será importante para capturar imágenes del entorno. Las imágenes obtenidas serán procesadas mediante redes neuronales convolucionales (CNN) implementadas en Python. El sistema llevará a cabo dos tareas principales: identificación, la cual se encargará de detectar y localizar el objeto en las imágenes, y clasificación, que asignará una etiqueta adecuada a cada objeto, como 'botella', 'taza' etc. Una vez que las imágenes han sido procesadas y los objetos han sido clasificados correctamente, el UAV se encargará de seguir el objeto identificado. Si el objeto detectado tiene un desplazamiento o movimiento, el robot aéreo ajustará su trayectoria para seguir su ruta la cuál utilizará datos de su IMU y sistema de posicionamiento. La comunicación se realizará mediante wifi. Además, el sistema del dron por defecto incorporara medidas de seguridad, como modos de emergencia en caso de pérdida de señal o batería baja. A continuación, en la Figura 61 se muestra un esquema general del sistema a implementar.

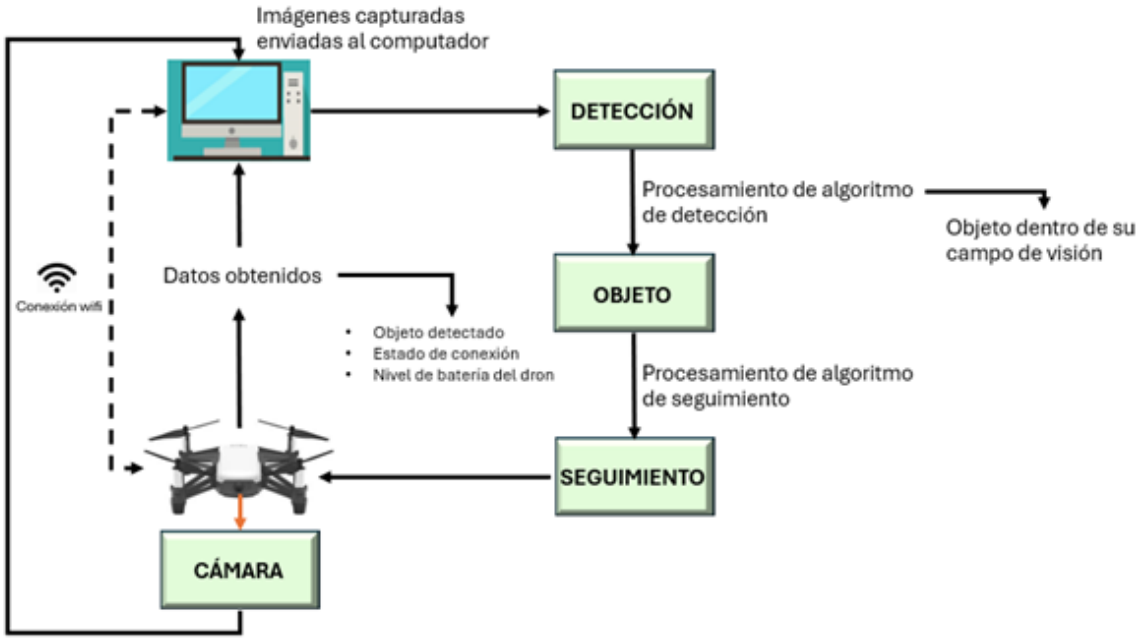


Figura 58: Diagrama general del sistema de detección y seguimiento de objetos. [Autoría propia]

2.2. Componentes de la propuesta

Para llevar a cabo la implementación del sistema inteligente de detección y seguimiento de objetos, es necesario contar con varios componentes que permitan tanto el procesamiento de imágenes como la movilidad y estabilidad del dron. En este apartado, se describirán los componentes que se utilizarán en este proyecto, con el fin de proporcionar una mejor comprensión de su funcionamiento al momento de poner el sistema en marcha.

2.2.1. Componentes físicos

En la robótica, los componentes físicos juegan un papel clave, ya que cada uno de ellos contribuye a la estructura y funcionamiento del robot. Estos componentes se clasifican en distintas categorías, como sensores, que recopilan información del entorno; actuadores, que permiten el movimiento y la ejecución de tareas; y el mecanismo estructural, que brinda soporte y estabilidad al sistema. A continuación, se mencionarán los siguientes componentes del robot y su función dentro de la estructura.

2.2.1.1 Dron Tello DJI

El Tello DJI es un dron desarrollado por Ryze Tech en colaboración con DJI e Intel. Es un robot aéreo de pequeñas dimensiones conformado con cuatro motores, y por esta razón se lo conoce como cuadricóptero. Puede ser controlado manualmente o programado para ejecutar tareas específicas mediante la incorporación de códigos. Además, permite la integración de inteligencia artificial, lo que amplía sus capacidades. Este dron representa una gran oportunidad para que las bibliotecas promuevan

el aprendizaje en tecnología y programación STEM. Cuenta con comunicación wi-fi, lo que le permite conectarse tanto a computadoras como a dispositivos móviles inteligentes.

Figura 59: dron Tello DJI.

Fuente: [78].

En el la Tabla 1 se muestran las especificaciones técnicas del dron Tello DJI, describiendo sus características importantes para su funcionamiento.

Especificaciones	**Detalles**
Peso total del dron	87 gr
Velocidad máxima	28,8 km/h (17,8 mph)
Tiempo máximo de vuelo	13 minutos (en ausencia de viento a 15 km/h [9 mph] constantes).
Rango de temperaturas operativas	De 0 °C a 40 °C (de 32 °F a 104 °F)
Rango de frecuencias operativas	De 2.4 a 2.4835 GHz
Transmisor (Pire)	20 dBm (FCC), 19 dBm (CE), 19 dBm (SRRC)

Tabla 1: Especificaciones técnicas del dron [1]

2.2.1.2 Estructura y chasis

El chasis y la estructura del dron, como se muestra en la Figura 60, están compuestos por materiales ligeros y resistentes, como plástico y carbono. La principal función del chasis es proporcionar soporte y protección a todos los componentes internos del dron, como los motores, la batería y la cámara, al mismo tiempo que ayuda a regular el peso del dron para su estabilidad durante el vuelo. Tener un chasis resistente permite que el dron soporte impactos leves sin que los componentes internos sufran daños.

Figura 60: Estructura de dron Tello DJI. [Autoría propia]

2.2.1.3 Placa base

La Placa base Tello Flight Controller Board como se muestra en la Figura 61, es el cerebro principal en donde se encuentran todos los circuitos electrónicos y componentes importantes que permiten el funcionamiento del dron. Esta placa conecta todos los elementos, como los motores, sensores, cámara, el controlador de vuelo y la batería. Su procesador de vuelo ejecuta algoritmos avanzados de estabilización y control de trayectoria, basándose en la información recibida, el sensor de posicionamiento visual y la cámara. También regula la velocidad y potencia de los motores mediante los controladores electrónicos de velocidad (ESC).

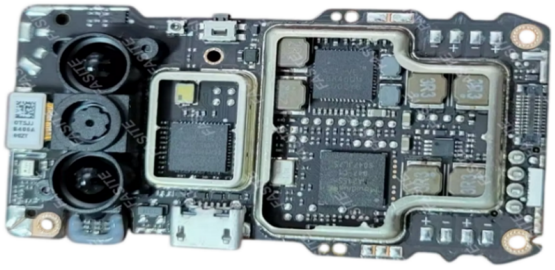


Figura 61: Placa base del dron Tello DJI. [Autoría propia]

Esta placa tiene integrado un módulo wifi que permite realizar conexiones con diferentes dispositivos. En términos de seguridad, la placa incorpora sistemas de protección que detectan variaciones en la energía y ajustan el rendimiento para evitar fallos en vuelo. Su capacidad de procesamiento permite realizar correcciones automáticas ante cambios en el entorno, como ráfagas de viento o movimientos inesperados. Gracias a esta placa, el Tello DJI puede ejecutar maniobras inteligentes, mantenerse en posición sin GPS y responder de manera rápida las órdenes del usuario.

2.2.1.4 Sistema de propulsión

La propulsión es el proceso mediante el cual se genera una fuerza para mover un objeto, en este caso, un robot aéreo. Este proceso se logra gracias a los motores

y las hélices que trabajan en conjunto con los actuadores produciendo la potencia necesaria para que la aeronave ascienda, descienda y se desplace en diferentes direcciones. En términos simples, estos dos componentes funcionan como el "sistema de impulso"del dron, ya que convierten la energía en movimiento mecánico.

Motores sin escobillas

Este dron está conformado por 4 motores de corriente continua sin escobillas, responsables de impulsar las hélices y permitir el despegue, vuelo y realización de maniobras. Los motores del Tello DJI tienen dos sentidos de giro, horario y antihorario. Los dos motores delanteros giran hacia adelante, mientras que los traseros lo hacen en sentido contrario, asegurando estabilidad y control durante el vuelo. Estos sentidos de giro se pueden identificar por los colores de los cables, donde los motores que giran hacia adelante están conectados a cables de colores azul y negro, y los que giran hacia atrás a cables de color blanco y negro como se muestra en la Figura 62.

Figura 62: Motores sin escobillas del dron Tello DJI. [Autoría propia]

En la Tabla 2 se muestran las especificaciones técnicas del motor sin núcleo, detallando sus especificaciones más relevantes para su uso.

Especificaciones	**Detalles**
Longitud del motor	20,0 mm
Diámetro del motor	8,5 mm
Diámetro del eje	1,0 mm
Longitud del eje	4 mm
Peso	∼5,0 gramos (incluye motor y cables)
Entrada de energía	$\leq$ 4,2 V (máximo 5 V pulsado)
Batería	LiPo 1S
Empuje máximo	∼40 g (1 pieza)
Velocidad sin carga	A 3 V: 35 000–37 000 rpm A 5 V: ∼50 000 rpm
Uso	Compatible con drones Ryze y otros drones RC de 100 mm y 110 mm

Tabla 2: Especificaciones técnicas del motor sin núcleo [2]

Hélices 3044P

Las hélices forman parte del sistema de propulsión del dron DJI Tello y van montadas en los cuatro motores sin escobillas. Están hechas especialmente para este modelo.

Las hélices se dividen en dos grupos: dos están destinadas a los motores que giran hacia adelante, mientras que las otras dos son para los motores traseros como se muestra en la Figura 63, ya que están diseñadas específicamente para adaptarse al giro de los motores. Son ligeras pero resistentes, lo que permite un uso prolongado sin perder eficiencia. Estas hélices están pensadas para mantener el dron DJI Tello con un desempeño único y original en cada vuelo.

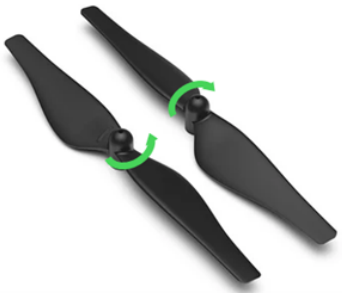


Figura 63: Hélices del dron Tello DJI. [Autoría propia]

En la Tabla 3 se especifican y detallan los datos de las hélices 3044P del dron Tello DJI

Especificaciones	**Detalles**
Longitud de la hélice	3 pulgadas (76,3 mm)
Ancho de la hélice	0,44 pulgadas (11 mm)
Apertura de la hélice	1,0 mm
Peso	0,5 g/ por pieza

Tabla 3: Especificaciones técnicas de la hélice [1]

2.2.1.5 Sensores

Los sensores son esenciales para que los robots, funcionando de manera similar a cómo los ojos nos permiten a los seres humanos ver el mundo. Por tal razón, se llaman sensores, ya que tienen la capacidad de captar o detectar el ambiente en el que se encuentran, gracias a estos dispositivos, los robots pueden identificar diversos aspectos del entorno, como la distancia a objetos, la temperatura o la velocidad del viento. Las cámaras también actúan como sensores, ya que permiten al robot observar su entorno a través de imágenes y videos en tiempo real. Si un código en específico se integrara a un dispositivo con cámara, este dispositivo operaría de manera autónoma. En ese caso, la cámara se convierte en un sensor adicional, contribuyendo a la interpretación del entorno del robot.

Sensor de posicionamiento visual

El Sistema de Posicionamiento Visual permite que la aeronave mantenga su posición de manera que flote en el aire con mayor estabilidad. Esto hace que el dron pueda volar tanto en interiores como en exteriores, especialmente cuando no hay viento. Este sistema consta de una cámara y un módulo de infrarrojos 3D ubicados en la parte inferior de la aeronave como se muestra en la Figura 64. Al encender el dron, el sistema se activa automáticamente sin necesidad de realizar ninguna acción extra.

Su funcionamiento es más eficiente cuando el dron se encuentra entre 0,3 y 6 metros de altura (1,0 a 19,7 pies), y su rendimiento puede verse afectado si el dron vuela a mayores altitudes, si el sistema está fuera de los limites descritos puede verse afectado [1].

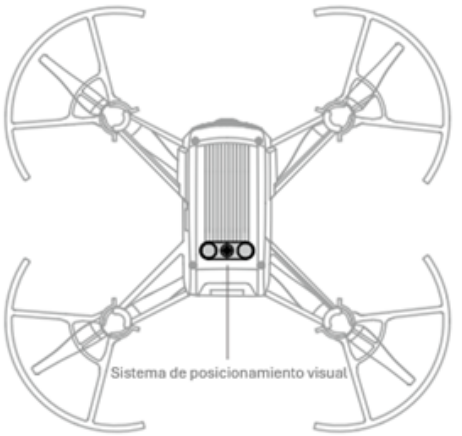


Figura 64: Sensor de posicionamiento visual del dron Tello DJI.

Fuente: [1].

Cámara

La cámara del dron DJI Tello cuenta con un sensor de 5 megapíxeles, capaz de capturar imágenes con una resolución de 2592 × 1936 píxeles la cual permite obtener fotografías bastante aceptables. En cuanto a la grabación de vídeo, es capaz de grabar en una resolución de 1280 × 720p a 30 cuadros por segundo (fps) en formato MP4, que nos da como resultado una calidad de imagen fluida. Además, tanto las fotos como los vídeos pueden visualizarse y administrarse mediante la aplicación Tello, permitiendo explorar cómodamente y descargarlos en un dispositivo móvil [79]. En la Figura 65 se observa la cámara del dron, ubicada en la parte frontal.

Figura 65: Cámara frontal del dron Tello DJI. [Autoría propia]

En la Tabla 4 se presenta las especificaciones y detalles de la cámara de 5 megapíxeles.

Especificaciones	Detalles
Resolución máxima de imagen	2592 x 1936
Modos de grabación de video	HD: 1280 x 720 30p
Formato de video	MP4

Tabla 4: Especificaciones de captura de imagen y video [1]

2.2.1.6 Sistema de alimentación

Los sistemas de alimentación en robótica son componentes importantes, ya que se encargan de suministrar energía a todo el sistema, desde robots pequeños hasta grandes, tanto terrestres como aéreos. Esta energía puede provenir de baterías o, en ciertos casos, a través de un puerto de conexión que abastece el dispositivo. Existen diversas opciones de baterías, pero en este caso nos centraremos en la alimentación y el mecanismo de carga del dron ya mencionado anteriormente.

Batería

La batería del dron Tello es fácilmente accesible y desmontable, se puede cargar utilizando un cargador especialmente diseñado para este componente como se observa en la (Figura 66 A) o, alternativamente, insertando la unidad directamente en el dron y conectándola a un puerto de entrada micro USB como muestra la (Figura 66 B). Esta batería, con una capacidad de 1100 mAh y un voltaje de 3,8 V, es responsable de energizar todo el circuito del dron. Está equipada con varias protecciones para garantizar su seguridad y funcionamiento. Antes de cada vuelo, es recomendable cargarla completamente. Incluye protección contra sobre corriente y sobretensión, lo que hace que deje de cargarse si detecta una corriente o voltaje excesivos. También dispone de protección contra cortocircuitos, desconectando el sistema de energía en caso de detectar uno. Además, cuenta con un sistema de protección contra sobre descarga, que interrumpe la descarga automáticamente para evitar daños por una descarga excesiva.

(A) Carga de batería por cargador especial (B) Carga de batería por USB

Figura 66: Comparación de resultados en distintas condiciones. [Autoría propia]

Tiene un tiempo de carga de 1 hora y 30 minutos. Para saber si está completamente cargada podremos observarlo en la parte frontal del cuadricóptero que cuenta con un indicador de estados justo alado de la cámara, como se muestra en la Figura 67, cuando el indicador se queda con un color azul fijo, sin parpadear, significa que la batería está completamente cargada.

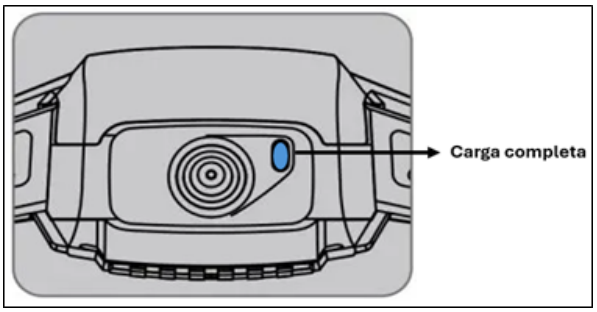


Figura 67: Indicador de estado del dron Tello DJI.

Fuente: [1].

En la Tabla 5 se describen las especificaciones técnicas de la batería que utiliza el dron Tello DJI.

Especificaciones	**Detalles**
Capacidad	1100 mAh
Voltaje	3.8 V
Tipo de batería	LiPo
Energía	4.18 Wh
Peso total	2 gr
Rango de temperaturas operativas	De 5 ° a 45 °C (de 41 ° a 113 °F)
Potencia de carga máxima	10 W

Tabla 5: Especificaciones de la batería [1]

2.2.2. Componentes lógicos

En esta sección nos enfocaremos en los componentes lógicos. Dentro de este tema, encontraremos ciertas librerías que se utilizarán, lenguaje de programación que se empleará y otros parámetros que serán clave para la realización de este proyecto. Estos componentes son de gran importancia, ya que el dron trabajará en conjunto con los comandos proporcionados al sistema de la aeronave, lo que permitirá una integración eficiente y un buen rendimiento.

2.2.2.1 Lenguaje de programación Python

Python es un lenguaje de programación de alto nivel, conocido a nivel mundial debido a su facilidad de comprensión y uso en diferentes sistemas, como en este caso, se ha elegido Python para programar el dron Tello DJI, ya que es compatible con el dron y permite desarrollar el proyecto de manera sencilla, aumentando las probabilidades de éxito. Este lenguaje facilita la integración de sistemas inteligentes que permite el procesamiento de imágenes y videos de manera eficiente. Para este proyecto, se utilizarán bibliotecas especializadas en visión artificial y machine learning, como OpenCV y NumPy, que proporcionan herramientas avanzadas para el análisis y manipulación de imágenes [80].

2.2.2.2 Entorno de desarrollo integrado (IDE)

los IDE son entornos diseñados para simplificar el proceso de programación. Incluyen herramientas que ayudan a los desarrolladores a escribir, probar y corregir código de manera más rápida. Además, estos entornos han sido creados gracias al trabajo conjunto de muchos programadores, con el objetivo de simplificar la creación de aplicaciones y sistemas inteligentes. Dentro de los IDE encontramos PyCharm, un entorno especializado en Python, que ofrece una interfaz intuitiva, herramientas avanzadas para edición de código y compatibilidad con múltiples frameworks que lo convierte en una buena opción para el desarrollo en este lenguaje.

PyCharm

Es uno de los entornos de desarrollo integrado (IDE) más reconocidos y utilizados para el lenguaje de programación Python, gracias a una serie de características que lo convierten en una opción integral y eficiente. Este IDE está disponible como una aplicación multiplataforma, lo que significa que es compatible con sistemas operativos como Linux, macOS y Windows, que ofrece flexibilidad para diferentes usuarios y plataformas noauthor[81].

Lo que hace que PyCharm se distinga de otros IDE es la variedad de herramientas y módulos que incluye, diseñados para agilizar el proceso de desarrollo de aplicaciones en Python, lo que permite a los desarrolladores reducir significativamente el tiempo y esfuerzo necesarios en la codificación. Además, este entorno es altamente personalizable, ayudando a los usuarios adaptarlo a sus necesidades específicas o preferencias de trabajo.

Entre las principales características de PyCharm se encuentran:

- Depurador gráfico: Esta herramienta facilita la identificación y corrección de errores en el código, mejorando el proceso de depuración.
- Compatibilidad con bases de datos: PyCharm permite trabajar con varias bases de datos de manera directa, sin necesidad de herramientas externas.
- Pruebas unitarias integradas: El IDE incluye un probador de unidad que facilita la creación y ejecución de pruebas automáticas que asegura la calidad de la codificación.
- Integración con sistemas de control de versiones (VCS): PyCharm tiene soporte para herramientas como Git, lo que permite llevar un control adecuado de las versiones del código y facilita la colaboración entre múltiples desarrolladores.
- Soporte para la ciencia de datos: Con la integración de Python, PyCharm es especialmente útil para proyectos relacionados con análisis de datos y machine learning, ofreciendo un entorno estable para trabajar con bibliotecas como NumPy y Pandas.

PyCharm también permite trabajar con otros lenguajes como HTML, CSS y JavaScript, lo que amplía sus capacidades más allá del ámbito de Python. Además, su interfaz de usuario es moderna y puede personalizarse mediante complementos (plugins), adaptándose a las preferencias y necesidades de cada programador.

2.2.2.3 Librerías

Las bibliotecas son herramientas fundamentales en la programación, ya que permiten ampliar las capacidades de un lenguaje sin necesidad de desarrollar todo desde cero. Facilitan tareas complejas como la creación de interfaces gráficas mediante código previamente escrito y compartido por otros usuarios. Además, simplifican la codificación para desarrollar el sistema inteligente de detección y seguimiento de objetos del proyecto, aunque muchas veces podríamos escribir ese código por nuestra cuenta, el uso de bibliotecas ahorra tiempo y esfuerzo. También es común que una biblioteca dependa de otras o que utilice funciones no estándar para ofrecer características adicionales que no están presentes en las bibliotecas estándar del lenguaje. A continuación, se describen las bibliotecas que se utilizarán, cómo funcionan dentro de este sistema y el desempeño en el desarrollo de este trabajo de implementación.

Djitellopy

Es una biblioteca disponible en el entorno de desarrollo de Python, diseñada especialmente para el control del dron DJI Tello. Esta biblioteca permite que el dron ejecute todos los movimientos que se le indiquen, ya sea mediante un control remoto o a través de comandos programados en un código. Djitellopy se basa en el SDK oficial del Tello, lo que significa que actúa como una interfaz entre el lenguaje Python y el dron, facilitando la comunicación mediante el envío de comandos por red wifi [82]. Gracias a esta integración, es posible acceder a múltiples funciones del dron de forma sencilla y eficiente. Además, no solo se encarga del desplazamiento del dron, sino que también permite realizar diversas acciones adicionales, estas acciones las realiza con bloques de código predefinido que pertenecen a distintas clases, las cuales se describen a continuación:

- **Control de vuelo:** En esta función podremos encontrar los comandos de los múltiples movimientos que realiza el dron. A continuación, en la Tabla 6 se muestran los comandos básicos que realiza el cuadricóptero.

Comando	**Descripción**
`tello.connect()`	Conecta con el dron Tello DJI
`tello.takeoff()`	Despegue
`tello.land()`	Aterrizaje
`tello.move_up()`	Subir
`tello.move_down()`	Bajar
`tello.move_left()`	Desplazamiento a la izquierda
`tello.move_right()`	Desplazamiento a la derecha
`tello.rotate_clockwise(90)`	Giro de 90° a la derecha en su mismo eje
`tello.rotate_counter_clockwise(90)`	Giro de 90° a la izquierda en su mismo eje

Tabla 6: Comandos básicos del dron DJI Tello usando Djitellopy [3]

- **Manejo de la cámara:** En esta sección, la biblioteca Djitellopy permite el manejo de la cámara del dron, que ofrece funciones como se observan en la Tabla 7.

Comando	Descripción
`tello.streamon()`	Iniciar transmisión de video
`tello.streamoff()`	Detener transmisión de video
`frame_read = tello.get_frame_read()`	Acceder al frame de la transmisión de video
`frame = frame_read.frame`	Tomar una foto (capturar un solo frame)

Tabla 7: Comandos de transmisión y captura de video con Djitellopy [3]

- **Datos obtenidos del dron:** Accede a la información del estado en el que se encuentra el dron como se muestra en la Tabla 8.

Comando	Descripción
`tello.get_battery()`	Nivel de batería
`tello.get_height()`	Altura del dron
`tello.get_temperature()`	Temperatura interna del dron
`tello.get_flight_time()`	Tiempo de vuelo
`tello.get_speed()`	Velocidad del dron
`tello.get_wifi()`	Estado de la conexión wifi

Tabla 8: Comandos para obtener datos del dron con Djitellopy [3]

- **Comandos de vuelo autónomos:** Para comandos de vuelo autónomos con el DJI Tello usando la biblioteca Djitellopy, no hay una implementación directa de rutas predeterminadas. Sin embargo, se pueden combinar los comandos de vuelo básicos con sensores o a través de una programación hecha por los usuarios o programadores, lo que permite personalizar las rutas de vuelo. Siendo este el caso, se integrará un sistema en el dron que realice movimientos autónomos al detectar un objeto.

Numpy

Es una biblioteca de Python que mejora el manejo de matrices en comparación con las listas tradicionales de este software. Aunque las listas de Python pueden usarse para trabajar con matrices, es recomendable utilizar Numpy ya que tiene matrices que trabajan de forma más rápida porque se van almacenando en cada instante en la memoria, facilitando un acceso y procesamiento más veloz en los datos [83].

Numpy interviene en el programa al manejar y procesar arreglos numéricos que contienen las probabilidades de que cada objeto detectado pertenezca a una determinada categoría. Gracias a esta biblioteca, es posible almacenar y manipular estos valores de manera eficiente, lo que permite realizar operaciones matemáticas y análisis precisos sobre los datos. Esto facilita la identificación de la categoría más probable para cada objeto detectado y su posterior etiquetado. Además, los modelos de detección proporcionan las posiciones de los objetos en valores normalizados, es decir, en una escala binaria de 0 y 1, sin considerar el tamaño de la imagen. Para obtener las coordenadas reales en píxeles, que dependen del tamaño del fotograma,

se utiliza Numpy para generar un vector con las dimensiones de la imagen. Al multiplicar los valores normalizados por este vector, se obtienen las coordenadas precisas que permiten dibujar cuadros y ubicar los objetos correctamente en la imagen.

OpenCV

Es una biblioteca de código abierto muy popular para el procesamiento de imágenes y visión por computadora. Permite realizar tareas como leer, escribir y modificar imágenes, así como aplicar transformaciones geométricas, detectar objetos y realizar análisis avanzados, como el seguimiento de objetos y la detección de rostros. Además, es compatible con varios lenguajes de programación, lo que la hace muy útil en una gran variedad de proyectos, desde investigaciones hasta desarrollos en inteligencia artificial [84],[85].

Procesamiento de imágenes: OpenCV es una herramienta poderosa para el procesamiento de imágenes y videos, que permite realizar tareas como cargar, modificar, transformar y guardar imágenes en diferentes formatos. A través de sus funciones, es posible cambiar el tamaño, recortar imágenes, aplicar filtros y detectar bordes. Además, tiene métodos para detectar características clave en las imágenes, como esquinas y puntos de interés, utilizando técnicas como el operador Canny para la detección de bordes, o métodos como SIFT y SURF (aunque existen alternativas como ORB debido a las patentes de los primeros) [84], [85].

También facilita el reconocimiento de objetos, como la detección de rostros usando clasificadores en cascada Haar hasta el uso de redes neuronales para tareas más complejas. Asimismo, se pueden encontrar herramientas de segmentación de imágenes, que permiten dividir una imagen en diferentes regiones para identificar objetos o características de manera más fácil [86]. Otras de sus características permiten el manejo de videos, capturando y procesando secuencias en tiempo real, realizando operaciones sobre cada fotograma y analizando el movimiento.

2.3. Diseño experimental

Este proyecto se centrará en el desarrollo de un sistema inteligente para el cuadricóptero, basado en diversas técnicas de visión artificial, que garantizarán su correcto funcionamiento. A través de este sistema, podremos evaluar el rendimiento del dron según parámetros clave, como la velocidad de vuelo, la distancia de seguimiento del objeto, el tiempo de reconocimiento, el procesamiento necesario durante la ejecución del programa y la iluminación requerida para la detección y seguimiento de objetos. Todo esto se realizará utilizando la cámara estándar del dron. Además, se podrán visualizar en tiempo real ciertos parámetros y actividades del cuadricóptero a través de una interfaz, permitiendo monitorear su desempeño mientras está en funcionamiento.

2.3.1. Implementación del hardware

El dron DJI Tello es una aeronave con una estructura fija no desarmable, lo que significa que todos sus componentes están ensamblados permanentemente desde fábrica. En la Figura 68 se observa sus elementos internos en donde se encuentran

los cuatro motores eléctricos con sus respectivas hélices, que le permiten ejecutar maniobras aéreas, así como un sensor infrarrojo ubicado en la parte inferior. Otros sensores, como el barómetro, giroscopio, acelerómetro, sensor de visión y sensores de posición están integrados en la tarjeta controladora interna. Además, incorpora una cámara frontal que, junto con el sistema de estabilización, permite mantener un vuelo controlado y estable. Esta cámara proporciona una plataforma adecuada para la captura continua de imágenes, lo cual es importante para el funcionamiento del algoritmo de seguimiento implementado internamente, ya que permite identificar y mantener al objeto en el centro del encuadre. Finalmente, su compatibilidad con el SDK facilita el envío de comandos de movimiento en respuesta a la posición detectada del objeto, que permite una interacción fluida entre el sistema de visión artificial y el dron.

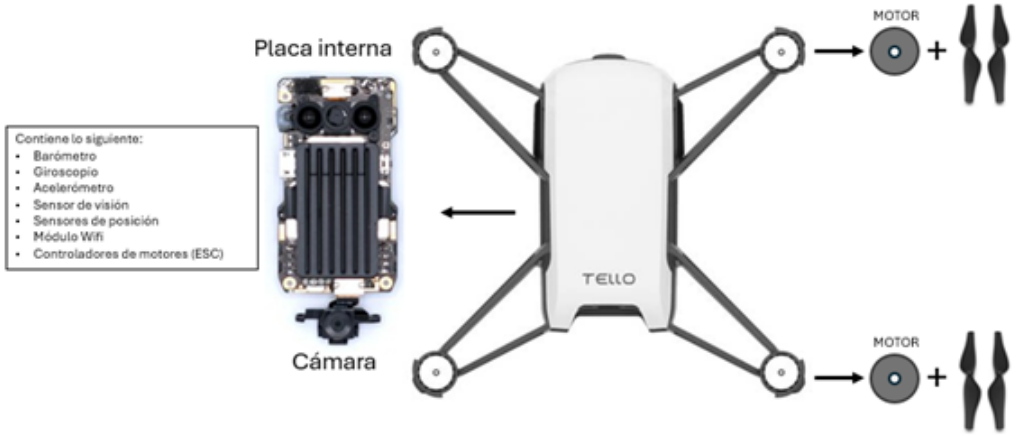


Figura 68: Estructura interna del dron Tello DJI. [Autoría propia]

2.3.1.1 Adaptación del sistema de detección y seguimiento al hardware del dron

Durante la implementación del sistema de seguimiento de objetos, fue fundamental considerar las características físicas del dron DJI Tello. Al tratarse de un equipo con cámara frontal fija y sin estabilización mecánica, el seguimiento del objeto no podía depender únicamente del software, sino también de la capacidad del dron para desplazarse con precisión y mantener al objetivo dentro del encuadre, como se muestra en la Figura 69. Esto llevó al desarrollo de un control de vuelo reactivo, capaz de responder en tiempo real a la posición detectada del objeto. La compatibilidad del dron con el SDK oficial facilitó esta comunicación directa entre el sistema de visión artificial y los movimientos del dron. No obstante, fue necesario adaptarse a ciertas limitaciones del hardware, tales como la autonomía reducida, la resolución media de la cámara y la ausencia de sensores de evasión de obstáculos, lo que obligó a definir cuidadosamente los entornos de prueba. Para mejorar la precisión del sistema, se incorporó una red neuronal convolucional, encargada de analizar las imágenes capturadas y realizar el reconocimiento de objetos de manera eficiente.

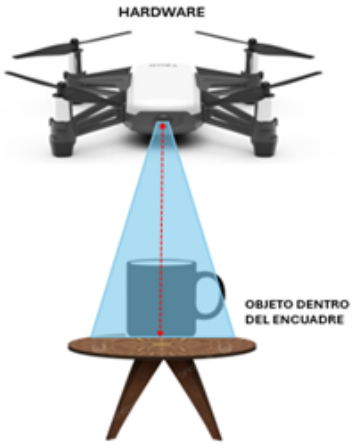


Figura 69: Detección del objeto dentro del encuadre. [Autoría propia]

2.3.1.2 Procesamiento externo

El sistema de detección y seguimiento visual se apoya en una arquitectura de hardware compuesta por el dron Tello DJI y una unidad de procesamiento externa, en este caso una computadora portátil. Ambos dispositivos se comunican mediante una red wifi, estableciendo un canal de transmisión de datos en tiempo real. El dron integra una cámara frontal fija encargada de capturar imágenes, las cuales son enviadas continuamente al equipo externo. Esta unidad, equipada con los recursos necesarios de cómputo, recibe y procesa la información visual para luego transmitir comandos de control de vuelo de regreso al dron. La cuál permite que la mayor carga de procesamiento sea asumida por el hardware externo, mientras que el dron se limita a las tareas de captura de imagen y ejecución de maniobras según las instrucciones recibidas. En la Figura 70 se muestra un esquema de sus interconexiones y el proceso.

Figura 70: Interconexiones del dron y el dispositivo de computo. [Autoría propia]

2.3.1.3 Calibraciones del dron Tello DJI

Las calibraciones del dron DJI Tello son fundamentales y deben realizarse antes de cada vuelo, especialmente cuando se incorpora un sistema inteligente de detección y seguimiento de objetos. Si estas calibraciones no se ejecutan correctamente, pueden presentarse comportamientos inestables o movimientos erróneos durante el vuelo, lo

cual afecta negativamente los resultados del sistema. Para este modelo de dron, se contemplan dos calibraciones principales, que permiten asegurar un funcionamiento estable y preciso en condiciones reales de operación. Estas son las siguientes:

Calibración del IMU

La calibración del IMU (Unidad de Medición Inercial) tiene como objetivo mejorar la estabilidad y la respuesta del dron durante el vuelo. Para realizar este procedimiento, el primer paso consiste en retirar todas las hélices del dron como medida de seguridad y estabilización de las posiciones del dron, luego se debe colocar la aeronave sobre una superficie completamente plana y nivelada. Desde la aplicación oficial de Tello para Android, se debe acceder al apartado de Configuraciones, donde se encuentra la opción de Calibración del IMU como se muestra en la Figura 71.

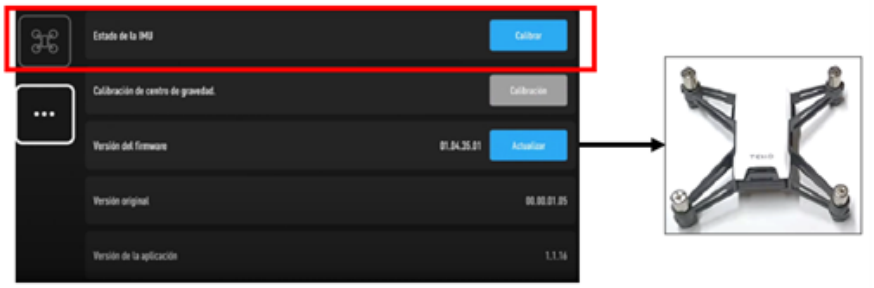


Figura 71: Calibración del IMU desde la aplicación.
[Autoría propia]

Una vez iniciada, la aplicación solicitará colocar el dron en seis posiciones diferentes, siguiendo una secuencia específica que se muestra en pantalla como se muestra en la Figura 72. Al completar correctamente cada posición, el indicador LED del dron comenzará a parpadear en color verde rápidamente, señalando que la calibración ha sido exitosa. Este proceso tiene una duración estimada de entre 5 a 10 minutos.

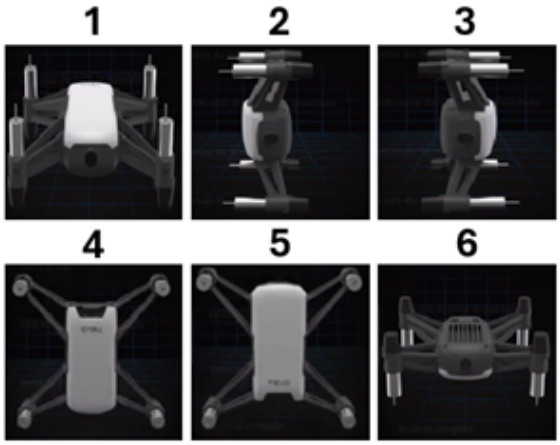


Figura 72: Posiciónes secuenciales del dron para la calibración.
[Autoría propia]

Calibración del centro de gravedad

La calibración del centro de gravedad consiste en ajustar o equilibrar físicamente la distribución del peso del dron para que esté correctamente centrado con respecto al punto de sustentación, que generalmente se encuentra en el centro entre los motores. Para realizar esta calibración es necesario que el dron tenga instaladas todas sus hélices, ya que el proceso se lleva a cabo durante el vuelo. Desde la aplicación Tello en dispositivos Android, se accede al apartado de Configuraciones, donde se encuentra la opción denominada Calibración del centro de gravedad.

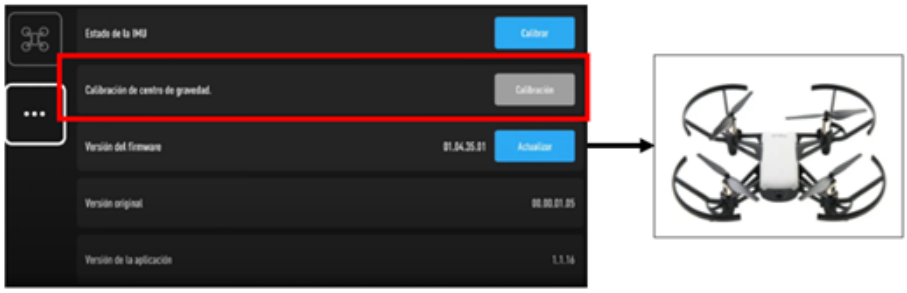


Figura 73: Calibración del centro de gravedad desde la aplicación.
[Autoría propia]

Una vez establecida la conexión entre el dron y el dispositivo Android, se da la orden de despegue, y con el dron en vuelo, se selecciona dicha opción. Al iniciar el proceso, el LED del dron cambiará a color morado, indicando que la calibración está en curso como se muestra en la Figura 74. Una vez completada, el LED cambiará a color verde, señalando que la calibración se ha realizado correctamente. Este procedimiento tiene una duración aproximada de 1 minuto.

Figura 74: Calibración del dron en curso.
[Autoría propia]

2.3.2. Diseño e implementación del software

En esta sección se analizará el modelo utilizado para la implementación del sistema, el cual será ejecutado en el entorno de desarrollo PyCharm en conjunto con el lenguaje de programación Python. Este entorno permitirá la visualización de datos adicionales en tiempo real. Cabe recalcar que, mediante una interfaz gráfica, se podrán observar parámetros relevantes para el monitoreo del dron. Además, esta interfaz también cumple una función de control, ya que desde ella se seleccionará el objeto que el dron deberá seguir. Para llevar a cabo la detección y el seguimiento de objetos, se implementará un algoritmo basado en redes neuronales convolucionales (CNN). A continuación, se revisará el modelo aplicado en este proyecto para comprender su funcionalidad y cómo contribuye al cumplimiento del objetivo planteado.

2.3.2.1 Modelo de detección YOLOv4

YOLO es el modelo que se estará utilizando en este proyecto. Es un algoritmo basado en redes neuronales convolucionales, y es uno de los modelos de detección más utilizados, ya que su algoritmo es avanzado y permite realizar detecciones precisas. Este modelo analiza la imagen dividiéndola en bloques, pero lo hace en una única pasada. Es decir, crea una cuadrícula sobre toda la imagen y analiza todas las celdas al mismo tiempo, lo que requiere menos cálculos y permite un procesamiento más rápido. Además, su arquitectura simplificada facilita la implementación en sistemas con recursos computacionales limitados. A continuación, se revisará en detalle el funcionamiento de este modelo de detección en el sistema, con el objetivo de comprender mejor sus principios y cómo será aplicado en el desarrollo del presente proyecto.

División en cuadrícula

El modelo Yolo divide la imagen de entrada en una cuadrícula, en este caso de 5×7 celdas. Cada celda es responsable de detectar los objetos cuyo centro se encuentre dentro de ella. Por ejemplo, si la cámara observa el entorno en donde se encuentra una botella sobre la mesa, y el centro de la botella está dentro de la celda (4,2) como se muestra en la Figura 75, será esa celda la encargada de hacer la predicción de detección y clasificación para ese objeto.

Figura 75: División en cuadrícula del entorno. [Autoría propia]

Predicción de cajas delimitadoras

Cada celda predice varias cajas delimitadoras con valores que indican la posición del centro de la caja (x, y), el ancho (w) y la altura (h) del objeto, todo en valores normalizados entre 0 y 1. También se incluye una puntuación de confianza que representa cuán probable es que la caja contenga un objeto y qué tan bien se ajusta esa caja al objeto real.

Donde:

- Centro de la caja: $(x, y) \in [0, 1]$ (relativo a la celda).
- Ancho y alto: $(w, h) \in [0, 1]$ (relativo al tamaño total de la imagen).
- Confianza: cuán probable es que haya un objeto y qué tan bien lo predice (IOU).

Por ejemplo, una celda puede predecir que hay una botella con el centro en medio de la celda, como se muestra en la Figura 76, con un tamaño del 10 % del ancho y 30 % del alto de la imagen, y una confianza de 0.85.

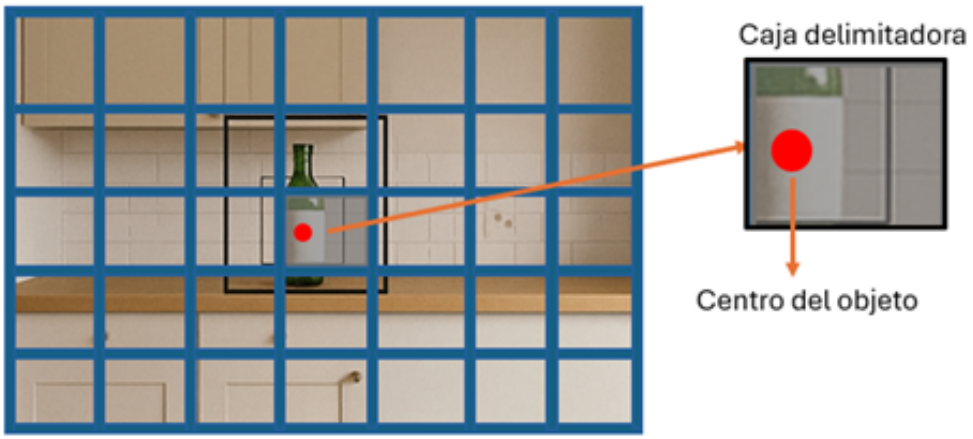


Figura 76: Predicción de un objeto con una caja delimitadora. [Autoría propia]

Predicción de clase del objeto

Cada celda también estima a qué clase pertenece el objeto. Esto se hace mediante un vector de probabilidades para cada clase posible, por ejemplo: botella, taza, plato. En la Figuran 77 se aprecia que la celda predice una probabilidad del 90 % de que el objeto detectado sea una botella, 5 % de que sea una taza y 5 % de que sea un plato, por lo que se toma *"botella"* como la clase con mayor probabilidad.

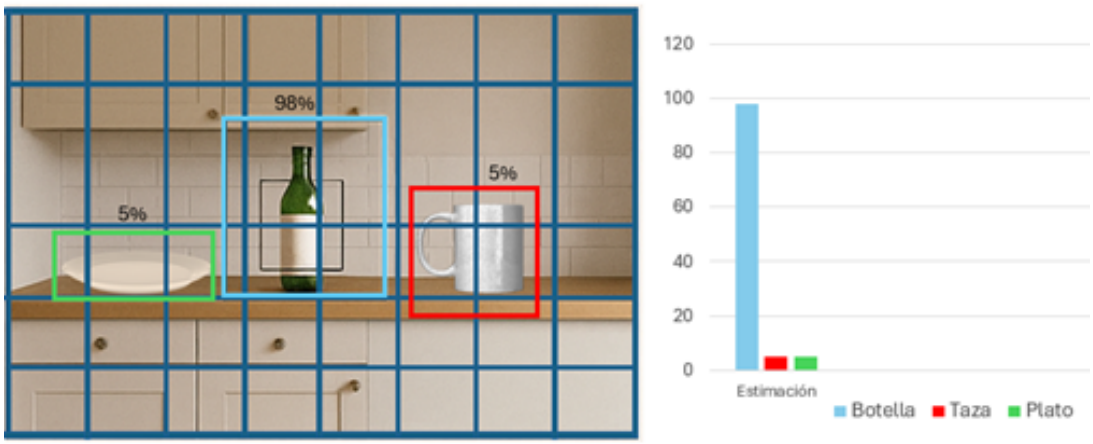


Figura 77: Predicción de clases de objetos. [Autoría propia]

Cálculo de la confianza final

La puntuación final para cada caja se obtiene multiplicando la confianza de que haya un objeto en la caja por la probabilidad de que el objeto sea de una clase específica. Por ejemplo, si la celda predice que hay un objeto con una confianza de 0.85 y que ese objeto tiene un 90 % de probabilidad de ser una botella, entonces la confianza final para "botella" será 0.765.

$$\text{Confianza final} = \text{Confianza(objeto)} \times P(\text{clase} \mid \text{objeto}) \tag{7}$$

$$0{,}85 \times 0{,}90 = 0{,}765 \quad \text{(para la clase “botella”)} \tag{8}$$

Cálculo del IOU (Intersección sobre unión)

El modelo compara la caja predicha con la caja real (verdad fundamental) mediante el IOU, que se calcula dividiendo el área de la intersección entre ambas cajas por el área total de su unión.

$$\text{IOU} = \frac{\text{Área}_{\text{intersección}}}{\text{Área}_{\text{unión}}} \tag{9}$$

O equivalentemente:

$$\text{IOU} = \frac{\text{Área}_{\text{intersección}}}{\text{Área}_{\text{predicción}} + \text{Área}_{\text{real}} - \text{Área}_{\text{intersección}}} \tag{10}$$

Por ejemplo, si la predicción y la caja real se superponen en 80 píxeles, y sus áreas individuales son 100 y 120 píxeles respectivamente, el IOU será aproximadamente 0.57. Cuanto mayor sea el IOU, mejor es la predicción.

$$\text{IOU} = \frac{80}{100 + 120 - 80} = \frac{80}{140} = 0{,}57 \tag{11}$$

Supresión de no máximos (NMS)

La NMS actúa como un filtro que, si detecta varios objetos en un solo punto o muy cercanos, se encarga de comparar sus niveles de confianza. Luego descarta las detecciones menos relevantes y conserva la que tiene mayor confianza, es decir, la más resaltante. Cuando hay múltiples cajas prediciendo el mismo objeto (por ejemplo, varias detectan una botella con diferentes tamaños y ubicaciones), YOLO aplica NMS para quedarse con la caja más confiable y eliminar las demás si se solapan mucho, es decir, si tienen un IOU alto.

$$\text{Si IOU}(A, B) > \text{umbral} \Rightarrow \text{Eliminar la de menor confianza} \tag{12}$$

Por ejemplo, en la Figura 78 se muestra que hay dos predicciones para una botella, una con confianza de 0.76 y otra de 0.6, y su IOU es mayor a un umbral de 0.5, por lo tanto se elimina la menos confiable.

$$\text{Si IOU}((0{,}76), (0{,}60)) > 0{,}5 \Rightarrow 0{,}65 \tag{13}$$

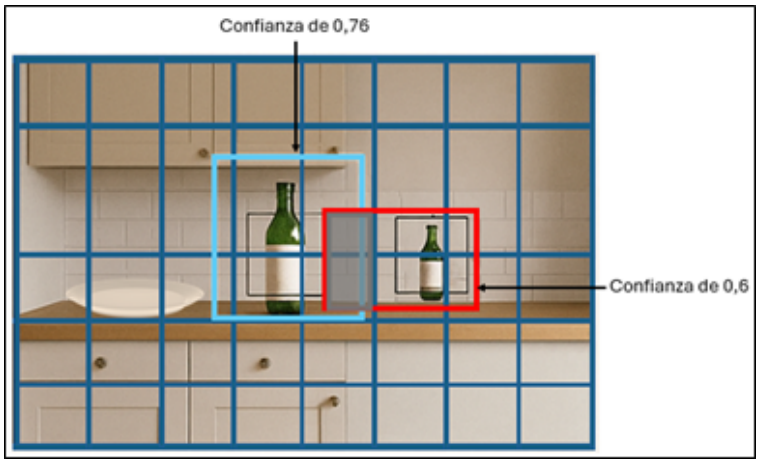


Figura 78: Filtración de dos objetos. [Autoría propia]

2.3.2.2 Configuraciones de los parámetros de la red Yolo

Para la configuración de la red yolo, se realizó un ajuste cuidadoso de múltiples parámetros con el objetivo de optimizar el rendimiento del sistema de detección tanto en el entorno computacional del dron como en la computadora utilizada durante el proceso. Estas modificaciones se enfocaron en lograr un equilibrio entre precisión, eficiencia computacional y estabilidad del proceso de entrenamiento. Los valores seleccionados responden a las características específicas del hardware disponible como la memoria de la GPU y a la necesidad de adaptar la red al tipo de imágenes captadas por el sistema embarcado. De esta forma, se buscó maximizar la capacidad de detección del modelo, reduciendo al mismo tiempo errores comunes como el sobreajuste, y facilitando la convergencia hacia una solución óptima. A continuación, se describen los principales parámetros configurados y su impacto en el desempeño de la red.

Tamaño del lote de entrenamiento (batch)

Este parámetro establece el número de imágenes que serán procesadas en cada iteración del entrenamiento. En este caso, la red procesará 64 imágenes simultáneamente en cada paso. Un mayor tamaño de batch puede estabilizar el entrenamiento, pero también incrementa el uso de memoria. Su elección depende de la capacidad de la GPU y la naturaleza del conjunto de datos.

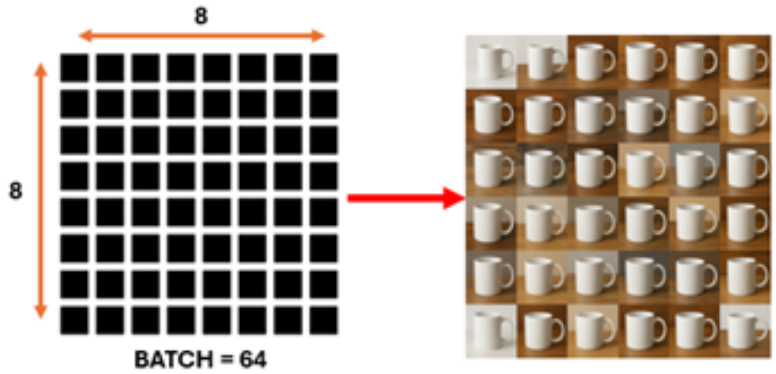


Figura 79: División en 64 cuadros. [Autoría propia]

El valor de 64 se dejó tal y como viene definido por la red original, ya que es un valor recomendado por la propia arquitectura de la red neuronal, y además se ajusta adecuadamente a las capacidades del sistema, el cual presenta limitaciones de rendimiento.

Subdivisión del batch (subdivisions)

Este parámetro divide el batch en partes más pequeñas para permitir el entrenamiento con menos memoria. Si subdivisions=1, entonces las 64 imágenes se procesan juntas. Si fuera 4, se dividiría el batch en cuatro partes de 16 imágenes cada una, procesadas secuencialmente y sumadas al final para actualizar los pesos. Esto es útil para entrenar en GPU con memoria limitada.

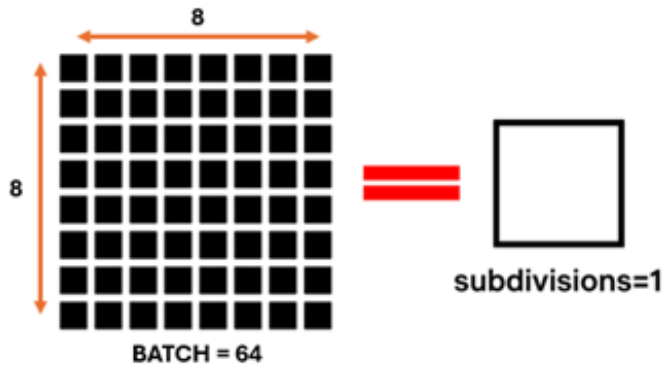


Figura 80: Procesamiento de 64 imágenes en un solo instante. [Autoría propia]

Se conservó el valor en 1, ya que como se observa en la Figura 80, las 64 imágenes pueden ser procesadas simultáneamente sin afectar el rendimiento, lo que permite una mayor rapidez en el entrenamiento del modelo.

Resolución de entrada (width, height)

Estos parámetros definen la resolución a la que se redimensionan todas las imágenes antes de ser introducidas a la red. Una resolución de 416x416 proporciona un equilibrio entre precisión y velocidad. Es importante que estos valores sean múltiplos de 32 para que la arquitectura convolucional funcione correctamente debido al tamaño de las capas subsiguientes.

Figura 81: Resolución de la imagen. [Autoría propia]

Se ajustó a 416x416, ya que este valor ofrece un buen rendimiento en el sistema actual de la computadora. Aumentar la resolución a valores más altos podría mejorar

levemente la precisión, pero generaría mayores exigencias de procesamiento, lo que podría causar dificultades de rendimiento o ralentización durante el entrenamiento y la inferencia.

Canales de color de entrada (channels)

Indica cuántos canales tiene cada imagen de entrada. Un valor de 3 representa imágenes en color RGB (Rojo, Verde, Azul). Si se trabajara con imágenes en blanco y negro, este valor sería 1. Es importante que coincida con el número de canales reales del conjunto de datos.

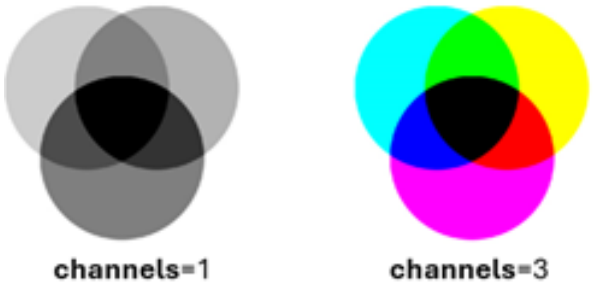


Figura 82: Canales de colores. [Autoría propia]

En este caso, se optó por utilizar 3 canales por obvias razones, ya que el sistema está orientado a realizar detecciones en ambientes reales, donde es necesario conservar la información del color original para una mejor precisión en la detección.

Regularización por decaimiento de pesos (decay)

Este término representa la penalización aplicada a los pesos grandes mediante regularización L2. Ayuda a evitar que el modelo se sobreentrene (overfitting) en el conjunto de entrenamiento. Se aplica en cada actualización de pesos como un freno que reduce gradualmente su magnitud.

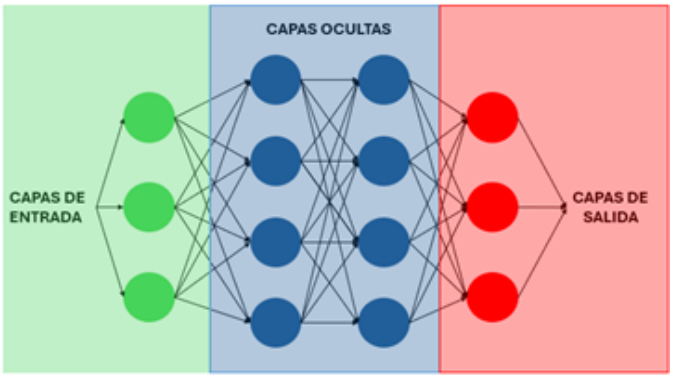


Figura 83: Red neuronal completa. [Autoría propia]

La penalización L2 (también llamada L2 Penalty) es una técnica de regularización que actúa sobre los pesos de una red neuronal, especialmente sobre los que son demasiado grandes. Un peso grande significa que una conexión entre neuronas tiene mucha influencia, lo que puede causar sobre entrenamiento (overfitting). Para evitar esto, se aplica un valor llamado decay (por ejemplo, decay = 0.0005), que reduce

ligeramente cada peso en cada actualización del entrenamiento. Así, la L2 Penalty actúa como un freno que evita que los pesos crezcan demasiado, ayudando al modelo a generalizar mejor.

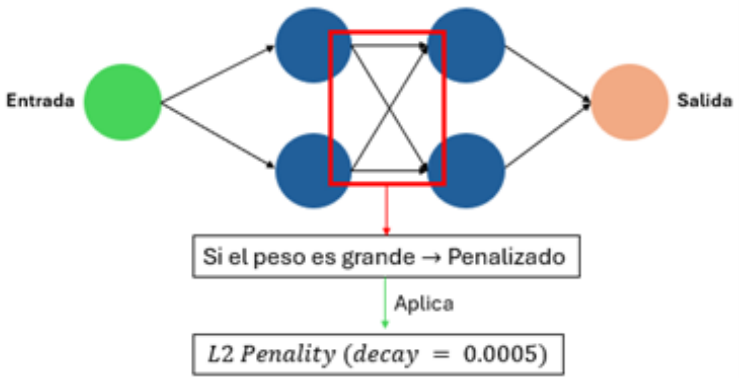


Figura 84: Penalización de los pesos de la red neuronal. [Autoría propia]

Por lo tanto, se optó por ajustar ese parámetro en decay = 0.0005 porque es un valor suficientemente pequeño como para no interferir en el aprendizaje, pero lo bastante significativo como para controlar el crecimiento de los pesos. Valores más altos podrían frenar demasiado el entrenamiento (underfitting), y valores más bajos no serían efectivos para evitar el sobreajuste.

Rotación aleatoria de imágenes (angle)

Establece el ángulo máximo de rotación que puede aplicarse aleatoriamente a las imágenes durante el entrenamiento. Un valor de 0 implica que no se aplicarán rotaciones. Este parámetro es parte del data augmentation, ayudando a generar ejemplos variados y mejorar la generalización si se activa.

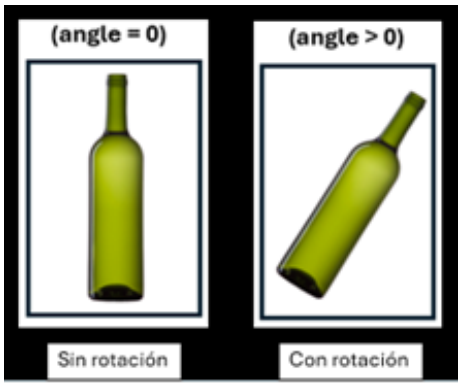


Figura 85: Rotación aleatoria. [Autoría propia]

Este parámetro establece el ángulo máximo de rotación que puede aplicarse aleatoriamente a las imágenes durante el entrenamiento. Un valor de 0 implica que no se aplicarán rotaciones. Es parte de la técnica de data augmentation, útil para generar ejemplos variados y mejorar la generalización del modelo si se activa.

Ajuste de saturación (saturation)

Controla cuánto se modifica la saturación de color de las imágenes durante el aumento de datos. Un valor de 1.5 aumenta la saturación un 50 %. Esto permite que la

red aprenda a detectar objetos en condiciones de iluminación o colores variables. El parámetro saturation controla la intensidad de los colores en las imágenes durante el entrenamiento; si su valor es 1.0, los colores permanecen normales, mientras que un valor mayor (como 1.5) incrementa la saturación haciendo los colores más vivos, y un valor menor (como 0.7) la reduce, generando colores más apagados o grisáceos. Este comportamiento está definido por el código fuente del framework de entrenamiento y ha sido documentado por los propios creadores del modelo, como parte de las técnicas de data augmentation.

Figura 86: Ajuste de saturación. [Autoría propia]

Por tal motivo, se ajustó a un valor de 1.5, ya que representa un aumento moderado (un 50 %) que mantiene un equilibrio, permitiendo que la red aprenda a reconocer objetos bajo distintas condiciones de iluminación y color sin afectar el rendimiento general del sistema.

Fase de calentamiento del aprendizaje (burn_in)

Durante las primeras 1000 iteraciones del entrenamiento, se incrementa gradualmente la tasa de aprendizaje desde cero hasta el valor normal. Este proceso se conoce como calentamiento (warm-up) y previene que los pesos del modelo cambien drásticamente al inicio del entrenamiento, mejorando la estabilidad inicial.

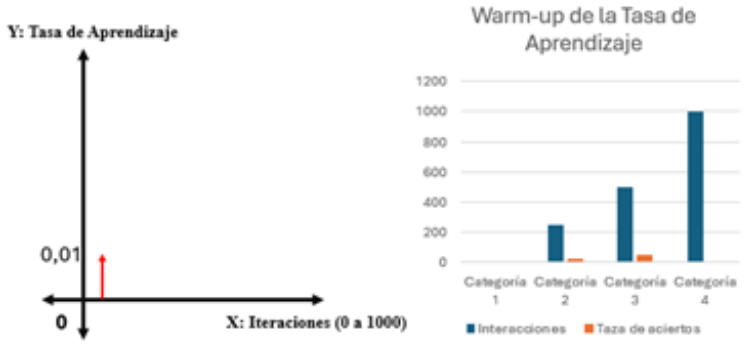


Figura 87: Fase del calentamiento del aprendizaje. [Autoría propia]

Durante la fase de calentamiento burn_in, la tasa de aprendizaje aumenta gradualmente desde cero hasta alcanzar su valor normal a lo largo de las primeras 1000 iteraciones. Esta gráfica ilustra ese proceso, mostrando cómo la tasa crece de forma progresiva, evitando que los pesos del modelo cambien bruscamente al inicio

del entrenamiento. Este enfoque mejora la estabilidad del aprendizaje en las etapas iniciales, facilitando una convergencia más controlada y eficiente del modelo. Por tal razón, este parámetro se fija en 1000, ya que proporciona un equilibrio adecuado entre estabilidad y velocidad al comenzar el proceso de entrenamiento.

Número máximo de iteraciones (max_batches)
El parámetro max_batches define el número total de iteraciones que se ejecutarán durante el entrenamiento de una red neuronal, y cuando se alcanza este número, el entrenamiento se detiene automáticamente. Cada batch es un grupo de imágenes que se procesan juntas antes de actualizar los pesos del modelo, por lo que max_batches indica cuántos de estos grupos pasarán por el modelo durante todo el entrenamiento.

Como regla práctica, este valor se calcula multiplicando el número de clases por 2000, asegurando que cada clase tenga suficiente representación y tiempo para que el modelo aprenda bien, incluso en casos de clases difíciles o desbalanceadas. Por ejemplo, para 5 clases, se recomiendan 10,000 iteraciones; sin embargo, en proyectos más complejos o con grandes volúmenes de datos, este número puede aumentarse (por ejemplo, clases $\times$ 4000), aunque con el costo de un tiempo de entrenamiento mayor. Por tal motivo, este parámetro se dejará intacto, ya que modificarlo podría generar cambios en el sistema que no convienen para el correcto funcionamiento del modelo.

2.3.2.3 Funcionamiento interno del sistema inteligente aplicado al dron

Se detallan los pasos del funcionamiento del sistema, desde el despegue del dron hasta su aterrizaje, luego de haber cumplido su objetivo: detectar y seguir los objetos de acuerdo con el objeto seleccionado. Este proceso se lleva a cabo de manera automatizada y ordenada, garantizando que el dron realice su misión de manera correcta. El sistema comienza con la activación del dron, seguido por la identificación del objeto de interés mediante visión artificial, y continúa con el seguimiento dinámico del mismo durante todo el recorrido. Finalmente, una vez alcanzado el objetivo o completada la ruta, el dron procede con su descenso controlado hasta aterrizar de forma segura.

En la Figura 88 se observa el diagrama de flujo de todo el sistema, el cual representa visualmente cada una de las etapas descritas anteriormente, facilitando así la comprensión global del funcionamiento del sistema desde una perspectiva lógica y secuencial.

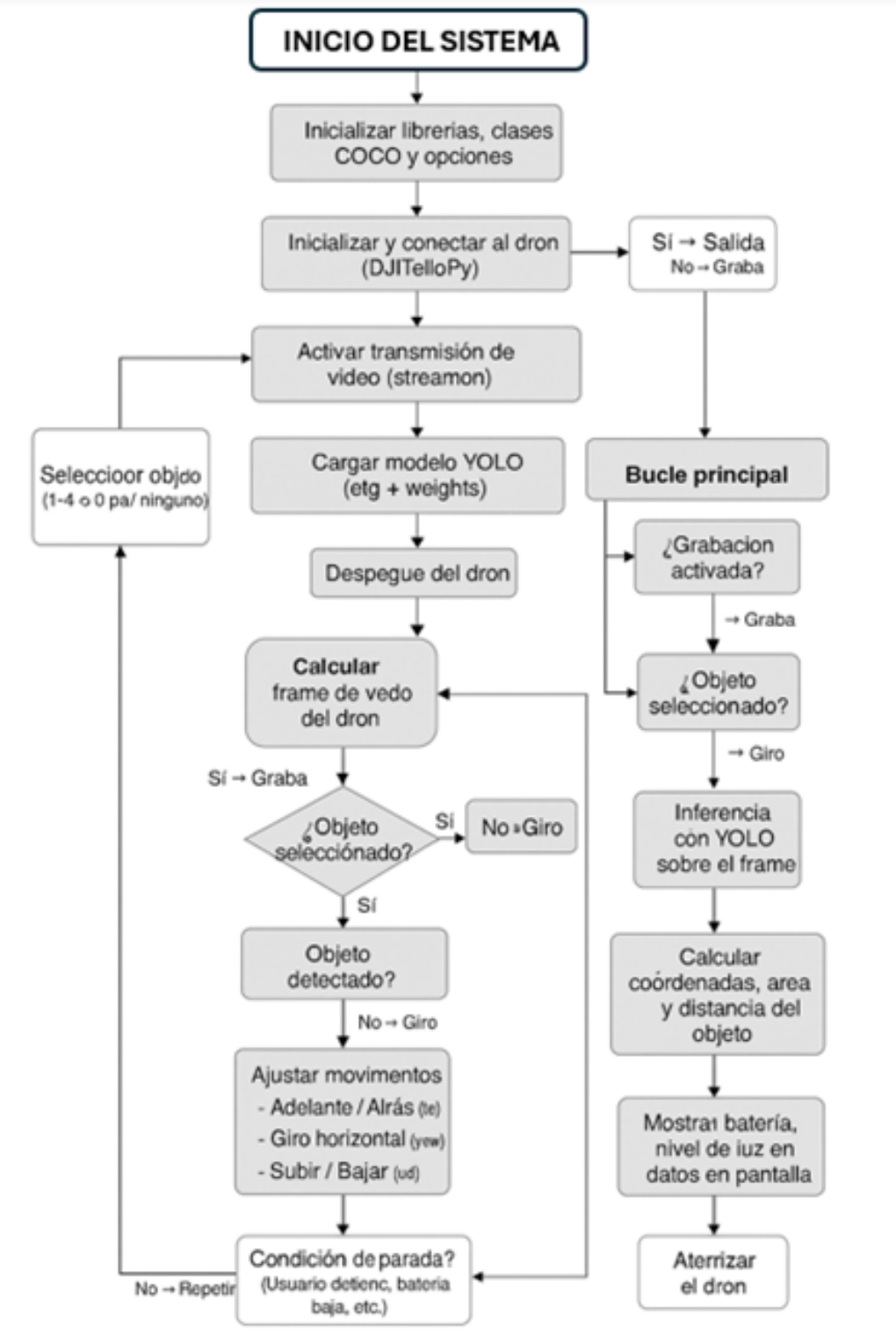


Figura 88: Diagrama de flujo del sistema.
[Autoría propia]

A. **Inicio del sistema y conexión con el dron**

Antes de comenzar cualquier operación, la computadora debe conectarse al dron Tello a través de la red wifi que emite el dron . Una vez establecida esta conexión, el algoritmo inicia el proceso verificando que todo esté en condiciones. Se asegura de que la batería tenga al menos un 20 % para evitar interrupciones durante el vuelo. Una vez confirmado esto, se activa el modo de transmisión de video en tiempo real, lo que permite trabajar con las imágenes capturadas por el dron. Cuando todo está listo, el sistema queda a la espera de que el usuario indique el momento de iniciar el vuelo. Para ello, debe escribir un comando desde el teclado, lo cual es importante para evitar que el dron despegue sin que el usuario esté preparado. Una vez que se da la orden, el dron despega y comienza la operación de detección y seguimiento.

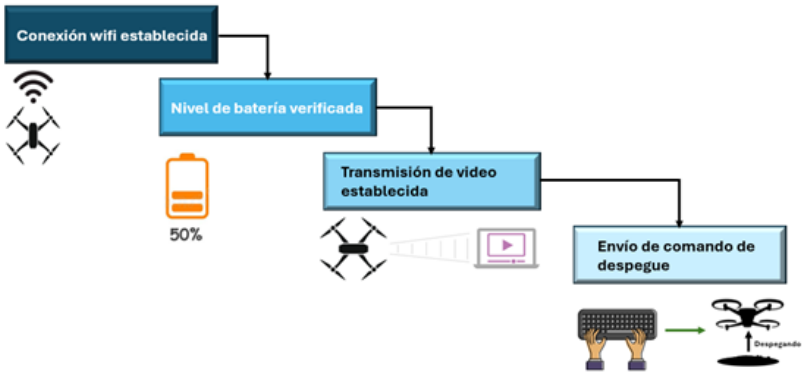


Figura 89: Esquema de inicio del sistema. [Autoría propia]

B. **Carga del modelo de detección (YOLO)**

Una vez que el dron ha despegado, el sistema procede a cargar el modelo de inteligencia artificial YOLOv4, el cual el modelo ya viene pre-entrenado para detectar objetos comunes utilizando una base de datos. Se cargan tanto el archivo de pesos como la configuración del modelo. Gracias a este modelo, el dron es capaz de interpretar lo que observa, es decir, reconocer si hay una botella, un celular, entre otros objetos. Es como dotarlo de visión e inteligencia visual. Para lograr esto, se utilizan tres archivos esenciales:

- **Archivo de configuración** (`yolov4.cfg`)**:** Define la arquitectura de la red neuronal, especificando las capas, tipos de capas, conexiones, filtros y parámetros necesarios para construir el modelo.
- **Archivo de pesos** (`yolov4.weights`)**:** Contiene los valores aprendidos durante el entrenamiento del modelo. Estos pesos, cargados junto con la configuración, permiten al modelo reconocer patrones visuales y realizar detecciones efectivas.
- **Archivo de nombres de clases** (`coco.names`)**:** Es un archivo de texto que contiene las 80 clases del conjunto de datos COCO. Dentro de este archivo se encuentran diferentes objetos de toda clase, y sirve para asignar un nombre legible al objeto detectado.

Los tres archivos han sido correctamente cargados al sistema, ya que cada uno cumple una función específica y son interdependientes. En conjunto, estos archivos conforman el modelo YOLO, específicamente la versión YOLOv4. Con estos archivos, el modelo está completamente configurado y listo para dar el siguiente paso en el proceso de detección de objetos. En la Figura 90 se ilustra el proceso de carga del modelo.

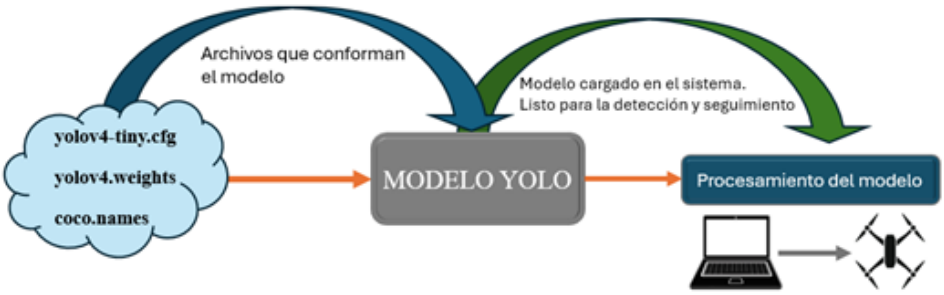


Figura 90: Proceso de carga del modelo yolov4. [Autoría propia]

C. Interfaz gráfica

El sistema desarrollado con OpenCV, muestra una interfaz gráfica con un menú donde se puede elegir qué objetos se desea que el dron siga. Se puede presionar diferentes teclas del 1 al 4 para seleccionar entre los objetos disponibles (botella, taza, tijera o celular). También tiene la opción de desactivar la detección con la tecla 0. Esta parte es importante porque permite que el usuario tenga el control total de la misión del dron, eligiendo qué objeto debe buscar y seguir. Además, en la misma interfaz se muestra información relevante como el nivel de batería del dron, la intensidad de la luz ambiente y el objeto actualmente seleccionado, como se observa en la figura 91.

Figura 91: Interfaz gráfica del sistema. [Autoría propia]

Selección del objeto desde el menú gráfico

Una vez que se se ejecuta el programa de inmediato aparece el menú de opciones desde OpenCV, se procede a seleccionar el objeto de interés; en este caso, una botella. Para ello, se presiona la tecla 1, lo que activa de inmediato el sistema para enfocarse exclusivamente en la detección de dicho objeto.

Al realizar la selección, el botón correspondiente se ilumina, indicando que la botella ha sido elegida correctamente. Además, en la pantalla se muestra un mensaje confirmando que la botella ha sido seleccionada. La Figura 92 evidencia este proceso, mostrando la detección exitosa y el correcto funcionamiento del sistema.

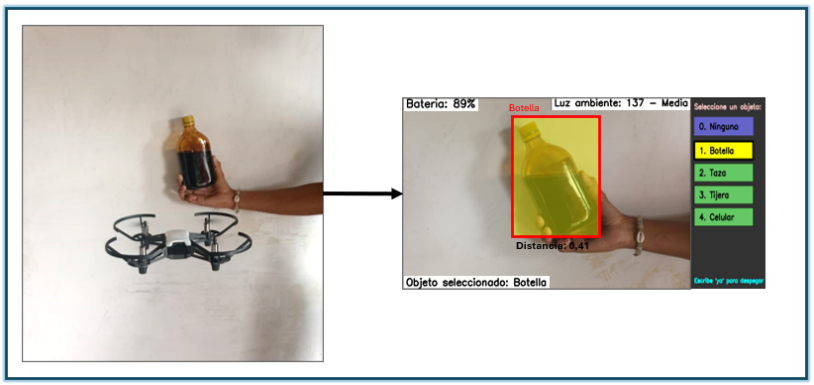


Figura 92: Selección del objeto, botella. [Autoría propia]

De igual manera, si se escoge la opción correspondiente a la taza, se presiona el número asignado en este caso, el número 2, lo que activa de inmediato el cambio de configuración para que el sistema se enfoque exclusivamente en la detección de tazas. Al igual que con el caso anterior, el botón se ilumina y se muestra en pantalla el mensaje correspondiente que confirma la selección.

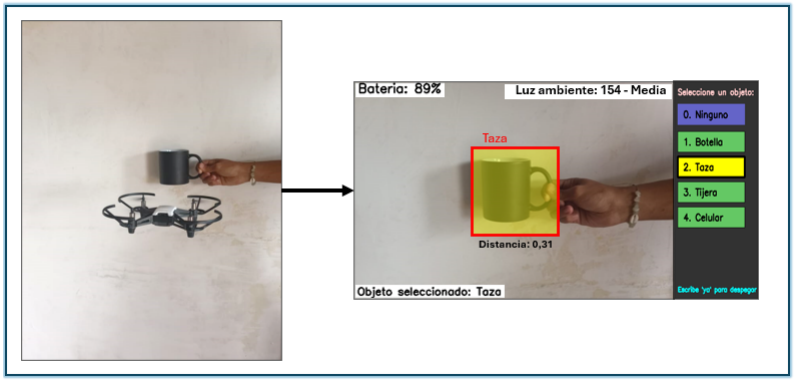


Figura 93: Selección del objeto, taza. [Autoría propia]

Si se selecciona el número 3, en este caso se iluminará el cuadro del botón correspondiente a la tijera, el cual se actualiza para que el sistema se enfoque exclusivamente en la detección de ese objeto.

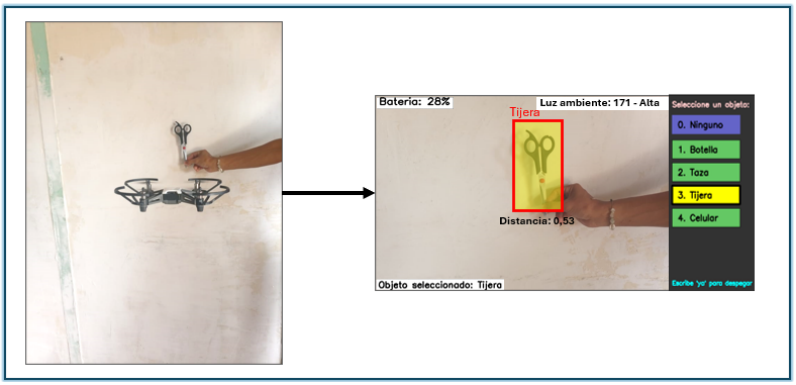


Figura 94: Selección del objeto, tijera. [Autoría propia]

Finalmente, si se presiona la tecla 4, el objeto ahora seleccionado será el celular, por lo que el sistema se actualizará para enfocarse exclusivamente en la detección de dicho elemento.

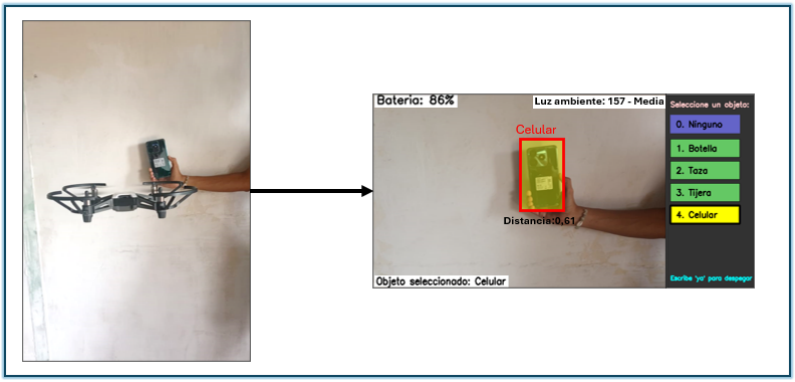


Figura 95: Selección del objeto, celular. [Autoría propia]

D. Detección y seguimiento del objeto

Cuando el modelo YOLO detecta un objeto en la imagen captada por el dron Tello, genera un cuadro delimitador que indica la posición y el tamaño del objeto dentro del video. Este cuadro proporciona coordenadas horizontales (x) y verticales (y), además de su ancho y alto. Con estos datos, el programa calcula el centro del objeto detectado y lo compara con el centro de la imagen captada por la cámara del dron, cuya resolución es de 960×720 píxeles. Por lo tanto, el punto medio de la imagen se encuentra en las coordenadas $(416, 416)$. Si la posición del objeto está más a la izquierda o derecha del centro, el dron gira hacia ese lado (yaw) para centrarlo. Si está más arriba o abajo, el dron sube o baja para alinear la altura.

Cálculos del Seguimiento

El sistema de seguimiento está diseñado para mantener un objeto centrado constantemente en la imagen capturada por la cámara del dron. Esto requiere que, ante cualquier movimiento del objeto, el dron también se desplace para conservarlo en el centro del encuadre. Para lograrlo, se aplican controles de corrección de posición que permiten al dron moverse de forma precisa en

un espacio tridimensional. El dron se desplaza libremente a lo largo de los tres ejes espaciales: X, Y y Z. El eje X corresponde al movimiento horizontal (izquierda-derecha), el eje Y al movimiento vertical (subida-bajada), y el eje Z al movimiento frontal (acercamiento o alejamiento). Esta libertad de movimiento le permite ajustar su posición en múltiples direcciones al mismo tiempo.

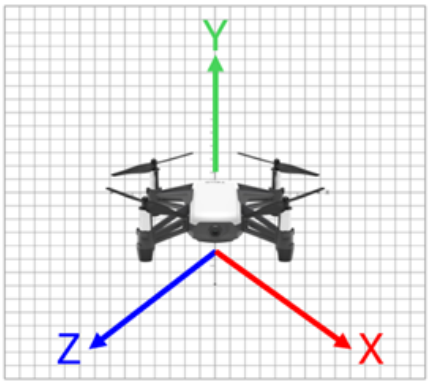


Figura 96: coordenadas X, Y y Z.
[Autoría propia]

Estas correcciones se realizan en tiempo real, gracias al procesamiento de las imágenes y la detección de la posición del objeto en cada cuadro. Así, el dron puede seguir al objetivo de forma estable, incluso cuando este cambia de dirección, velocidad o altura repentinamente. A continuación, se mostrará más a detalle cómo se ejecutan estas maniobras de corrección en cada uno de los ejes, de forma individual y combinada.

Movimientos rotacionales

Los movimientos rotacionales permiten que el dron gire sobre su propio eje vertical para alinear su vista con el objeto detectado. En el código, este tipo de movimiento se controla con la variable *yaw*, la cual se ajusta en función del error horizontal (*error_x*) entre el centro del objeto y el centro del encuadre de la cámara. Si el objeto se encuentra a la izquierda o a la derecha del centro de la imagen, el dron rota en la dirección correspondiente para centrarlo. Esta rotación lateral es esencial para que el objeto permanezca dentro del campo de visión frontal del dron mientras lo sigue. A continuación, se mostrará con mayor detalle cómo el dron ejecuta este tipo de rotación para mantener siempre centrado el objeto en su campo visual.

Movimientos rotacional derecha

El valor lr se traduce como izquierda y derecha se obtiene como:

$$\texttt{lr} = \texttt{clip}(\texttt{error\_x} \times 0{,}1,\ -40,\ 40) \tag{14}$$

Esto significa que, si el objeto está hacia la derecha del centro ($\texttt{error}_x > 0$), el dron rotará hacia la derecha para recentrarlo, como se muestra en la Figura 97.

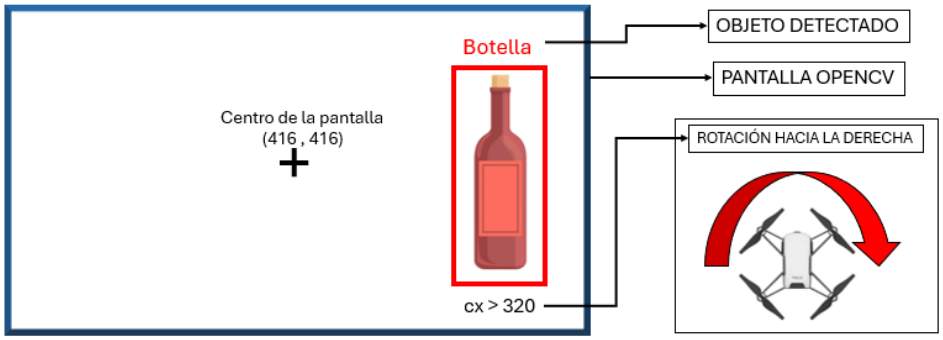


Figura 97: Objeto detectado en la parte lateral izquierda de la pantalla. [Autoría propia]

Movimientos rotacional izquierda

Si el valor del error horizontal es negativo ($\text{error}_x < 0$), el dron rotará hacia la izquierda. El valor de error_x se escala por 0,1 para suavizar el movimiento y se limita dentro del rango $[-40, 40]$.

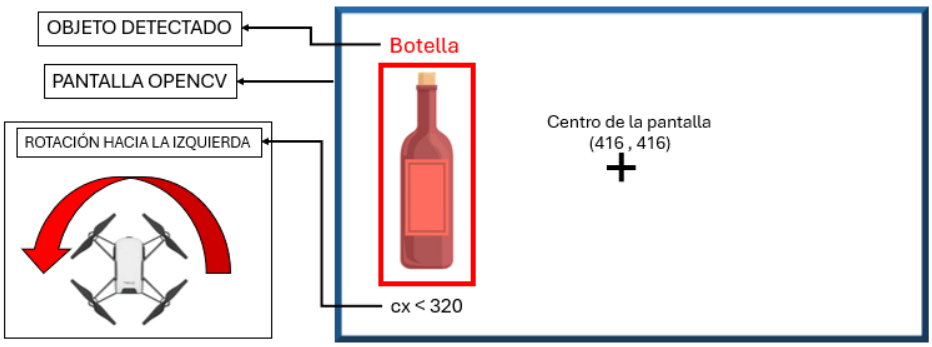


Figura 98: Objeto detectado en la parte lateral derecha de la pantalla. [Autoría propia]

Movimientos verticales

Los movimientos verticales permiten que el dron se desplace hacia arriba o hacia abajo para alinear su altura con la del objeto detectado. Este ajuste se calcula utilizando el error vertical (*error_y*), que es la diferencia entre el centro del objeto y el centro vertical de la imagen. Si el objeto aparece más arriba o abajo de lo esperado, el dron subirá o bajará respectivamente para mantenerlo centrado en el plano vertical. Este movimiento mejora la capacidad del dron para seguir objetos. En los siguientes apartados, se explicará con mayor precisión cómo se implementan estos desplazamientos verticales para mantener la alineación correcta con el objetivo.

Movimientos vertical arriba

El valor *ud*, que se traduce como arriba y abajo, se calcula como:

$$\text{ud} = \text{clip}(-\text{error}_y \times 0{,}15,\ -40,\ 40) \tag{15}$$

Aquí, el signo negativo invierte el eje Y, ya que, en la imagen, Y crece hacia

abajo, de modo que, si el objeto está más alto que el centro ($\text{error}_y < 0$), el dron subirá, como se muestra en la Figura 99.

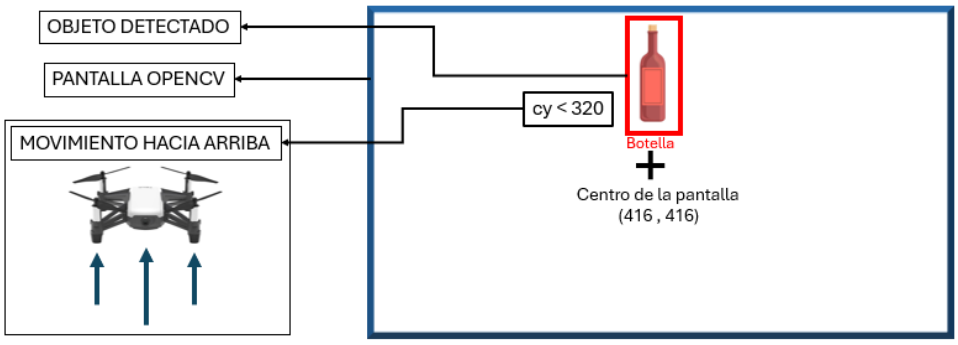


Figura 99: Objeto detectado en la parte superior de la pantalla.
[Autoría propia]

Movimientos vertical abajo

Si el objeto está más abajo ($\text{error}_y > 0$), el dron bajará, como se muestra en la Figura 100. El coeficiente 0,15 ajusta la sensibilidad del movimiento vertical.

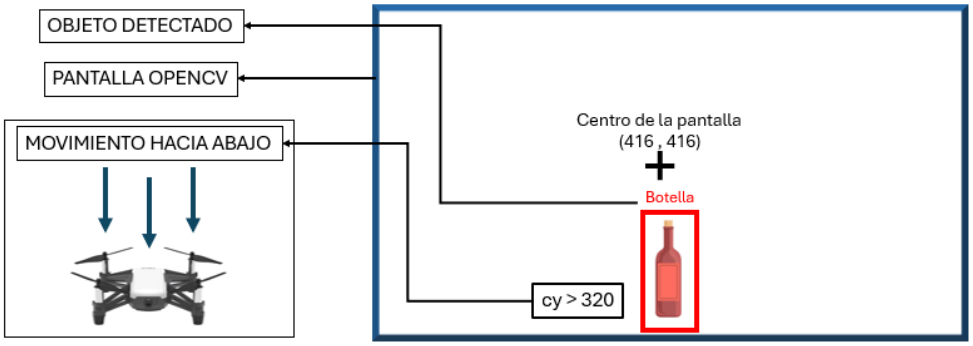


Figura 100: Objeto detectado en la parte inferior de la pantalla.
[Autoría propia]

Cuando el dron detecta un objeto, calcula su área utilizando la fórmula:

$$\text{área} = w \times h \tag{16}$$

donde w es el ancho del objeto y h es su alto, ambos medidos en píxeles. Esta área representa cuán grande se ve el objeto desde la perspectiva de la cámara. Luego, el dron compara esta área con un valor de área ideal, definido empíricamente como:

$$\text{área}_{\text{ideal}} = \frac{w \times h}{7} \tag{17}$$

Esta fórmula establece que el objeto debe ocupar aproximadamente una séptima parte del área total de la imagen, lo cual indica que está a una distancia visual óptima para el seguimiento. Este valor puede ser ajustado según el comportamiento deseado.

A continuación, se calcula el error de área:

$$\text{error}_{\text{área}} = \text{área} - \text{área}_{\text{ideal}} \tag{18}$$

Movimientos frontales

Los movimientos hacia adelante y hacia atrás permiten que el dron mantenga una distancia constante con el objeto detectado. En el código, esta distancia se estima en función del ancho del objeto en la imagen (mayor ancho implica menor distancia real). Si el objeto está demasiado lejos, el dron se acerca (*fb* positivo); si está demasiado cerca, se aleja (*fb* negativo). Esta lógica asegura que el dron mantenga una posición ideal para observar al objeto sin colisionar ni perderlo de vista. Además, el área del objeto también se considera para un ajuste más fino en la proximidad del dron al objetivo. A continuación, se detallará cómo se controlan estos desplazamientos frontales para garantizar una distancia óptima entre el dron y el objeto durante todo el seguimiento.

Movimiento frontal hacia adelante

Cuando el dron detecta un objeto y la distancia estimada a este objeto es mayor a la distancia deseada (50 cm), el dron realiza un movimiento hacia adelante para acercarse.

Si distancia > 50 cm $\Rightarrow$ fb positivo $\Rightarrow$ el dron avanza hacia el objeto.

Esta distancia se estima con una ecuación proporcional basada en el ancho del objeto detectado en píxeles:

$$\text{Distancia estimada} = \frac{8000}{w} \tag{19}$$

Donde:

- w es el ancho del objeto detectado en la imagen (en píxeles),
- el valor 8000 es un factor empírico ajustado según pruebas reales de calibración.

Luego se calcula el error de distancia:

$$\text{error}_{\text{distancia}} = \text{Distancia estimada} - \text{Distancia deseada} \tag{20}$$

Si este error es mayor que 5 cm, el dron se moverá hacia adelante, con una velocidad de avance proporcional a dicho error:

$$fb = \text{clip}(1{,}5 \times \text{error}_{\text{distancia}},\ -40,\ 40) \tag{21}$$

La función `clip` restringe el valor resultante del cálculo entre -40 y 40, lo cual corresponde a las limitaciones físicas de velocidad del dron, asegurando que no se envíen comandos que excedan su capacidad de respuesta. Esta limitación es especialmente importante para evitar movimientos bruscos o inestables que puedan comprometer la seguridad del vuelo o la precisión del seguimiento del

objeto.

El valor resultante se asigna al canal `fb`, que representa el eje de movimiento frontal del dron (forward/backward). Un valor positivo de `fb` indica un movimiento hacia adelante como se muestra en la Figura 101. Este canal es uno de los cuatro parámetros de velocidad que se envían al dron a través del comando `tello.send_rc_control`.

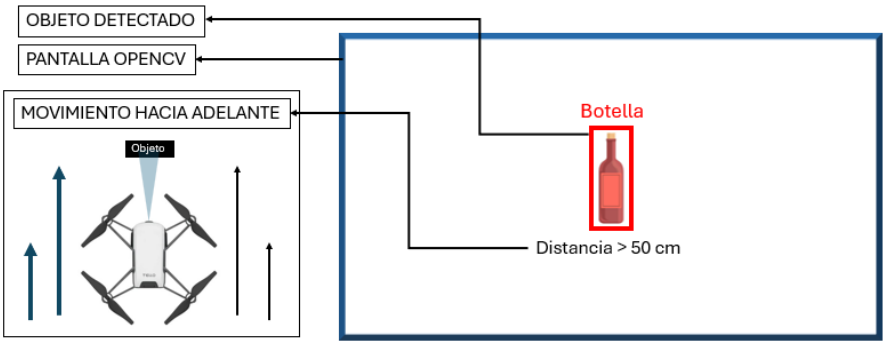


Figura 101: Objeto detectado lejos de la cámara del dron.
[Autoría propia]

El uso de valores escalados y limitados mediante `clip` garantiza que el dron se mueva con suavidad y control, ajustando dinámicamente su posición en función del error medido entre la posición actual del objeto y su posición deseada. Esta técnica forma parte de una estrategia de control proporcional simple pero efectiva, que puede ser ajustada para distintas condiciones ambientales o necesidades específicas del sistema de seguimiento visual.

Movimiento frontal hacia atrás

Por otro lado, si la distancia estimada al objeto es menor que 50 cm, significa que el dron está demasiado cerca y debe retroceder para mantener una distancia segura.

Si distancia < 45 cm $\Rightarrow$ fb negativo $\Rightarrow$ el dron retrocede.

En este caso, se aplica el mismo cálculo de error que en el caso anterior:

$$\text{error}_{\text{distancia}} = \text{Distancia estimada} - 50 \tag{22}$$

Al ser el error negativo, la velocidad resultante de avance (fb) también será negativa. Este valor negativo le indica al dron que debe alejarse del objeto, con una velocidad proporcional a la magnitud del error. De esta forma, el dron frena automáticamente a medida que se acerca nuevamente al objetivo de 50 cm de distancia.

$$fb = \text{clip}(1{,}5 \times \text{error}_{\text{distancia}},\ -40,\ 40) \tag{23}$$

Estos movimientos ocurren de forma dinámica y continua, ajustando la velocidad en tiempo real según la distancia medida. El sistema sigue regulando el

avance o retroceso hasta que el dron se mantenga dentro del rango deseado, que es de ±5 cm respecto a la distancia ideal de 68 cm.

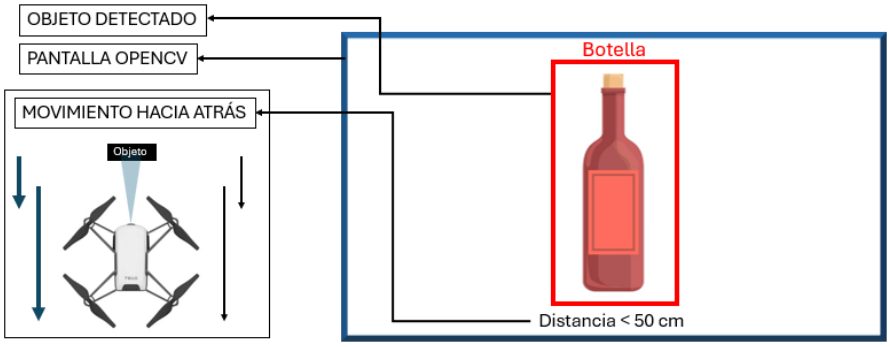


Figura 102: Objeto detectado cerca de la cámara del dron.
[Autoría propia]

E. **Finalización del sistema**

El sistema de detección finaliza solo cuando el usuario lo indica a través del teclado. Al presionar la tecla 'l', el dron aterriza y se interrumpe la ejecución del ciclo principal. De forma similar, si se presiona la tecla Esc, el programa también sale del ciclo sin necesidad de un aterrizaje previo. En ambos casos, una vez que se sale del bucle while, el sistema ejecuta los comandos necesarios para detener la grabación de video (si estaba activa), apagar la transmisión de video en tiempo real desde el dron, cerrar todas las ventanas de OpenCV que estaban abiertas y liberar todos los recursos utilizados, incluyendo el modelo de detección y objetos de captura. Finalmente, se termina la conexión con el dron, concluyendo así por completo el proceso de detección y seguimiento.

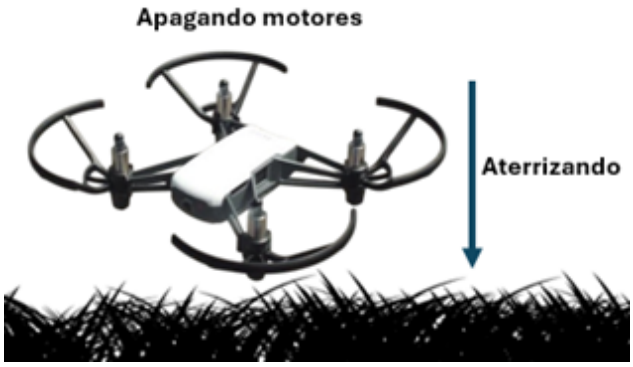


Figura 103: Aterrizaje del dron.
[Autoría propia]

Liberación de recursos

Al concluir la ejecución del sistema, uno de los primeros recursos que se libera es el flujo de video del dron. Esto se realiza mediante un comando que detiene la transmisión de imágenes desde la cámara del dron hacia el programa. Esta acción es esencial, ya que permite que el dron deje de enviar datos de video que ya no son necesarios, lo cual ayuda a conservar la batería y a prevenir posibles conflictos si se desea reiniciar el programa o utilizar el dron con otra aplicación.

Sin esta liberación explícita, el módulo de cámara del dron podría continuar trabajando de forma innecesaria. Otro recurso importante que debe liberarse es el objeto de captura de video, en caso de que se haya utilizado una fuente como una cámara USB o la cámara del propio sistema. Este recurso se libera con una instrucción que cierra el acceso a dicho dispositivo. Al no liberar esta captura, el sistema operativo podría seguir considerándola ocupada, lo que bloquearía su uso por otros programas o futuras ejecuciones del mismo sistema.

Asimismo, el programa se encarga de cerrar todas las ventanas creadas mediante la biblioteca OpenCV. Esto se lleva a cabo con una función que destruye cualquier interfaz visual abierta por el sistema, como las que muestran el video con los objetos detectados. Si estas ventanas no se cierran adecuadamente, podrían quedar activas en segundo plano, generando una mala experiencia de usuario o incluso impidiendo que el programa termine correctamente. Finalmente, el sistema libera la conexión establecida con el dron. Esta desconexión formal garantiza que no haya canales de comunicación abiertos una vez que el programa ha finalizado. Además, al liberar esta conexión, se prepara al dron para una nueva sesión de control o programación, evitando errores como múltiples intentos de conexión simultánea o bloqueos del canal de comunicación.

3. Pruebas y Resultados

En esta sección se presentan los resultados obtenidos a partir de las pruebas realizadas para la detección y seguimiento de cuatro objetos distintos, los cuales se muestran en el menú de opciones. Dichas pruebas fueron ejecutadas utilizando el entorno de desarrollo integrado PyCharm. Asimismo, se incluye la visualización de la interfaz gráfica implementada, la cual permite observar en tiempo real el comportamiento del dron y verificar de manera eficiente el seguimiento de los objetos.

3.1. Resultados

Durante las pruebas, se realizaron tres ensayos por cada tipo de objeto, con el fin de evaluar la consistencia y precisión del sistema en distintos escenarios y condiciones. Se verificó la capacidad del dron para identificar el objeto seleccionado dentro del campo visual, ajustar su orientación y desplazamiento en función de la posición y tamaño del objeto detectado, y mantener un seguimiento dinámico del mismo.

Durante las pruebas se evaluaron dos aspectos importantes: en primer lugar, la distancia a la que un objeto puede ser detectado de manera más efectiva por el sistema y, en segundo lugar, la influencia de la forma del objeto en la detección. Para esto, se utilizaron objetos con distintas geometrías con el fin de determinar cuál de ellas es reconocida con mayor precisión y consistencia. Estos ensayos permitieron identificar qué formas facilitan una mejor respuesta del sistema de detección, optimizando así su desempeño en distintas condiciones.

3.1.1. Evaluación del rango de detección del sistema

Para la evaluación del sistema de detección de objetos implementado, se considerarán cuatro objetos de tamaño normal, con colores oscuros y proporciones claramente reconocibles, de manera que puedan ser identificados visualmente como representaciones válidas de dichos objetos. El objetivo es que el sistema los detecte correctamente bajo condiciones realistas. A partir de esto, se utilizarán criterios de análisis que permitan examinar tanto el rendimiento del modelo como su comportamiento en escenarios reales. En primer lugar, se evaluarán tres diferentes distancias; cada valor de distancia será configurado en el sistema y se analizará la diferencia entre la distancia deseada y la distancia real alcanzada por el dron, con el fin de determinar si este avanzó correctamente hacia el objetivo o presentó un margen de error. Asimismo, se registrará el tiempo de detección, es decir, el intervalo desde que el sistema se activa (cuando el dron inicia la búsqueda del objeto) hasta que logra detectar dicho objeto. Este tiempo será considerado en los análisis de eficiencia. También se medirá el tiempo de seguimiento, que corresponde al período durante el cual el dron mantiene el seguimiento del objeto detectado, desde el inicio de la detección hasta que deja de seguirlo.

3.1.1.1 Evaluación de distancia para una botella

Para esta primera evaluación, se seleccionó una botella de tamaño estándar como objeto de detección, con un color que contraste con el entorno y bajo condiciones de iluminación ambiental controlada.

Figura 104: Análisis de distancia de una botella.

En la Tabla 9 se presentan los resultados correspondientes a los diferentes rangos de distancia utilizados para el análisis de la botella.

Tabla 9: Parámetros evaluados en diferentes distancias

Dist.	**Val. (cm)**	**Real (cm)**	**Error (%)**	**T. Det. (s)**	**T. Seg. (s)**
1	50	50.21	0.21	5.02	30.20
2	100	100.89	0.89	5.02	32.23
3	150	151.10	1.10	6.13	28.86

En la primera prueba, el sistema fue configurado para que el dron mantuviera una distancia deseada de 50 cm respecto al objeto a seguir. El dron logró acercarse con buena precisión, alcanzando una distancia real de 50,21 cm, lo que representa un margen de error de 0,21 cm. Aunque la desviación es pequeña, sigue indicando una diferencia con respecto a la distancia establecida. El tiempo de detección, es decir, desde el despegue hasta la identificación de la botella fue de 5,02 segundos. En cuanto al seguimiento, el dron logró mantener el rastreo del objeto durante 30,20 segundos, lo cual representa un desempeño favorable en esta fase inicial.

En la segunda prueba, se ajustó la distancia deseada a 100 cm. El dron alcanzó una distancia real de 100,89 cm, aumentando el error a 0,89 cm. Este incremento representa una mayor desviación en comparación con la primera prueba. A pesar de ello, el tiempo de detección se mantuvo constante en 5,02 segundos. El tiempo de seguimiento, por otro lado, fue mayor, alcanzando los 32,23 segundos, lo que sugiere que el dron pudo mantener el rastreo de forma más prolongada.

En la tercera prueba, la distancia deseada se estableció en 150 cm. El dron se aproximó hasta los 151,10 cm, registrando un error de 1,10 cm, el más elevado entre las tres pruebas, lo cual indica una disminución en la precisión con respecto a las configuraciones anteriores. Además, el tiempo de detección aumentó a 6,13 segundos, superando en más de un segundo a las pruebas previas. En contraste, el tiempo de seguimiento se redujo a 28,86 segundos, lo que evidencia una menor duración en la capacidad de rastreo.

3.1.1.2 Evaluación de distancia para una taza

En esta segunda prueba se eligió una taza de dimensiones comunes como objeto a detectar. La taza presenta un color que facilita su diferenciación respecto al fondo,

y la evaluación se realizó bajo una iluminación ambiental regulada para asegurar condiciones estables durante la detección.

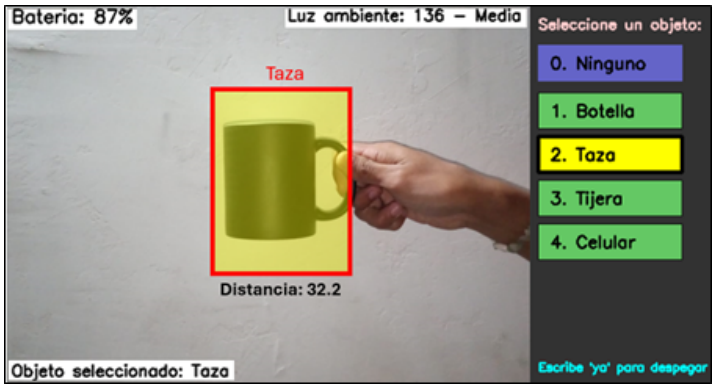


Figura 105: Análisis de distancia de una Taza.

En la Tabla 10 se presentan los resultados de las detecciones del celular, donde se observa un desempeño inferior en comparación con el resto de los objetos analizados.

Tabla 10: Parámetros evaluados en diferentes distancias

Dist.	**Val. (cm)**	**Real (cm)**	**Error (%)**	**T. Det. (s)**	**T. Seg. (s)**
1	50	50.31	0.31	6.18	21.23
2	100	101.24	1.24	6.02	23.16
3	150	150.62	0.62	6.51	20.02

En la primera fase, se configuró manualmente una distancia deseada de 50 cm. El dron alcanzó una distancia real de 50,31 cm, lo que representa un error del 0,31 %. Aunque se trata de una leve desviación, esta se encuentra dentro de márgenes aceptables. El tiempo que tardó en detectar la taza fue de 6,18 segundos, y el tiempo de seguimiento registrado fue de 21,23 segundos, lo cual refleja un desempeño adecuado en esta etapa inicial.

En la segunda fase, la distancia deseada se ajustó a 100 cm. En esta ocasión, el dron alcanzó una distancia real de 101,24 cm, aumentando el error a 1,24 %, es decir, una desviación mayor en comparación con la fase anterior. A pesar de ello, el tiempo de detección se redujo ligeramente a 6,02 segundos, y el tiempo de seguimiento aumentó a 23,16 segundos, indicando una mejora en la capacidad del sistema para seguir el objeto durante más tiempo.

En la tercera fase, se estableció una nueva distancia deseada de 150 cm. El dron logró aproximarse hasta los 150,62 cm, con un error del 0,62 %, menor que el registrado en la segunda fase, pero aún superior al de la primera. En este caso, el tiempo de detección aumentó a 6,51 segundos, siendo el más alto entre las tres pruebas. Este incremento se asocia directamente a la mayor distancia, ya que, a mayor separación, el dron tarda más en localizar el objeto. Por otro lado, el tiempo de seguimiento se redujo a 20,02 segundos, lo que indica una menor duración en la etapa de rastreo.

3.1.1.3 Evaluación de distancia para una tijera

Para esta detección se utilizaron unas tijeras, las cuales representan un desafío para el modelo. Su pequeño tamaño y forma irregular ya sea con las cuchillas abiertas o cerradas dificultan su identificación, lo que puede llevar a que la red las confunda o no las detecte correctamente.

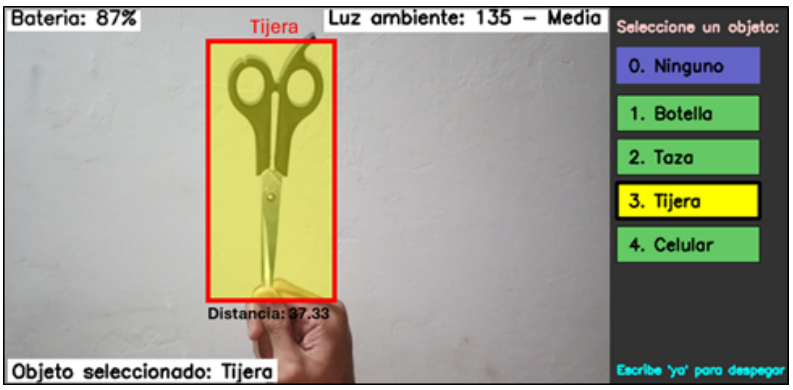


Figura 106: Análisis de distancia de una Tijera.

En la Tabla 11 se muestran los resultados de la detección de este objeto, donde se observa que el modelo presentó mayores dificultades tanto para detectar como para realizar un seguimiento preciso. Esto puede atribuirse a la complejidad en la forma de las tijeras, lo que afecta negativamente el rendimiento del sistema de detección.

Tabla 11: Parámetros evaluados en diferentes distancias

Dist.	Val. (cm)	Real (cm)	Error (%)	T. Det. (s)	T. Seg. (s)
1	50	50.91	0.21	5.02	16.89
2	100	102.10	2.10	5.43	16.31
3	150	152.47	2.47	6.97	15.02

En la primera prueba, se configuró una distancia deseada de 50 cm. El dron logró acercarse al objeto con buena precisión, alcanzando una distancia real de 50,91 cm, lo que representa un error mínimo del 0,91 %. Este valor se considera una aproximación bastante precisa. El tiempo de detección de las tijeras fue de 5,43 segundos, mientras que el tiempo de seguimiento alcanzó los 16,89 segundos, ofreciendo un rendimiento inicial adecuado.

En la segunda prueba, la distancia deseada se incrementó a 100 cm. El dron se acercó hasta los 102,10 cm, aumentando el margen de error al 2,1 %, es decir, se alejó 2,1 cm más de lo previsto. A pesar de este mayor error, el tiempo de detección se mantuvo igual que en la primera prueba, con 5,43 segundos. Sin embargo, el tiempo de seguimiento disminuyó levemente, registrando 16,31 segundos, lo que representa una ligera reducción en comparación con la prueba anterior.

En la tercera prueba, se configuró una distancia de 150 cm. El dron alcanzó una distancia real de 152,47 cm, con un error del 2,47 %, el más alto entre las tres pruebas, indicando una mayor desviación respecto a la distancia deseada. Además, el tiempo de detección aumentó notablemente a 6,97 segundos, siendo el más elevado registrado. Por otro lado, el tiempo de seguimiento volvió a disminuir, ubicándose

en 15,02 segundos, lo que sugiere que, al incrementar la distancia, el seguimiento se hizo más corto y la respuesta del sistema se tornó más lenta.

3.1.1.4 Evaluación de distancia para un celular

El siguiente objeto en detectar será un celular. Aunque el celular suele ser un objeto bien definido y con contrastado, en este caso en particular, el modelo no detecta de manera confiable en el entorno actual.

Figura 107: Análisis de distancia de un celular.

En la Tabla 12 se muestran los resultados obtenidos para esta detección.

Tabla 12: Parámetros evaluados en diferentes distancias

Dist.	Val. (cm)	Real (cm)	Error (%)	T. Det. (s)	T. Seg. (s)
1	50	50.61	0.61	5.02	13.21
2	100	103.87	3.87	5.26	13.18
3	150	152.94	2.94	5.69	12.19

En la primera fase, se ajustó la distancia deseada a 50 cm. El dron alcanzó una distancia real de 50,61 cm, lo que representa un margen de error del 0,61 %, una medición bastante precisa. El tiempo que tardó en detectar el celular fue de 5,02 segundos, con un tiempo de seguimiento de 13,21 segundos. Estos resultados muestran una buena inicialización en la detección, especialmente considerando la complejidad del objeto para ser identificado correctamente.

En la segunda fase, al aumentar la distancia deseada a 100 cm, el dron se aproximó hasta los 103,87 cm, lo que implica un aumento notable del error, que pasó a ser del 3,87 %. En cuanto al tiempo de detección, se registró un ligero incremento a 5,26 segundos, manteniéndose muy cercano al valor de la fase anterior. El tiempo de seguimiento, por su parte, se mantuvo casi igual, con 13,18 segundos, mostrando una estabilidad en el rastreo del objeto a esta distancia mayor.

En la tercera fase, con la distancia deseada establecida en 150 cm, el dron alcanzó una distancia real de 152,94 cm, lo que representa un error del 2,94 %, menor que en la fase anterior pero aún superior a la inicial. El tiempo de detección aumentó a 5,69 segundos, reflejando un mayor tiempo requerido para identificar el objeto a mayor distancia. Por último, el tiempo de seguimiento disminuyó a 12,19 segundos, el valor más bajo entre las tres pruebas, evidenciando una reducción en la capacidad de rastreo conforme la distancia aumenta.

3.1.2. Pruebas del sistema detección de objetos

Para estas evaluaciones se seleccionaron 3 objetos de la misma clase, pero con diferentes formas, es decir, 3 objetos por cada clase de objeto disponible en el sistema. Dado que el menú de opciones contempla 4 clases distintas, se trabajó con un total de 12 objetos en las pruebas. Para ello, se utilizaron objetos con distintas geometrías con el objetivo de identificar cuáles formas son reconocidas con mayor precisión y consistencia por el sistema, lo que permite optimizar su rendimiento bajo diferentes condiciones.

Estas pruebas se realizaron específicamente para calcular las métricas de detección más importantes. Se llevaron a cabo en un lapso de 15 segundos, durante el cual se calculó el Recall, que mide la proporción de objetos reales correctamente detectados respecto al total que deberían haber sido identificados. Además, se evaluó la precisión, que indica cuántas de las detecciones realizadas por el sistema fueron realmente acertadas. Una alta precisión implica que el modelo comete pocos errores al identificar objetos. Cabe mencionar que, para que una detección sea considerada válida, el objeto debe presentar un valor de IoU (Intersection over Union) de confianza superior a 0.40; si este valor es inferior, el sistema no tomará en cuenta la detección como válida.

3.1.2.1 Pruebas de detección de diferentes tipos de botellas

Para este tipo de objetos se seleccionaron tres botellas con formas diferentes, cada una de las cuales presentó distintos resultados. Las pruebas con todas las botellas se realizaron en un ambiente con alta iluminación para garantizar una buena detección y asegurar que ninguno de los tres objetos tuviera ventaja sobre los demás, de modo que la luz no afectara los resultados.

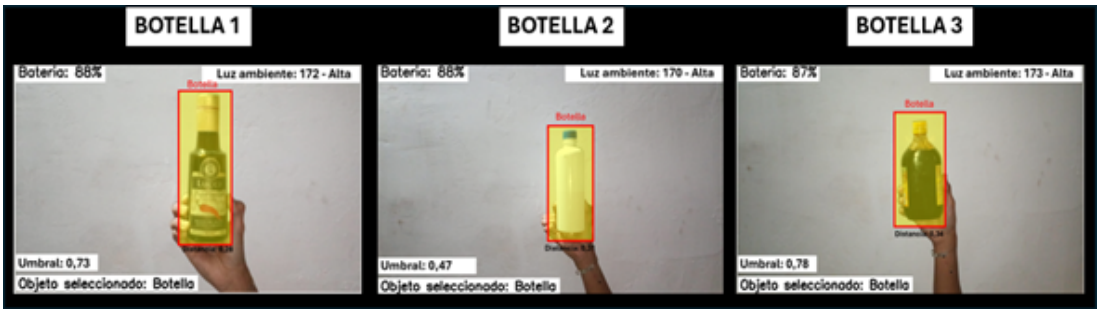


Figura 108: Detección de 3 diferentes botellas.

En la Tabla 13 se muestran los resultados obtenidos para los diferentes tipos de pruebas, en las que los resultados fueron variados debido a las distintas formas de los objetos.

Tabla 13: Resultados por tipo de botella

Métricas. / Tipos. Botellas	**Tiempo (s)**	**IoU 0.4**	**Recall**	**Prec. (%)**
Botella 1	15	0.73	0.93	90
Botella 2	15	0.47	0.53	90
Botella 3	15	0.78	0.93	90

En esta serie de pruebas, se evaluaron diferentes botellas con formas cilíndricas y ovaladas para determinar la efectividad del sistema de detección. La primera botella, de forma cilíndrica y color claro, destacó por un contraste favorable con el entorno, lo que permitió una detección precisa y confiable. En este caso, el sistema alcanzó un umbral de 0,73, y tanto el Recall como la precisión alcanzaron un valor de 0,93. Estos resultados indican un excelente desempeño del sistema, ya que no solo detectó correctamente la mayoría de las veces cuando debía, sino que además cometió muy pocos errores al detectar cuando no correspondía.

Posteriormente, se analizó una botella también cilíndrica, pero con un tono menos contrastante respecto al fondo. Aunque el umbral descendió a 0,47, lo que sugiere una menor confianza en la detección, el sistema logró identificar el objeto con un Recall de 0,53 y una precisión de 0,88. El valor de Recall indica que el objeto no fue detectado todas las veces que debía, lo que representa una sensibilidad moderada, mientras que la precisión se mantuvo relativamente alta, lo cual sugiere que, aunque el sistema falló en algunas detecciones necesarias, las detecciones que sí hizo fueron en su mayoría correctas. Esto evidencia que la falta de contraste afecta negativamente la sensibilidad del sistema, a pesar de que la forma cilíndrica sigue siendo favorable.

Finalmente, se probó una botella con una forma más ovalada, que, a pesar de su ligera variación en la geometría, mostró un desempeño superior al de las pruebas anteriores. En un periodo de evaluación de 15 segundos, el sistema alcanzó un umbral de 0,78, con un Recall y precisión nuevamente en 0,93. Estos valores reflejan una alta sensibilidad y exactitud, demostrando que el sistema puede adaptarse eficazmente a ligeras variaciones en la forma del objeto, siempre que el contras

3.1.2.2 Pruebas de detección de diferentes tipos de Tazas

En el caso de las tazas, se eligieron tres unidades con diseños variados, las cuales ofrecieron resultados distintos durante las pruebas. Estas se llevaron a cabo bajo condiciones de alta iluminación ambiental, con el objetivo de mantener igualdad en la detección y evitar que las diferencias en iluminación.

Figura 109: Detección de 3 diferentes tazas.

En la Tabla 14 se muestran los resultados de las métricas de detección de la taza. Aunque en general las detecciones fueron consistentes y con un desempeño sólido, el sistema presentó inconvenientes en algunas detecciones, incluyendo al menos un objeto que no fue detectado.

Tabla 14: Resultados por tipo de taza

Métricas. / Tipos. Tazas	**Tiempo (s)**	**IoU 0.4**	**Recall**	**Prec. (%)**
Taza 1	15	0.51	0.53	90
Taza 2	15	0.37	0.00	0
Taza 3	15	0.64	0.86	90

Durante las pruebas realizadas con objetos del tipo taza, se evidenció cómo el rendimiento del sistema de detección depende en gran medida de las características visuales del objeto, como su contraste con el fondo y la definición de sus contornos. En la primera prueba se evaluó una taza sin asa, con un diseño no convencional, pero con un color que contrastaba adecuadamente con el entorno. A pesar de no tener la forma típica de una taza, el sistema logró detectarla con un umbral de 0,51, superando el mínimo requerido de 0,4. El Recall fue de 0,53, lo que indica que el objeto fue detectado poco más de la mitad del tiempo en que estuvo presente. La precisión alcanzó un 0,80, lo cual refleja que, aunque no fue detectada constantemente, las veces que sí lo fue correspondieron mayormente a detecciones correctas, con una cantidad moderada de falsos positivos.

En la segunda prueba se utilizó una taza pequeña, redondeada y de color muy similar al entorno, lo que redujo drásticamente la capacidad del sistema para distinguirla del fondo. Esta escasa diferenciación visual tuvo un impacto crítico en el rendimiento, reflejado en un umbral de 0,37, por debajo del mínimo establecido. En consecuencia, el objeto no fue detectado en ningún momento, resultando en valores de Recall y precisión iguales a 0. Este resultado demuestra la total ineficacia del sistema en condiciones de bajo contraste, y evidencia cómo la falta de diferenciación visual afecta tanto la sensibilidad como la fiabilidad del modelo.

Finalmente, se evaluó una taza de color claro, similar al de la taza anterior, pero con contornos más angulosos y definidos, lo que mejoró notablemente la detección. A pesar de que el color seguía siendo poco contrastante, la forma cuadrada permitió al sistema identificar el objeto con mayor facilidad. El umbral alcanzado fue de 0,64, el más alto de todas las pruebas con tazas. El Recall fue de 0,93, indicando que el sistema logró detectar el objeto casi todo el tiempo en que estuvo presente. Además, la precisión se elevó a 0,92, superando ampliamente a las pruebas anteriores y reflejando que prácticamente todas las detecciones fueron correctas, sin generar falsos positivos significativos. Este caso demuestra que la forma y definición de los contornos tienen un impacto determinante en el rendimiento del sistema, incluso cuando el color del objeto no es ideal.

3.1.2.3 Pruebas de detección de diferentes tipos de Tijeras

La siguiente prueba consistió en la selección de tres tijeras, cada una con formas geométricas distintas. Al igual que en los casos anteriores, las pruebas se realizaron bajo una iluminación ambiental elevada, asegurando así que todas las tijeras

estuvieran en igualdad de condiciones y que la luz no influyera en los resultados obtenidos.

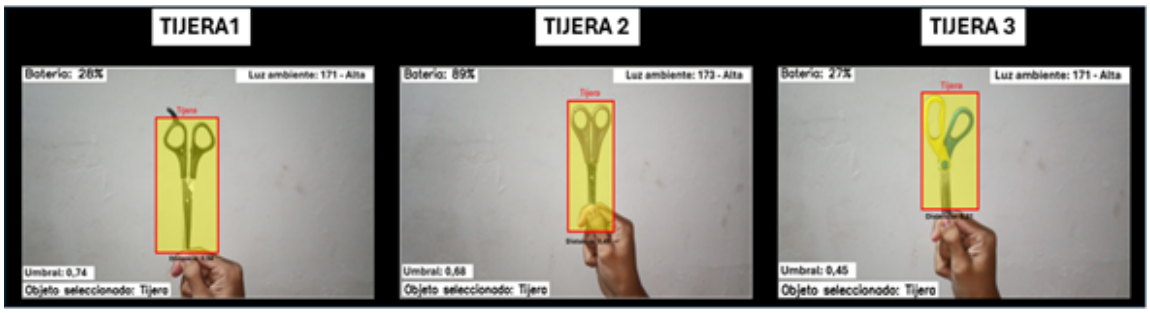


Figura 110: Detección de 3 diferentes tijeras.

En la Tabla 15 se muestran los resultados de las métricas de detección de la tijera. De igual manera, para este objeto se presentaron varios percances en sus detecciones, lo cual indica que, aunque el modelo logra identificarla, aún enfrenta ciertas dificultades en la precisión y consistencia del reconocimiento.

Tabla 15: Resultados por tipo de tijera

Métricas. / Tipos. Tijeras	**Tiempo (s)**	**IoU 0.4**	**Recall**	**Prec. (%)**
Tijera 1	15	0.74	0.73	90
Tijera 2	15	0.68	0.66	90
Tijera 3	15	0.45	0.46	90

Para esta prueba se seleccionó una tijera con contornos convencionales y tonalidades oscuras, lo que generó un buen contraste con el entorno y facilitó su detección. La evaluación, realizada durante 15 segundos, arrojó un umbral de detección de 0,74. El objeto fue correctamente identificado en la mayoría de los casos, alcanzando un Recall de 0,73. Además, la precisión fue de 0,91, lo que indica que las detecciones realizadas por el sistema fueron en su mayoría acertadas, sin una presencia significativa de falsos positivos. Estos resultados evidencian un desempeño sólido del modelo bajo condiciones favorables de forma y color.

En la segunda prueba se utilizó una tijera con forma similar a la anterior, pero con un tono ligeramente más claro. Esta variación afectó moderadamente la capacidad de detección: el umbral se redujo a 0,68 y el Recall descendió a 0,66, lo que indica que el sistema fue menos consistente para identificar el objeto en todos los instantes que estuvo presente. Sin embargo, la precisión se mantuvo en 0,90, lo que confirma que las detecciones continuaron siendo confiables, con muy pocos falsos positivos. Esto sugiere que, aunque el cambio de tonalidad redujo la sensibilidad del modelo, no comprometió su fiabilidad.

Finalmente, en la tercera prueba se analizó una tijera de menor tamaño, con aspas más ovaladas y dos tonalidades distintas, características que se alejaban del patrón morfológico habitual. Como resultado, el umbral de detección bajó considerablemente a 0,45 y el Recall se redujo a 0,46, evidenciando dificultades del sistema para identificar el objeto cuando realmente estaba presente. A pesar de ello, la precisión se mantuvo en 0,87, lo que indica que, aunque el modelo tuvo problemas

para reconocer el objeto constantemente, las pocas veces que lo hizo, las detecciones fueron mayormente correctas y sin falsas alarmas.

3.1.2.4 Pruebas de detección de diferentes tipos de celulares

Para esta última prueba se utilizó el objeto 'celular', seleccionando tres modelos con diseños distintos. De igual manera que en las pruebas anteriores, se mantuvieron las mismas condiciones de alta iluminación ambiental, garantizando que ninguno de los dispositivos tuviera ventaja sobre los otros y asegurando que la detección no se viera afectada por la luz.

Figura 111: Detección de 3 diferentes celulares.

En la Tabla 16 se presentan los resultados de las métricas de detección para la taza, donde se evidencian mayores inconvenientes en comparación con los objetos detectados previamente, ya que se observa una disminución significativa en los valores obtenidos.

Tabla 16: Resultados por tipo de celular

Métricas. / Tipos. Celular	**Tiempo (s)**	**IoU 0.4**	**Recall**	**Prec. (%)**
Celular 1	15	0.46	0.66	90
Celular 2	15	0.51	0.73	90
Celular 3	15	0.43	0.40	90

Durante las pruebas realizadas con objetos del tipo celular, se pudo observar cómo el rendimiento del sistema estuvo fuertemente influenciado por el tamaño, tonalidad y diseño visual del dispositivo. En la primera evaluación se utilizó una tablet de tonos oscuros, cuya forma rectangular es similar a la de un celular, pero con dimensiones considerablemente mayores. A pesar de que este mayor tamaño podría haber favorecido su detección, el sistema no la identificó con la eficiencia esperada. El umbral alcanzado fue de 0,46, apenas por encima del mínimo aceptado, y el Recall fue de 0,66, lo que indica que el objeto fue detectado en un porcentaje moderado del tiempo. La precisión se ubicó en 0,90, evidenciando que la mayoría de las detecciones realizadas fueron correctas, con pocos falsos positivos. Aunque los resultados fueron aceptables, se observa que el tamaño no compensó del todo otras limitaciones visuales.

En la segunda prueba se empleó un celular convencional, con bordes bien definidos y tonalidades claramente diferenciables del entorno. Estas características facilitaron una detección más precisa. El umbral aumentó a 0,51, superando el valor de la

prueba anterior. El sistema logró detectar el objeto con mayor constancia, reflejado en un Recall de 0,73, y la precisión también mejoró levemente, alcanzando un 0,91. Estos valores posicionan esta prueba entre las más destacadas del conjunto, evidenciando que un diseño moderno y buen contraste visual mejoran significativamente el rendimiento del modelo.

En la última prueba se utilizó un celular de tonos claros, correspondiente a un modelo antiguo, pero con bordes bien definidos. A pesar de esa claridad en los contornos, el tono pálido del objeto dificultó su identificación en un entorno de iluminación homogénea. El umbral descendió a 0,43, el más bajo de las tres pruebas, y el Recall se redujo considerablemente a 0,40, reflejando una detección poco consistente. La precisión, aunque también disminuyó, se mantuvo en un nivel aceptable de 0,85, lo que indica que, aunque pocas veces fue detectado, esas detecciones fueron en su mayoría correctas. Este caso demuestra que, si bien los contornos ayudan, la falta de contraste y el diseño obsoleto afectan negativamente la sensibilidad del sistema.

4. Conclusiones y recomendaciones

4.1. Conclusiones

- La red neuronal YOLO fue la opción que mejor se ajustó a las necesidades de este proyecto, gracias a su capacidad para detectar objetos en tiempo real, algo importante para que el dron pudiera operar de forma autónoma. Se eligió específicamente la versión YOLOv4 porque ofrece un equilibrio ideal de velocidad y facilidad de uso, especialmente en equipos con recursos limitados, como en este caso el dron Tello de DJI. Otra gran ventaja fue que el procesamiento de imágenes que requiere no es tan exigente, lo que permitió trabajar cómodamente sin que la computadora se viera sobrecargada. A diferencia de versiones más recientes que suelen necesitar configuraciones más complejas, también se adapta muy bien a entornos sencillos lo que facilitó mucho su implementación.

- La configuración de los parámetros del modelo YOLO se realizó con cuidado para asegurar un rendimiento óptimo en el dron y los recursos computacionales, considerando las limitaciones del hardware. En la subdivisión del batch se dejó en 1, ya que este valor viene predeterminado por el modelo y no fue necesario modificarlo, permitiendo procesar las 64 imágenes simultáneamente sin afectar el rendimiento. En cambio, la resolución de entrada se modificó de su valor original de 1024x1024 a 416x416 píxeles, un valor más adecuado para el sistema del dron y la computadora, ya que reduce la carga computacional manteniendo un equilibrio entre precisión y velocidad. Se eligieron imágenes en color con 3 canales RGB porque, evidentemente, la detección en blanco y negro no sería eficiente en este caso. El parámetro de decay se conservó en 0.0005, un valor lo suficientemente pequeño para no interferir con el aprendizaje, pero efectivo para controlar el crecimiento excesivo de los pesos y evitar sobreajuste. La saturación se modificó a 1.5 para evitar que las imágenes se vean apagadas en ambientes oscuros, para mejorar la capacidad del modelo bajo diferentes condiciones de iluminación. No se aplicó rotación aleatoria en las imágenes durante el entrenamiento, manteniendo la consistencia de los datos y evitando posibles confusiones para la red. Todos estos ajustes facilitaron que tanto el hardware del dron como el de la computadora se adapten adecuadamente al sistema, evitando sobrecargas y asegurando un entrenamiento eficiente y estable.

- El entorno de programación PyCharm resultó esencial para el desarrollo exitoso del sistema. Se pudo programar e integrar fácilmente los distintos algoritmos utilizando librerías como djitellopy, NumPy y OpenCV. Además, facilitó el diseño de la interfaz gráfica, donde se desarrolló el menú de opciones, y el manejo del procesamiento de imágenes. Un aspecto importante fue que también permitió establecer la conexión inalámbrica entre el dron y el sistema principal, lo que hizo posible controlar el dron de forma remota y recibir datos en tiempo real. Gracias a estas funcionalidades, se logró monitorear el comportamiento del sistema durante las pruebas y realizar ajustes precisos cuando fue necesario.

- En conjunto, los resultados obtenidos en las distintas pruebas permiten concluir que el sistema de detección y seguimiento del dron presenta un mejor

rendimiento a distancias medias. Específicamente, se observó que el rango de entre 50 cm y 100 cm representa el intervalo óptimo para la detección precisa y el seguimiento estable de los objetos. Dentro de este rango, los errores en la distancia real se mantuvieron por debajo del 2,1 %, lo que evidencia una aproximación bastante confiable a la distancia deseada. Además, el tiempo de detección se mantuvo relativamente bajo y constante, entre 5 y 6 segundos, mientras que el tiempo de seguimiento alcanzó valores más prolongados, lo que refleja una buena capacidad de rastreo. Sin embargo, al aumentar la distancia a 150 cm, se evidenció una disminución en la precisión del acercamiento, un incremento en el tiempo de detección y una reducción en la duración del seguimiento. Esto indica que, aunque el sistema es capaz de operar a mayores distancias, su estabilidad disminuye gradualmente fuera del rango óptimo.

- Las métricas de detección fueron fundamentales para evaluar el desempeño del sistema y comprender su comportamiento durante las pruebas. El umbral IoU permitió identificar el nivel de confianza con el que el sistema reconocía los objetos, determinando cuán precisa era la coincidencia entre la predicción y la ubicación real. Por su parte, la precisión, evaluada en intervalos de 15 segundos, permitió verificar si el sistema generaba detecciones erróneas, es decir, si identificaba objetos que no estaban realmente presentes. En este caso, no se presentaron mayores problemas relacionados con falsas detecciones, ya que los valores de precisión se mantuvieron elevados en la mayoría de las pruebas. Las variaciones más relevantes se observaron en el valor de Recall, el cual también fue evaluado en ese mismo intervalo de tiempo y ayudó a determinar con qué frecuencia el sistema lograba detectar un objeto mientras este estaba presente.

- • En base a las pruebas realizadas, se concluye que los objetos con formas definidas y buen contraste son los más detectables, destacando las botellas como las de mejor desempeño, con umbrales entre 0,73 y 0,78 y Recall/Precisión de hasta 0,93. Les siguen las tijeras, con umbrales de 0,68 a 0,74, buen contraste y bordes definidos, logrando Precisión de 0,91. En tercer lugar están los celulares, donde los de diseño moderno y bien contrastados alcanzaron umbrales de 0,51 y Recall de 0,73, aunque modelos más claros bajaron a 0,43 con menor sensibilidad. Finalmente, las tazas mostraron el peor rendimiento: solo la de forma cuadrada con contornos definidos logró buenos resultados (0,93 y 0,92), mientras que la taza redondeada y sin contraste no fue detectada teniendo como resultado que todas sus métricas estén en 0. Las formas cilíndricas, cuadradas u ovaladas con buen contraste visual son más favorables, mientras que los objetos pequeños, redondeados y poco diferenciados del fondo presentan mayores dificultades para el sistema de detección.

4.2. Recomendaciones

- Para futuras implementaciones, se sugiere considerar el uso de un dron de gama media o alta, que cuente con mayores capacidades de procesamiento y mejor estabilidad. Esto permitirá ejecutar sistemas de visión artificial más exigentes y con mejores resultados.

- Es fundamental verificar que el equipo donde se ejecutará el sistema (CPU o dispositivo externo) cuente con los recursos suficientes para soportar el procesamiento de imágenes en tiempo real mediante cualquier algoritmo basado en redes neuronales convolucionales. Un sistema con bajo rendimiento puede ocasionar demoras, fallos o pérdida de precisión.

- En caso de contar con un sistema operativo con buena capacidad de cómputo, se recomienda utilizar un modelo CNN de detección más preciso, incluso si este requiere más recursos, ya que esto puede mejorar significativamente la calidad de la detección.

- Antes de cada vuelo, es importante realizar una calibración del dron. Esto incluye revisar sus sensores, giroscopio y cámara, ya que en ciertas ocasiones puede desconfigurarse, afectando su desempeño en vuelo y el funcionamiento del sistema de seguimiento.

- Se recomienda realizar las pruebas en entornos bien iluminados, ya que el dron depende en gran medida de la luz para una correcta detección de objetos. La falta de iluminación puede generar errores en el procesamiento de imágenes y afectar negativamente el desempeño del sistema.

- Se recomienda mantener al dron a distancias cortas o moderadas, específicamente entre 50 y 100 cm del objeto, ya que en este rango el sistema demostró mayor estabilidad y exactitud. Las pruebas evidenciaron que las distancias más largas tienden a generar márgenes de error inesperados en la detección, mientras que operar dentro de este intervalo permite reducir esos errores, lograr tiempos de detección más cortos y una mayor duración en el seguimiento, mejorando significativamente el desempeño general.

- En el proceso de detección, se recomienda elegir objetos físicos que posean contornos bien definidos y características visuales estructuradas y similares al objeto de referencia utilizado en el entrenamiento de la red convolucional. Esto ayudará a minimizar la posibilidad de errores en la detección del modelo, reduciendo la incidencia de falsos negativos y mejorando la precisión general del sistema.

5. Bibliografía

Referencias

[1] ahsan, "Manual de usuario de RYZE Tello Drone," Jul. 2021. [Online]. Available: https://manuals.plus/es/ryze/tello-drone-manual

[2] "Motor sin núcleo 8520 de 8,5 mm x 20 mm (par de sentido horario y antihorario) – Art of Circuits." [Online]. Available: https://artofcircuits.com/product/coreless-motor-8520-8-5mmx20mm-cw-and-ccw-pair

[3] "Tello - DJITelloPy API Reference." [Online]. Available: https://djitellopy.readthedocs.io/en/latest/tello/

[4] " ¿Quién inventó los drones?: Origen e historia." [Online]. Available: https://www.crehana.com/blog/estilo-vida/quien-invento-drones/

[5] R. Briend, "HISTORIAS DE LA AVIACIÓN: EL PRIMER BOMBARDEO AÉREO: VENECIA, 15 DE JULIO DE 1849," Mar. 2020. [Online]. Available: https://historiasdelaviacion.blogspot.com/2020/03/el-primer-bombardeo-aereo-venecia-15-de.html

[6] J. Roa, "inteligencia artificial: en la decada de los 60," Jun. 2012. [Online]. Available: https://josegerma.blogspot.com/2012/06/en-la-decada-de-los-60.html

[7] "Detección de bordes: Cómo las máquinas reconocen formas en imágenes - IA Blog," Mar. 2025, section: Aprendizaje. [Online]. Available: https://iartificial.blog/aprendizaje/deteccion-de-bordes-como-las-maquinas-reconocen-formas-en-imagenes/

[8] p. E. R. Moraguez, "Drones Autónomos: Programación y Aplicaciones Innovadoras," Dec. 2024, section: Guías. [Online]. Available: https://lovtechnology.com/drones-autonomos-programacion-y-aplicaciones-innovadoras/

[9] "Uso de Drones en Inspecciones Industriales: Seguridad y Eficiencia Producción Segura: Estrategias Clave para Evitar Paradas No Planificadas," Feb. 2025. [Online]. Available: https://www.serbusa.net/2025/02/28/uso-de-drones-en-inspecciones-de-infraestructura-industrial-innovacion-y-seguridad/

[10] "El impacto de la Inteligencia Artificial en el desarrollo de drones autónomos." [Online]. Available: https://umilesgroup.com/inteligencia-artificial-y-drones/

[11] C. L. America, "Drones en agricultura, una tecnología con potencial para apoyar especialmente a los pequeños agricultores - CropLife Latin America." [Online]. Available: https://croplifela.org/es/actualidad/drones-en-agricultura-una-tecnologia-con-potencial-para-apoyar-especialmente-a-los-pequenos-

[12] "(PDF) Robots and Artificial Intelligence in the Military," *ResearchGate*, Oct. 2024. [Online]. Available: https://www.researchgate.net/publication/376610260_Robots_and_Artificial_Intelligence_in_the_Military

[13] p. R. d. l. Construcción, "El uso de drones revolucionan las industrias – Revista de la Construcción," Jun. 2024. [Online]. Available: https://www.revistadelaconstruccion.com/el-uso-de-drones-revolucionan-las-industrias/

[14] L. Martinez Figueroa, D. Martinez Vega, and A. Gomez Rodriguez, "Monitoreo activo con drones del estado de plantas de arándanos en invernaderos," Mar. 2024, publisher: Pontificia Universidad Javeriana. [Online]. Available: http://hdl.handle.net/10554/66810

[15] A. G. Tamm, "Detección e identificación de formas utilizando imágenes tomadas de un UAV."

[16] S. A. P. Cabrera and W. R. C. Lote, "Universidad Militar Nueva Granada. Facultad de Ingeniería. Ingeniería en Mecatrónica."

[17] W. Ojeda-Bustamante, A. González-Sánchez, A. Mauricio-Pérez, J. Flores-Velázquez, W. Ojeda-Bustamante, A. González-Sánchez, A. Mauricio-Pérez, and J. Flores-Velázquez, "Aplicaciones de los vehículos aéreos no tripulados en la ingeniería hidroagrícola," *Tecnología y ciencias del agua*, vol. 8, no. 4, pp. 157–166, Aug. 2017, publisher: Instituto Mexicano de Tecnología del Agua. [Online]. Available: http://www.scielo.org.mx/scielo.php?script=sci_abstract&pid=S2007-24222017000400157&lng=es&nrm=iso&tlng=es

[18] "(PDF) Aerial Robotics," in *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/225247482_Aerial_Robotics

[19] D. R. Baixas, "Tipos de controles remotos en drones: ¡Guía definitiva!" Jan. 2025. [Online]. Available: https://dronesriasbaixas.es/tipos-de-controles-remotos-en-drones/

[20] A. Sánchez, "Tipos de Drones Clasificación por uso y características," Oct. 2022. [Online]. Available: https://umilesgroup.com/tipos-de-drones/

[21] "¿Qué es un dron hexacóptero? | Grepow." [Online]. Available: https://www.grepow.com/blog/what-is-a-hexacopter-drone.html

[22] "Bi-copter, tri copter, quadcopter, hexacopter, and octocopter." [Online]. Available: https://www.linkedin.com/pulse/bi-copter-tri-copter-quadcopter-hexacopter-xnklc

[23] VK, "Cómo funcionan y vuelan los cuadricópteros: una introducción a los multirotores -," May 2016, section: Quadcopters/Multirotors. [Online]. Available: https://www.droneybee.com/how-quadcopters-work/

[24] A. Sánchez, "¿Cuáles son las partes de un Dron? Listado completo," Mar. 2023. [Online]. Available: https://umilesgroup.com/partes-de-un-dron/

[25] "OpenELAB Technology Ltd." [Online]. Available: https://openelab.io/es/blogs/learn/comprehensive-guide-to-drone-motors

[26] R. RPAS-Drones, "¿Cómo Son las Hélices para Drones?" Feb. 2025. [Online]. Available: https://rpas-drones.com/helices-para-drones-guia-completa/

[27] N. Cast, "How Drone Propellers Work?" Apr. 2022. [Online]. Available: https://www.remoteflyer.com/how-drone-propellers-work/

[28] "Controladores de vuelo para drones: Guía completa | Grepow." [Online]. Available: https://www.grepow.com/blog/what-is-a-drone-flight-controller.html

[29] "(PDF) Advancements in Quadcopter Development through Additive Manufacturing : A Comprehensive Review," *ResearchGate*, Oct. 2024. [Online]. Available: https://www.researchgate.net/publication/382547719_Advancements_in_Quadcopter_Development_through_Additive_Manufacturing_A_Comprehensive_Review

[30] Natalia, "Tipos de baterías de Drones," Jun. 2023. [Online]. Available: https://idc.apddrones.com/noticias/baterias-de-drones/

[31] "¿Qué es la visión artificial? Concepto y aplicaciones." [Online]. Available: https://www.unir.net/revista/ingenieria/vision-artificial/

[32] marketingbcnvision, "Componentes de la visión artificial para aplicaciones industriales," Sep. 2017. [Online]. Available: https://bcnvision.es/blog-vision-artificial/componentes-vision-artificial/

[33] "Iluminación del sistema de visión: técnicas y filtros | Cognex." [Online]. Available: https://www.cognex.com/es-mx/what-is/machine-vision/the-importance-of-lighting

[34] R. Kashyapa, "Componentes de un sistema de visión artificial - Qualitas Technologies," Jul. 2020. [Online]. Available: https://qualitastech.com/image-acquisition/parts-of-a-machine-vision-system/

[35] "_USComprender la anatomía de un sistema de visión artificial." [Online]. Available: https://emergentvisiontec.com/es/tech-portal/anatomy-of-a-machine-vision-system/

[36] "What Is a Machine Vision Camera? | Zebra." [Online]. Available: https://www.zebra.com/content/zebra1/us/en/resource-library/faq/what-is-machine-vision-camera.html

[37] jmcunat, "Visión artificial: tipos de cámaras y aplicaciones," Oct. 2021. [Online]. Available: https://www.orbitaingenieria.com/noticias/tipos-camaras-vision-artificial/

[38] E. Couth, "Sistemas de visión artificial: tipos y aplicaciones," Mar. 2023. [Online]. Available: https://e2mcouth.com/blog/camaras-para-vision-artificial-tipos-y-aplicaciones/

[39] "Aplicaciones de las cámaras hiperespectrales que están revolucionando la industria - ATRIA Innovation," Oct. 2024. [Online]. Available: https://atriainnovation.com/blog/aplicaciones-camaras-hiperespectrales/

[40] "What is a machine vision SDK?" [Online]. Available: https://www.baslerweb.com/en/learning/machine-vision-sdk/

[41] "Image Classification in AI: How it works." [Online]. Available: https://levity.ai/blog/image-classification-in-ai-how-it-works

[42] "¿Qué es la visión artificial? - Explicación de la IA y el aprendizaje automático de imágenes - AWS." [Online]. Available: https://aws.amazon.com/es/what-is/computer-vision/

[43] "¿Qué es la segmentación de imágenes? | IBM," Dec. 2024. [Online]. Available: https://www.ibm.com/mx-es/topics/image-segmentation

[44] "Fig. 4. Segmentación por umbral." [Online]. Available: https://www.researchgate.net/figure/Segmentacion-por-umbral_fig1_326673260

[45] "Figura 2.8: Ejemplos de histogramas: (a) Imagen con varias..." [Online]. Available: https://www.researchgate.net/figure/Figura-28-Ejemplos-de-histogramas-a-Imagen-con-varias-intensidades-b-Su_fig13_267295870

[46] "Detector de bordes Canny, cómo contar objetos con OpenCV y Python," Apr. 2017, section: Vision Artificial. [Online]. Available: https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/

[47] "Fig.4. Imagens após a aplicação do filtro de Canny. [A] Imagem após a..." [Online]. Available: https://www.researchgate.net/figure/magens-apos-a-aplicacao-do-filtro-de-Canny-A-Imagem-apos-a-primeira-iteracaoB_fig4_265796684

[48] franzpc, "Delimitar una cuenca hidrográfica en ArcGIS," Nov. 2011, section: SIG. [Online]. Available: https://acolita.com/delimitar-automaticamente-micro-cuenca-hidrografica-especifica-en-arcgis/

[49] "Segmentación Basada Regiones." [Online]. Available: https://fastercapital.com/keyword/segmentacin-basada-regiones.html

[50] "Customer Segmentation via Cluster Analysis." [Online]. Available: https://www.optimove.com/resources/learning-center/customer-segmentation-via-cluster-analysis

[51] "Machine Learning | Segmentación de clientes usando ML.NET – Acelera.Tech." [Online]. Available: https://acelera.tech/2020/01/07/machine-learning-segmentacion-de-clientes-usando-ml-net/

[52] "Convolutional Neural Networks: La Teoría explicada en Español | Aprende Machine Learning." [Online]. Available: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[53] "Ejemplo de Kernel en Deep Learning | KeepCoding Bootcamps," Aug. 2022, section: Blog. [Online]. Available: https://keepcoding.io/blog/ejemplo-de-kernel-en-deep-learning/

[54] Ultralytics, "YOLOv4." [Online]. Available: https://docs.ultralytics.com/es/models/yolov4

[55] D. S. R. Raneros, "Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning," Jan. 2021. [Online]. Available: https://uvadoc.uva.es/bitstream/handle/10324/45359/TFM-G1316.pdf?sequence=1&isAllowed=y

[56] "C_5. Yolo V4 explained in full detail-EN - Deep Learning Bible - 4. Object Detection - Eng." [Online]. Available: https://wikidocs.net/167833

[57] "¿Qué son las Redes Convolucionales?" Mar. 2019. [Online]. Available: https://codificandobits.com/blog/redes-convolucionales-introduccion/

[58] "Batch-Normalization para red convolucional," Aug. 2022, section: Blog. [Online]. Available: https://keepcoding.io/blog/batch-normalization-red-convolucional/

[59] "Explicación de la función de activación de Leaky ReLU | Ultralytics." [Online]. Available: https://www.ultralytics.com/es/glossary/leaky-relu

[60] C. Bircanoglu, "Review of YOLOv4 Architecture," Feb. 2024. [Online]. Available: https://cenk-bircanoglu.medium.com/review-of-yolov4-architecture-f488ec32c1c4

[61] "Papers with Code - PANet Explained." [Online]. Available: https://paperswithcode.com/method/panet

[62] "Explicación de Bounding Box | Visión por computador - Ultralytics." [Online]. Available: https://www.ultralytics.com/es/glossary/bounding-box

[63] J. H. Sejr, P. Schneider-Kamp, and N. Ayoub, "Surrogate Object Detection Explainer (SODEx) with YOLOv4 and LIME," *Machine Learning and Knowledge Extraction*, vol. 3, no. 3, pp. 662–671, Sep. 2021, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2504-4990/3/3/33

[64] "Intersección sobre la unión." [Online]. Available: https://digifarm.io/es/blog/intersection-over-union

[65] H. Chakka, "Programming Languages for Drone Development," Aug. 2024. [Online]. Available: https://www.analyticsinsight.net/coding/programming-languages/programming-languages-for-drone-development

[66] A. Titov, "Python logo and symbol, meaning, history, PNG." [Online]. Available: https://1000logos.net/python-logo/

[67] A. Memel, "English: The logo of the C++ programming code," Oct. 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:C%2B%2B_logo.png

[68] "Java Logo," Dec. 2020. [Online]. Available: https://1000marcas.net/java-logo/

[69] J. Llamas, "¿Qué es un sistema de comunicación? Componentes y ejemplos," Oct. 2021. [Online]. Available: https://economipedia.com/definiciones/sistemas-de-comunicacion.html

[70] "¿Qué significa USB? Descubre qué es Universal Serial Bus," May 2023. [Online]. Available: https://www.ionos.com/es-us/digitalguide/servidores/know-how/que-es-usb/

[71] "¿Qué es el wifi? - Tipos de conexiones wifi y seguridad | Proofpoint ES," Jun. 2023. [Online]. Available: https://www.proofpoint.com/es/threat-reference/wifi

[72] "¿Qué es la tecnología Bluetooth®?" [Online]. Available: https://www.intel.com/content/www/xl/es/products/docs/wireless/what-is-bluetooth.html

[73] "Drones: las tecnologías inalámbricas que permiten la operación y el control." [Online]. Available: https://www.data-alliance.net/es/drones-las-tecnologas-inalmbricas-que-permiten-la-operacin-y-el-control

[74] J. Stoner, "Drone Communication Systems," Oct. 2024. [Online]. Available: https://www.flyeye.io/drone-technology-communication/

[75] Luna, "Regulación Drone internacional," Jul. 2022. [Online]. Available: https://idc.apddrones.com/regulacion/regulacion-drone-internacional/

[76] Dronics, "NORMATIVA EUROPEA DRONES / UAS 2023," Jan. 2023. [Online]. Available: https://dronics-technology.com/nueva-normativa-europea-drones-uas/

[77] "_USDrone Laws in Ecuador [Updated January 23, 2025]," Jan. 2025. [Online]. Available: https://drone-laws.com/drone-laws-in-ecuador/

[78] "Dron dji tello combo." [Online]. Available: https://www.vibrant.com.ec/tienda/producto/dr91-dron-dji-tello-combo

[79] "Powered By DJI Tello Propellers." [Online]. Available: https://www.dronenerds.com/products/dji-tello-propellers-cp-pt-00000221-01-dji

[80] M. Martorana, "Programming the DJI Tello Drone with Python (ep. 2)." [Online]. Available: https://marcomartorana.it/post/2020/jan/dji-tello-drone-programming/

[81] "What is PyCharm? Features, Advantages & Disadvantages." [Online]. Available: https://hackr.io/blog/what-is-pycharm

[82] "djitellopy2: Tello drone library including support for video streaming, swarms, state packets and more."

[83] "Introducción a NumPy." [Online]. Available: https://www.w3schools.com/python/numpy/numpy_intro.asp

[84] "OpenCV: Procesamiento de imágenes en OpenCV." [Online]. Available: https://docs.opencv.org/4.x/d2/d96/tutorial_py_table_of_contents_imgproc.html

[85] "Detección y descripción de características — documentación de OpenCV 2.4.13.7." [Online]. Available: https://docs.opencv.org/2.4/modules/features2d/doc/feature_detection_and_description.html

[86] "OpenCV: Clasificador en cascada." [Online]. Available: https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html#autotoc_md1130

Referencias

[1] J. D. J. Lester y L. J. D. Jr, *Writing Research Papers: Pearson New International Edition: A Complete Guide*. Pearson Education, Limited, 2013.

[2] D. Borghys, M. Shimoni, G. Degueldre, y C. Perneel, "Improved object recognition by fusion of hyperspectral and Sar data", Rma.ac.be. [En línea]. Disponible en: `https://www.sic.rma.ac.be/~dborghys/Publications/BorghysEarselSIGIS.pdf`. [Consultado: 08-jul-2023].

[3] S. P. Kuznetsov, "Chaos in three coupled rotators: From Anosov dynamics to hyperbolic attractors", arXiv [nlin.CD], 2017.

Anexos

```
1
2  import cv2
3  import numpy as np
4  from djitellopy import Tello
5  import time
6  import os
7
8  # Clases COCO
9  classes = open("coco.names").read().strip().split("\n")
10
11 # Opciones
12 opciones = {
13     "Botella": "bottle",
14     "Taza": "cup",
15     "Tijera": "scissors",
16     "Celular": "cell phone"
17 }
18 opciones_traducidas = {v: k for k, v in opciones.items()}
19
20 # Tama o del men
21 button_width, button_height, button_margin = 180, 45, 10
22 menu_width = button_width + 2 * button_margin + 30
23
24 # Inicializar el dron
25 tello = Tello()
26 tello.connect()
27 if tello.get_battery() < 20:
28     print("    Bater a baja")
29     exit()
30 tello.streamon()
31
32 # Cargar modelo YOLO
33 net = cv2.dnn.readNet("yolov4-tiny.weights", "yolov4-tiny.cfg")
34 layer_names = net.getLayerNames()
35 try:
36     output_layers = [layer_names[i - 1] for i in net.
          getUnconnectedOutLayers().flatten()]
37 except:
38     output_layers = [layer_names[i[0] - 1] for i in net.
          getUnconnectedOutLayers()]
39
40 # Comando de inicio
41 tello.send_command_with_return("command")
42 print("       Esperando despegue... escribe 'ya' en el teclado")
43
44 # Variables
45 objeto_seleccionado = None
46 despegado = False
47 entrada = ""
48 grabando = False
49 video_writer = None
50 directorio_actual = os.path.dirname(os.path.abspath(__file__))
51
52 # Funci n para seleccionar objeto
53 def seleccionar_objeto(tecla):
```

```
    global objeto_seleccionado
    if tecla in "1234":
        indice = int(tecla) - 1
        objeto_seleccionado = opciones[list(opciones.keys())[indice
            ]]
        print(f"Objeto seleccionado: {objeto_seleccionado}")
    elif tecla == "0":
        objeto_seleccionado = None
        print("       Detecci n desactivada. Objeto seleccionado:
            Ninguno")

# Crear ventana
cv2.namedWindow("Dron Seguidor")

# Loop principal
while True:
    frame = tello.get_frame_read().frame
    frame_resized = cv2.resize(frame, (640, 480))
    height, width = frame_resized.shape[:2]

    # Grabaci n
    if grabando and video_writer is not None:
        video_writer.write(frame_resized)

    # Crear men
    menu_height = height
    menu = np.ones((menu_height, menu_width, 3), dtype=np.uint8) *
        50

    # Instrucciones
    cv2.putText(menu, "Seleccione un objeto:", (10, 30), cv2.
        FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 255), 2)

    # Bot n "0. Ninguno"
    x1 = button_margin
    y1 = 50
    x2 = x1 + button_width
    y2 = y1 + button_height
    cv2.rectangle(menu, (x1, y1), (x2, y2), (200, 100, 100), -1)
    cv2.putText(menu, "0. Ninguno", (x1 + 15, y1 + 30), cv2.
        FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2)



    # Botones de objetos
    botones = []
    for i, (nombre, clave_modelo) in enumerate(opciones.items()):
        y1 = 110 + i * (button_height + button_margin)
        y2 = y1 + button_height

        # Verificar si este bot n corresponde al objeto
            seleccionado
        if objeto_seleccionado == clave_modelo:
            color_fondo = (0, 255, 255)  # Amarillo brillante
            grosor_borde = 4
        else:
            color_fondo = (100, 200, 100)  # Verde normal
            grosor_borde = 1
```

```
106
107         cv2.rectangle(menu, (x1, y1), (x2, y2), color_fondo, -1)
108         cv2.rectangle(menu, (x1, y1), (x2, y2), (0, 0, 0),
                grosor_borde)  # Borde negro para mayor visibilidad
109         cv2.putText(menu, f"{i + 1}. {nombre}", (x1 + 15, y1 + 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2)
110         botones.append((nombre, (x1, y1, x2, y2)))
111
112
113     # Variables de control
114     lr = fb = ud = yaw = 0
115
116     # Despegue
117     if not despegado:
118         cv2.putText(menu, "Escribe 'ya' para despegar", (10,
                menu_height - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
119                     (255, 255, 0), 2)
120     else:
121         if objeto_seleccionado:
122             blob = cv2.dnn.blobFromImage(frame_resized, 0.00392,
                    (416, 416), swapRB=True, crop=False)
123             net.setInput(blob)
124             detections = net.forward(output_layers)
125
126             boxes, confidences, class_ids = [], [], []
127             target_class_id = classes.index(objeto_seleccionado)
128
129             for detection in detections:
130                 for obj in detection:
131                     scores = obj[5:]
132                     class_id = np.argmax(scores)
133                     confidence = scores[class_id]
134
135                     if class_id == target_class_id and confidence >
                            0.3:
136                         cx, cy, w, h = (obj[0:4] * np.array([width,
                                height, width, height])).astype("int")
137                         boxes.append([cx - w // 2, cy - h // 2, w,
                                h])
138                         confidences.append(float(confidence))
139                         class_ids.append(class_id)
140
141             indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5,
                    0.3)
142
143             if len(indices) > 0:
144                 for i in indices.flatten():
145                     x, y, w, h = boxes[i]
146                     cx, cy = x + w // 2, y + h // 2
147                     area = w * h
148
149
150                     # Calcular distancia estimada (basado en ancho
                            del objeto detectado)
151                     distancia = 8000 / w  # Ajusta el valor 8000
                            seg n tus pruebas/calibraci n
152                     distancia_texto = f"Distancia: {distancia:.2f}
                            cm"
```

```
153
154                     ultima_posicion = (cx, cy, area)
155
156
157                     # ---------- Simulaci n de contorno (OPCI N
                            3) ----------
158                     overlay = frame_resized.copy()
159
160                     # Dibujar un rect ngulo relleno
                            semitransparente sobre el objeto
161                     cv2.rectangle(overlay, (x, y), (x + w, y + h),
                            (0, 255, 255), -1)  # Relleno amarillo
162
163                     # Combinar la imagen original con el overlay
164                     alpha = 0.3  # Transparencia
165                     frame_resized = cv2.addWeighted(overlay, alpha,
                            frame_resized, 1 - alpha, 0)
166
167                     # Dibujar contorno visible
168                     cv2.rectangle(frame_resized, (x, y), (x + w, y
                            + h), (0, 255, 255), 3)  # Borde amarillo
169
170                     # Etiqueta
171                     cv2.putText(frame_resized, objeto_seleccionado,
                            (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
172                             (0, 255, 255), 2)
173
174                     cv2.putText(frame_resized, distancia_texto, (x,
                            y + h + 20), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
175                     (0, 0, 255), 2)
176
177                     # Mostrar coordenadas y tama o del objeto
178                     info_text = f"Pixeles: x={x}, y={y}, w={w}, h={
                            h}"
179                     cv2.putText(frame_resized, info_text, (x, y + h
                            + 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
180                             (255, 255, 255), 2)
181
182                     error_x = cx - width // 2
183                     error_y = cy - height // 2
184                     ideal_area = (width * height) // 7
185                     error_area = area - ideal_area
186
187                     distancia_deseada = 70  # en cm
188                     error_distancia = distancia - distancia_deseada
189
190                     print("Distancia actual:", distancia)
191                     print("Error de distancia:", error_distancia)
192
193                     if abs(error_distancia) < 2:
194                         fb = 0
195                         print("Dentro del rango, no se mueve")
196                     elif error_distancia > 0:
197                         fb = int(np.clip(error_distancia * 0.8, 15,
                                70))  # Acercarse (moverse hacia
                                adelante)
198                         print("El objeto est  lejos     acercarse
                                con fb =", fb)
```

```
                    else:
                        fb = int(np.clip(-error_distancia * 1.2,
                            15, 70))  # Retroceder (moverse hacia
                            atr s)
                        print("El objeto est  cerca     retroceder
                             con fb =", fb)

                    lr = 0
                    yaw = int(np.clip(error_x * 0.1, -40, 40))
                    ud = int(np.clip(-error_y * 0.15, -40, 40))
                    fb = int(np.clip(-error_area * 0.0002, -90, 30)
                        )
            else:
                yaw = 15
                fb = 0
                ud = 0
        else:
            yaw = 15

        tello.send_rc_control(lr, fb, ud, yaw)

    battery = tello.get_battery()
    intensidad_luz = np.mean(frame_resized)
    nivel_luz = "Baja" if intensidad_luz < 85 else "Media" if
        intensidad_luz < 170 else "Alta"
    text_battery = f"Bateria: {battery}%"
    text_luz = f"Luz ambiente: {int(intensidad_luz)} - {nivel_luz}"
    texto_objetos = f"Objeto seleccionado: {opciones_traducidas.get
        (objeto_seleccionado, 'Ninguno')}"
    estado_dron = "En vuelo" if despegado else "En tierra"

    font = cv2.FONT_HERSHEY_SIMPLEX
    scale = 0.8
    thickness = 2
    margin = 5

    (w_bat, h_bat), _ = cv2.getTextSize(text_battery, font, scale,
        thickness)
    (w_luz, h_luz), _ = cv2.getTextSize(text_luz, font, 0.7,
        thickness)
    (w_obj, h_obj), _ = cv2.getTextSize(texto_objetos, font, 0.7,
        thickness)
    (w_est, h_est), _ = cv2.getTextSize(estado_dron, font, 0.7,
        thickness)

    # --- Indicador de bater a ---
    cv2.rectangle(frame_resized, (5, 5), (5 + w_bat + 2 * margin, 5
        + h_bat + 2 * margin), (255, 255, 255), -1)
    cv2.putText(frame_resized, text_battery, (5 + margin, 5 + h_bat
        + margin - 5), font, scale, (0, 0, 0), thickness)

    # --- Indicador de luz ambiente ---
    cv2.rectangle(frame_resized, (width - w_luz - 2 * margin - 5,
        5), (width - 5, 5 + h_luz + 2 * margin),
                  (255, 255, 255), -1)
    cv2.putText(frame_resized, text_luz, (width - w_luz - margin -
        5, 5 + h_luz + margin - 5), font, 0.7, (0, 0, 0),
                thickness)
```

```
243
244     # --- Indicador de objeto seleccionado ---
245     cv2.rectangle(frame_resized, (5, height - h_obj - 2 * margin -
            10), (5 + w_obj + 2 * margin, height - 10),
246                   (255, 255, 255), -1)
247     cv2.putText(frame_resized, texto_objetos, (5 + margin, height -
            margin - 10), font, 0.7, (0, 0, 0), thickness)
248
249     # --- Indicador de estado del dron (En vuelo / En tierra) ---
250     cv2.rectangle(frame_resized, (width - w_est - 2 * margin - 5,
            height - h_est - 2 * margin - 10),
251                   (width - 5, height - 10), (255, 255, 255), -1)
252     cv2.putText(frame_resized, estado_dron, (width - w_est - margin
            - 5, height - margin - 10), font, 0.7, (0, 0, 0),
253                 thickness)
254
255
256
257     # Mostrar interfaz
258     frame_resized_rgb = cv2.cvtColor(frame_resized, cv2.
            COLOR_BGR2RGB)
259     interfaz = np.hstack((frame_resized_rgb, menu))
260     cv2.imshow("Dron Seguidor", interfaz)
261
262     # Teclas
263     key = cv2.waitKey(1) & 0xFF
264     if key != 255:
265         entrada += chr(key)
266         entrada = entrada[-2:]
267
268         if entrada == "ya" and not despegado:
269             try:
270                 tello.takeoff()
271                 despegado = True
272                 print("       Dron despeg ")
273             except Exception as e:
274                 print(f"Error al despegar: {e}")
275             entrada = ""
276
277     if key == ord('l'):
278         tello.land()
279         break
280     elif key == 27:
281         break
282     elif chr(key) in "01234":
283         seleccionar_objeto(chr(key))
284     elif key == ord('g') and not grabando:
285         print("       Iniciando grabaci n...")
286         grabando = True
287         fourcc = cv2.VideoWriter_fourcc(*'XVID')
288         video_writer = cv2.VideoWriter(os.path.join(
                directorio_actual, "grabacion.avi"), fourcc, 20.0, (640,
                 480))
289     elif key == ord('s') and grabando:
290         print("       Deteniendo grabaci n...")
291         grabando = False
292         video_writer.release()
293         video_writer = None
```

```
294
295 # Finalizar
296 cv2.destroyAllWindows()
297 tello.streamoff()
298 tello.end()
```

Listing 1: Código del sistema de detección y seguimiento de objetos

Reporte de coincidencias

CERTIFICADO DE ANÁLISIS
magister

Trabajo_Integración_sincodigo

5% Textos sospechosos

- 2% Similitudes
  - < 1% similitudes entre comillas
  - 0% entre las fuentes mencionadas
- 3% Idiomas no reconocidos (ignorado)
- 20% Textos potencialmente generados por la IA (ignorado)

Nombre del documento: Trabajo_Integración_sincodigo.pdf
ID del documento: 72a22a7239f87c35e7ca8f956d6fe91bb9267ae5
Tamaño del documento original: 8,05 MB

Depositante: Junior Rafael Figueroa Olmedo
Fecha de depósito: 19/6/2025
Tipo de carga: interface
fecha de fin de análisis: 19/6/2025

Número de palabras: 39.942
Número de caracteres: 257.201

Ubicación de las similitudes en el documento:

Fuentes principales detectadas

N°	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	Trabajo_Integración_Curricular_Auto_Quirumbay.pdf | Trabajo_Integr... #5b7613 El documento proviene de mi grupo 14 fuentes similares	15%		Palabras idénticas: 15% (6102 palabras)
2	Tesis_Final.pdf | Tesis_Final #516fde El documento proviene de mi biblioteca de referencias 12 fuentes similares	< 1%		Palabras idénticas: < 1% (233 palabras)
3	Trabajo_Integración_Curricular_Auto.pdf | Trabajo_Integración_Curricu... #10c8e3 El documento proviene de mi grupo 7 fuentes similares	< 1%		Palabras idénticas: < 1% (163 palabras)
4	Trabajo_Integración_Curricular_Auto_Roger_Carbo.pdf | Trabajo_Integ... #ad1c59 El documento proviene de mi grupo 4 fuentes similares	< 1%		Palabras idénticas: < 1% (113 palabras)
5	doi.org | Régimen jurídico de los drones: el nuevo Reglamento (UE) 2018/1139 = ... https://doi.org/10.20318/cdt.2019.4618 2 fuentes similares	< 1%		Palabras idénticas: < 1% (32 palabras)

Fuentes con similitudes fortuitas

N°	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	www.boe.es https://www.boe.es/buscar/pdf/2022/BOE-A-2022-15286-consolidado.pdf	< 1%		Palabras idénticas: < 1% (35 palabras)
2	Trabajo_de_Integracion_Curricular_Tigrero_2022.pdf | Trabajo_de_Inte... #8e3518 El documento proviene de mi grupo	< 1%		Palabras idénticas: < 1% (28 palabras)
3	Documento de otro usuario #89b2cf El documento proviene de otro grupo	< 1%		Palabras idénticas: < 1% (32 palabras)
4	capitulo 1.pdf | capitulo 1.pdf #5b0c6b El documento proviene de mi grupo	< 1%		Palabras idénticas: < 1% (17 palabras)
5	codigosdelmundo.puntanetwork.com | Regulaciones de drones internacionales... https://codigosdelmundo.puntanetwork.com/global/fronteras-invisibles-como-regulaciones-d...	< 1%		Palabras idénticas: < 1% (17 palabras)

Fuentes mencionadas (sin similitudes detectadas)

Estas fuentes han sido citadas en el documento sin encontrar similitudes.

1 https://manuals.plus/es/ryze/tello-drone-manual
2 https://artofcircuits.com/
3 https://www.crehana.com/blog/estilo-vida/quien-invento-drones/
4 https://historiasdelaviacion.blogspot.com/2020/
5 https://josegerma.blogspot.com/2012/06/en-la-decada-de-los-60.html