industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Diseño de un sistema de monitorización para invernadero basado en un robot aéreo autónomo.

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

Autor: Daniel Martínez Zapata
Directora: Dra. Nieves Pavón Pulido
Codirector: Dr. Juan Antonio López Riquelme

Cartagena, a 8 de diciembre de 2020

Agradecimientos

Primero de todo, agradecer a mis padres por todo el apoyo que me han dado, ya que sin el esfuerzo que realizan día a día no me habría sido posible haber completado mis estudios. También quiero agradecer a Alba por el apoyo y la ayuda que me ha brindado durante el desarrollo de este trabajo, así por confiar en mí día y día y no hacerme perder la confianza en mí mismo.

RESUMEN

En este trabajo se presenta el desarrollo de una aplicación para agricultura de precisión basada en el uso de un vehículo aéreo no tripulado para la detección del estado fenológico de las flores de la variedad *strelitzia reginae* en cultivo de interior. La detección se realizará mediante el uso de tecnologías de Deep Learning para la clasificación de imágenes. Se propondrá, a su vez, un sistema de navegación para el dispositivo basado en visión por computador de forma que este sea capaz de desplazarse por un entorno estructurado. Así mismo, se estudiará el efecto de la aplicación de estrategias de *data augmentation* y *transfer learning* para mejorar el aprendizaje del modelo propuesto.

Índice

Figuras.

Tablas

1 Introducción.

En este capítulo se definirán la motivación del trabajo y los objetivos a alcanzar en la realización del mismo. A su vez, se presentará una síntesis de los capítulos que prosiguen al presente.

1.1 Motivación y objetivos.

Con la introducción de la agricultura de precisión se pretende conseguir un grado alto de automatización en las plantaciones, con el fin de mejorar la productividad y eficiencia de los recursos. Un punto muy importante es la posibilidad de conocer y predecir la cantidad de frutos que se van a producir en una campaña agrícola, con el fin de realizar una estimación de ganancia y coste para estimar los beneficios. Es en este aspecto en el que se ha enfocado este trabajo.

Si bien la observación del estado de maduración de los frutos de forma manual resulta muy efectiva para predecir la producción – y un agricultor experto es capaz de realizarla sin problema alguno – esta tarea se puede automatizar con el doble objetivo de liberar al agricultor de la necesidad de realizar una tarea repetitiva (la cual no aporta valor material al producto) y de contar con una consistencia en los resultados ya que no está presente el juicio subjetivo humano.

A su vez, esta automatización de la tarea del reconocimiento de frutos puede traer nuevas oportunidades laborales y el surgimiento de empresas que se dediquen a realizar dicha función. A parte de las ventajas prácticas y económicas que se proponen, no hay que desestimar el valor científico con el que cuenta el presente trabajo ya que se ofrecen y contrastan sistemas y algoritmos dentro de los campos de la robótica móvil, la visión artificial y el Machine Learning.

El objeto del presente trabajo, por tanto, consiste en el desarrollo e implementación de una aplicación basada en robótica móvil y tecnologías de Deep Learning en el ámbito de la agricultura de precisión. La finalidad es que, mediante el uso de un drone dentro de un invernadero de floricultura, se pueda reconocer el estado fenológico de flores de la variedad *strelitzia reginae*. Para que se considere que se ha alcanzado el objeto de este trabajo es necesario cumplir una serie de subobjetivos presentes en la etapa de desarrollo. Estos objetivos específicos son los siguientes:

- Desarrollo de la arquitectura de Deep Learning necesaria para el reconocimiento de las flores.
- Desarrollo de un sistema de navegación para un robot aéreo no tripulado en un entorno estructurado.
- Y, por último, la implementación de esta aplicación en un robot real.

1.2 Resumen de capítulos.

En este apartado se realiza un resumen de los capítulos que continúan a este, con el fin de que el lector tenga una visión global de los contenidos tratados en el presente trabajo antes de ahondar en el contenido del mismo.

- Capítulo 2. Estado del arte:

  En este capítulo se realizará un planteamiento completo del problema propuesto, así como los precedentes a éste en materia de investigación y desarrollo. Se centrará el foco sobre el uso de robots y drones en el contexto de la agricultura de precisión, haciendo énfasis en el uso de estos dentro de invernaderos. A su vez, se tratarán los sistemas utilizados para el reconocimiento de frutos, tanto por el lado de la visión artificial como por el lado del Deep Learning.

- Capítulo 3. Diseño del sistema:

  En este capítulo se realizará, en primer lugar, una descripción global de la arquitectura empleada enumerando los elementos, tanto de hardware como de software, que la componen. Posteriormente, se describirán en detalle los elementos de hardware y software utilizados, así como la integración entre ellos. Dentro del apartado de hardware se detallarán los componentes físicos utilizados, principalmente el drone y su versión simulada,, así como el uso de una placa de desarrollo ESP32 en conjunto con sensores de ultrasonidos HC-SR04. En el apartado de software se describirá cual es el sistema de navegación propuesto para el drone basado en visión artificial, así como los sistemas de control necesarios para este. Finalmente se presentará la arquitectura de visión artificial y Machine Learning utilizada para el reconocimiento de los frutos.

- Capítulo 4. Análisis de resultados:

  En este capítulo se expondrán las pruebas realizadas durante el desarrollo y la puesta en marcha del sistema propuesto, con el fin de comprobar y validar su eficacia. A su vez, se presentarán los resultados obtenidos en cada una de estas pruebas y se realizará una discusión de estos. En esta discusión, además, se compararán los resultados obtenidos con los obtenidos por otras investigaciones y se justificará la metodología seguida. Adicionalmente se presentarán las ventajas y desventajas del sistema.

- Capítulo 5. Conclusiones y trabajo futuro:

  En el último capítulo se contemplarán las conclusiones que se han obtenido del desarrollo de este trabajo y se marcarán las líneas futuras de trabajo con el fin de mejorar el sistema propuesto.

2 Estado del arte.

En el presente capítulo se realiza una introducción a los temas tratados en el trabajo, así como una visión histórica de los mismos, los avances en materia de investigación y la aplicación de estos. Se centra el foco en el concepto de agricultura de precisión, el uso de robots en este entorno y el reconocimiento de frutos mediante visión e inteligencia artificiales.

2.1 Agricultura de precisión.

La agricultura de precisión (AP) hace referencia a una serie de técnicas basadas en la aplicación de las tecnologías de la comunicación y la información (TIC) para mejorar la eficiencia del proceso productivo, así como la de los insumos utilizados, y reducir el impacto ambiental del proceso productivo en el entorno.

Uno de los puntos más importantes de la AP es la de permitir gestionar el uso de insumos atendiendo a factores temporales y geográficos. Esto se puede realizar gracias al uso de la tecnología de geolocalización (GPS) que permite, junto con el uso de elementos sensores, establecer un mapa de las necesidades y características de los cultivos o el sustrato con el fin de poder, a partir de estos datos, establecer estrategias de gestión de recursos y planificación. Este punto es el que permite establecer una diferenciación entre la agricultura tradicional y la AP.

Mientras que en la agricultura tradicional se utilizan técnicas y generalidad transmitidas entre generaciones – las cuales no tienen una aplicabilidad precisa para todo tipo de terrenos, cultivos y situaciones climáticas, llevando, en ocasiones a una gestión de los recursos que carece de un grado alto de eficiencia – la AP pretende establecer una metodología aplicable a cada situación con la máxima de mejorar la eficiencia. [1] [2]

Sin embargo, aunque la AP presenta grandes ventajas sobre la agricultura tradicional, la aplicación de esta no siempre resulta necesaria. Para plantaciones y cultivos que no cuentan con una gran superficie de cultivo o explotaciones agrícolas que, mediante el refinamiento de las técnicas tradicionales de agricultura han conseguido optimizar de forma eficiente el uso de insumos, es posible que la aplicación de la AP no presente una ventaja o un beneficio económico que compense la gran inversión que conlleva.

Aún con todo esto, el uso de la AP se encuentra en aumento y cada vez existe menos reticencia por parte de los agricultores tradicionales en implementar estas técnicas en sus cultivos. Dentro del territorio nacional existen multitud de empresas cuya actividad empresarial está enfocada a la AP. Algunas de estas empresas son SGS España, Agrosap, Drónica o Aerocamaras, entre otras. Así mismo, existen cooperativas agrícolas que apoyan el uso de tecnologías innovadoras con el objetivo de mejorar la eficacia y sostenibilidad en los cultivos. Una de estas cooperativas agrícolas es la de Surinver, con sede en Pilar de la Horadada (Alicante).

Al igual que en plantaciones al aire libre, la AP de precisión también tiene aplicabilidad en el cultivo de interior. La sensorización de variables propias del suelo o de los cultivos también resulta necesaria en el cultivo dentro de un invernadero con el fin de maximizar la producción. Los invernaderos automatizados también son una realidad tangible, controlando los ciclos de apertura y cierre para asegurar unas condiciones térmicas y lumínicas adecuadas, así como del grado de humedad y la ventilación dentro del mismo. Un ejemplo de esto es el invernadero inteligente más grande del continente asiático gestionado por la empresa China *Kaisheng Haofeng* [3].

2.2 Uso de la robótica móvil en la agricultura de precisión.

Cada vez está más presente en la AP el uso de robots para realizar diversas tareas; desde la labranza del terreno con mediante tractores autónomos, como el que ofrece la marca John Deere (Figura 1), tareas de pulverización de pesticidas utilizando drones, también ofrecido por la marca John Deere (Figura 2), robots encargados de la eliminación de maleza, como el ofrecido por la empresa francesa VITIROVER (Figura 3), o los robots recolectores, como el proyecto europeo SWEEPER (Figura 4) para recolección de pimientos entre otros.

Figura 1. Tractor autónomo John Deere.

Figura 2. Robot aéreo John Deere para fumigación.

Figura 3. Robot para corte de maleza.

Figura 4. Proyecto europeo SWEEPER

Centrando más la atención sobre robots aéreos, encontramos multitud de aplicaciones como el uso de estos para la detección del estado hídrico de los cultivos, flujo de agua, índices de vegetación, cartografía de parcelas o fumigación de estas, como el mencionado anteriormente de la marca John Deere. Así mismo, existen investigaciones y desarrollos para la aplicación de los robots aéreos en interior, como es el caso del proyecto conjunto entre el Centro de Automática y Robótica (CAR), la Universidad Politécnica de Madrid (UPM) y el Consejo Superior de Investigaciones Científicas (CSIC) para desarrollar una aplicación capaz de monitorizar las variables ambientales y de crecimiento dentro de un invernadero [4].

El uso de la robótica móvil está cada vez más presente en la AP. Según un estudio económico realizado en el mes de Julio del presente año por el portal online *researchandmarkets.com*, el mercado de robots para agricultura en EEUU tiene un valor estimado de 566.4 millones de dólares americanos en el año 2020 y prevé un crecimiento de sector económico anual en china entre el año 2020 y el año 2027 de un 20.6% alcanzando el valor estimado de 1,3 mil millones de dólares americanos para el año 2027 [5].

2.3 Visión artificial y Deep Learning en agricultura de precisión.

El reconocimiento de frutos dentro de la AP resulta un objetivo muy interesante puesto que permite diseñar sistemas encargados de la recolección de estos y permite realizar predicciones de la producción. Esta tarea se puede realizar mediante el empleo de visión artificial, del Deep Learning o, como en el caso de este trabajo, realizando una combinación de ambos enfoques.

2.3.1 Estrategias de Visión Artificial en Agricultura.

Para afrontar el problema de la detección de frutos y flores mediante visión artificial se han propuesto múltiples métodos. El enfoque clásico consiste en la detección por color como proponen Thorp et al. [6] realizando una conversión de la imagen del espacio de color RGB al espacio de color HSI y umbralizando para obtener la región de interés donde se encuentran las flores en función del color.

Arivazhagan et al. [7] proponen un método de detección de frutos basado en color y en textura. El método que proponen consiste en, primero, extraer la región de interés mediante una conversión de la imagen al espacio de color HSV y umbralizando la imagen; teniendo ya la región de interés se extraen las características de color y textura mediante un modelo entrenado. La conclusión a la que se llega en este artículo es que el uso conjunto del color y la textura permiten una mayor precisión a la hora de detectar y clasificar frutos en imágenes.

Biradar et al. [8] proponen como método de detección de flores el uso de técnicas de segmentación basadas en el algoritmo de Otsu para umbralizar la imagen, realizando posteriormente una extracción de bordes mediante el algoritmo de Canny. Posteriormente se realiza el conteo de los contornos.

Otro enfoque es el propuesto por Tiay et al. [9] quienes presentan un método similar al anterior, ya que se realiza una extracción de bordes a través del método de Canny. La diferencia es que en lugar de contabilizar el número de contornos se calculan los momentos de Hu de cada uno de ellos. Estos momentos permiten extraer características propias de los contornos los cuales son invariantes a la rotación, traslación o escala. A partir de esto se compara el valor de estos momentos para determinar qué tipo de flor es atendiendo a la forma de esta.

Estas son algunas de las estrategias propuestas por diferentes grupos de investigación. En el caso de este trabajo, como se pretende extraer de una imagen la región de interés en la que se encuentran las flores de la variedad *strelitzia reginae* (Figura 5), se realizará una segmentación por color ya que el contraste del color de estas flores respecto al del fondo es considerable.

Figura 5. Planta de strelitzia reginae en cultivo de interior.

Se utilizará un método similar al propuesto por [7], sin embargo, no se tendrá en cuenta el efecto de la textura en la clasificación de las imágenes.

2.3.2 Uso de Deep Learning para reconocimiento y clasificación.

Antes de poder hablar del Deep Learning es necesario definir qué es y en qué rama del conocimiento se encuentra. El Deep Learning se encuentra como subconjunto del Machine Learning, el cual, a su vez, se encuentra dentro de la Inteligencia Artificial (AI). El Machine Learning consiste en dotar a una máquina o software de la capacidad de desarrollar algoritmos, con el fin de reconocer patrones en los datos de entrada y ser capaz de generar datos de salida o clasificar los de entrada. Estos sistemas clasifican las características de los datos a analizar mediante árboles de decisiones y clasificadores estadísticos.

De la misma manera, el Deep Learning tiene el mismo objetivo que el Machine Learning, pero la diferencia clave entre ambos es la forma en la que se generan estos algoritmos de clasificación. Mientras que el Machine Learning utiliza árboles de decisiones, el Deep Learning utiliza una estructura que se conoce como Red Neuronal Artificial (RNA). El método de aprendizaje de estas redes neuronales intenta emular el mecanismo de aprendizaje de las redes neuronales biológicas –tratándolo siempre desde un alto grado de abstracción- [10]; esto es, a través de la interconexión de un gran número de células –o neuronas- con el objetivo de que, a partir de un dato de entrada –denominado como característica o *feature*-, produzca una salida –denominada como etiqueta o *label*-.

Las RNA están formadas por una red de capas conectadas entre ellas y que comparten información. Estas capas, a su vez, están formadas por neuronas, las cuales dentro de una RNA suponen la unidad más básica de procesamiento de datos. Cada neurona está compuesta por unas variables internas denominadas pesos y sesgos. Los pesos ponderan las diferentes entradas a la neurona y el sesgo compensa la suma de estos valores compensados. Cada una de las neuronas de una RNA se puede definir por la siguiente ecuación:

$$a_i = \sum_{j=1}^{n} W_{ij} x_j \ + b_i$$

Donde: a_i representa la salida de la neurona i; x_j es cada una de las entradas de datos a la neurona i; w_{ij} es el peso correspondiente a cada una de las entradas x_j; y b_i es el sesgo correspondiente a la neurona i.

Durante el proceso de entrenamiento es cuando estas variables internas se ajustan de manera que, ante un dato de entrada, se produzca una respuesta [11]. Cuando todas las neuronas de una capa se encuentras conectadas con todas las neuronas de la capa anterior y la capa siguiente, se produce lo que se denomina una red neuronal profunda. Este es el tipo de redes RNAs que se utilizarán en este trabajo. En la aplicación del Deep Learning al tratamiento de imagen se presentan diferentes estratégicas dependiendo del número de objetos presentes en la imagen y el método utilizado. En la Figura 6 se muestran las diferentes estratégicas asociadas al tratamiento de imagen.

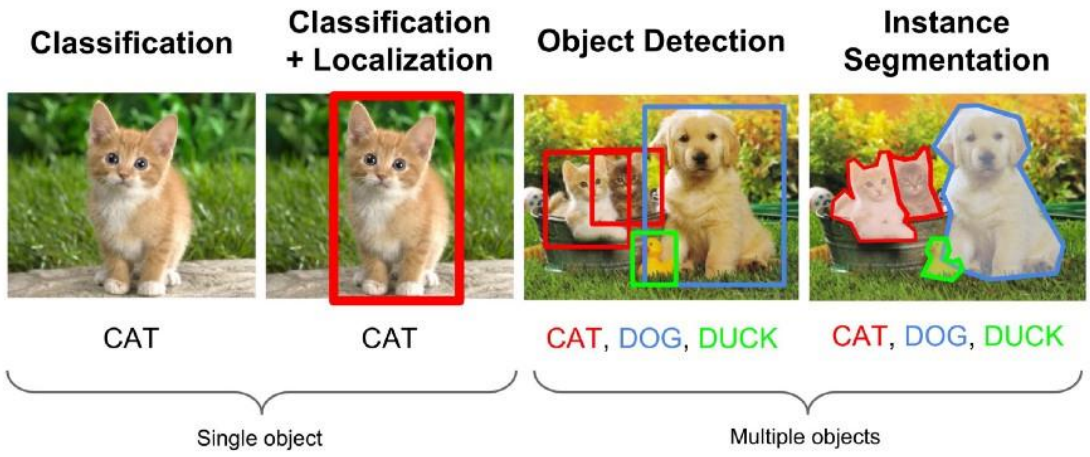

Figura 6. Aplicaciones de Deep Learning al tratamiento de imagen.

Tanto en el problema de clasificación de imagen como de clasificación y localización, la RNA cataloga la imagen de entrada asignándole una de las etiquetas previamente definidas por la persona usuaria. En la detección de objetos y en la segmentación, la RNA asigna etiquetas a los diferentes elementos presentes en la imagen. La diferencia entre estas dos estrategias reside en las limitaciones que presenta el primer método y la complejidad del segundo.

Mientras que en la clasificación de imagen sólo se puede asignar una etiqueta por imagen, y en estas debe aparecer únicamente un objeto a detectar, la detección de objetos permite realizar detecciones múltiples, si bien presenta el inconveniente de que requiere más recursos computacionales, teniendo la necesidad de recurrir en ocasiones a métodos de computación paralela utilizando tarjetas gráficas.

En materia de detección de frutos y flores mediante Deep Learning se han realizado una gran cantidad de investigaciones con enfoques muy distintos para el mismo problema. En el caso de Mureşan et al. [12] se presenta un método de clasificación de imágenes de diversas frutas utilizando Redes neuronales convolucionales (CNN) consiguiendo una precisión del 96,3%.

Por parte de Philipe et al. [13] se presenta un método de detección y clasificación de flores de manzano mediante el uso de CNN y máquinas de soporte de vectores (SVM) en un entorno real de una plantación.

En el método propuesto por Farjon et al. [14] consiste en la detección de flores de manzano a través de Redes Neuronales Convolucionales basadas en Region (R-CNN), las cuales son utilizadas en detección de objetos.

En el caso de este trabajo se realiza una clasificación de imágenes mediante CNNs de forma similar al método propuesto en [12] con la excepción de que el modelo no se desarrolla desde cero, sino que se emplea un modelo ya entrenado de la misma forma que en [14].

3 Diseño del sistema.

En este capítulo se exponen los elementos, tanto a nivel de hardware como de software, que componen el sistema. A su vez, se realiza una explicación de cada uno de estos elementos y como interaccionan entre ellos.

3.1 Arquitectura del sistema.

El sistema propuesto en este trabajo se compone de dos elementos principales: el desarrollo de un sistema de navegación para un drone en un entorno de un invernadero de floricultura basado en visión artificial; y un algoritmo de detección del estado fenológico de las flores, compuesto por una segmentación de la imagen a partir de visión artificial y posteriormente una clasificación de las imágenes extraídas a través de Machine Learning.

El sistema de navegación propuesto consiste en un algoritmo de seguidor de línea a través del procesamiento de imagen mediante visión artificial de la cámara cenital con la que cuenta el drone. Este algoritmo cuenta con los lazos de control necesarios para asegurar que el movimiento se mantiene dentro de la línea en todo momento y a una altura determinada. Adicionalmente al sistema de navegación, para conseguir un mayor control y evitar posibles colisiones, se desarrolló un sistema de detección de obstáculos basado en sensores de ultrasonidos.

Todos los procesos desarrollados para la navegación del drone interaccionan entre ellos mediante el ecosistema de ROS, el cual se explicará más adelante.

El sistema está compuesto a nivel de hardware por:

- Drone parrot AR Drone 2.0 y su versión simulada.
- Placa de desarrollo ESP32.
- Sensores de ultrasonidos HC-SR04

Y a nivel de software está compuesto por:

- Ecosistema ROS/GAZEBO.
- Sistema de navegación basado en visión artificial, así como sus elementos de control.
- Algoritmo de segmentación de imagen para obtener las fotos de las flores basado en visión artificial.
- Algoritmo de clasificación de imágenes basado en Deep Learning.

3.2 Arquitectura hardware.

La arquitectura hardware del sistema tiene como elemento fundamental el drone Parrot AR. Drone 2.0. Adicionalmente, el sistema sobre el que se ha trabajado cuenta con una placa de desarrollo de tipo ESP32, así como cuatro sensores de ultrasonidos HC-SR04. También ha sido necesario utilizar un ordenador con el sistema operativo Ubuntu 16.04, con los paquetes y librerías correspondientes, para ejecutar los programas realizados y establecer la comunicación entre todos los elementos.

3.2.1 Parrot AR Drone 2.0.

El parrot AR. Drone 2.0. (Figura 7) es un drone de bajo coste con un precio de mercado en torno a 300 euros. Cuenta con unas dimensiones de 58.23 x 58.23 cm y un peso de 420 gr. El armazón está compuesto de fibra de vidrio lo que le aporta una gran resistencia contra impactos. Este modelo cuenta además con una batería de 1000 mAh, lo que le proporciona una autonomía de vuelo aproximada a los 10 minutos.

Figura 7. Drone Parrot AR. Drone 2.0.

A nivel de sensores, el dispositivo cuenta con dos cámaras con una resolución de 720p (1280x720 píxeles) a 30 FPS (Fotogramas por segundo con sus siglas en inglés), una de ellas colocada en el frente y la otra en la parte inferior. Cuenta además con un sensor de ultrasonidos con el que se es capaz de medir la altura a la que se encuentra el drone, un giroscopio, magnetómetro y acelerómetro de tres ejes con los que se es capaz de obtener información sobre la odometría. En nuestra aplicación se hace uso de ambas cámaras, la frontal para tomar las imágenes de los frutos y la cenital con el fin de desarrollar un sistema de navegación, al igual que se emplea el sensor de ultrasonidos para controlar en todo momento la altura. Adicionalmente, el drone cuenta con conectividad WiFi, lo que permite establecer la conexión entre este, la placa ESP32 y el ordenador con el fin de controlar el movimiento y ejecutar los diferentes programas y directivas. A nivel dinámico, un quadrotor cuenta con 6 grados de libertad, ya que es capaz de desplazarse libremente en un entorno 3D. Esto grados de liberta

son el desplazamiento lineal sobre los ejes X, Y, Z y el movimiento angular sobre los ejes X, Y, Z, representado por los ángulos *roll*, *pitch* y *yaw*. En la Figura 8 se pueden observar estos ángulos y sobre el quadrotor.

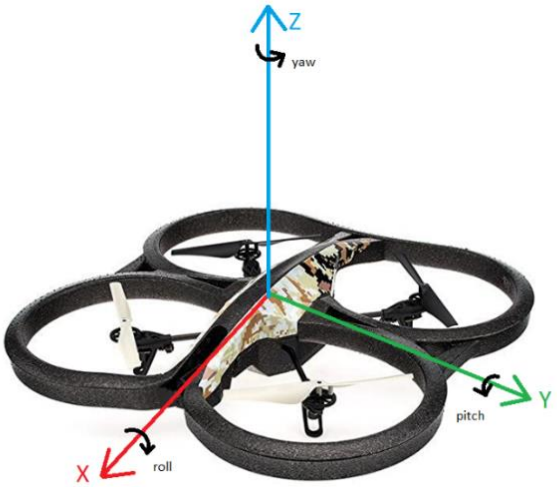

Figura 8. Ejes de referencia del drone con ángulos de giro indicados.

Para que se produzca el movimiento del drone sobre el espacio es necesario controlar la velocidad vertical y el giro sobre cada uno de los ejes *roll, pitch* y *yaw*. Con la combinación de diferentes comandos de velocidad a los rotores se puede conseguir la rotación sobre estos ejes. En la Figura 9 se puede ver esto.

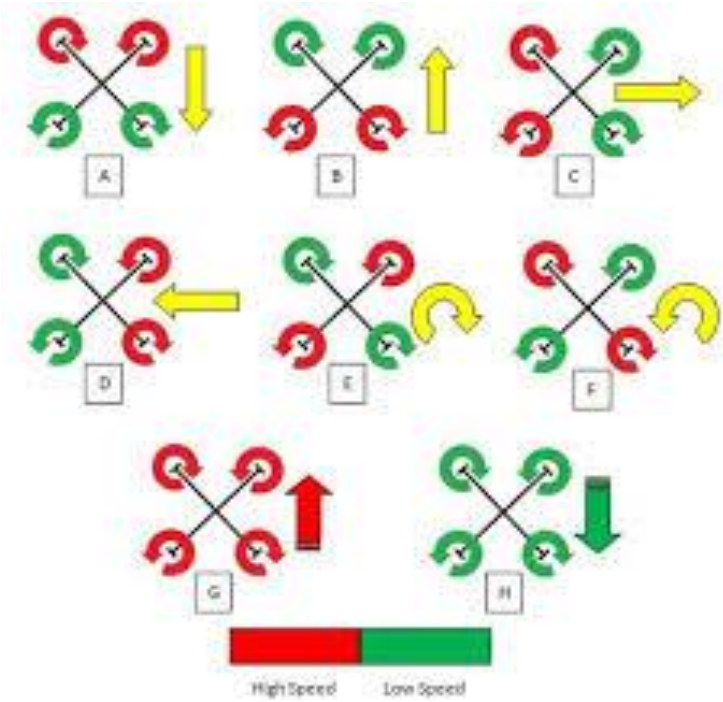

Figura 9. Movimiento del quadrotor en función del giro de los rotores.

Para este caso, no ha sido necesario controlar de forma independiente cada uno de los motores puesto que el controlador que se ha utilizado transcribe el comando de velocidad en una dirección del espacio en los comandos necesarios a aplicar a cada rotor. Debido a las dificultades por el contexto de la pandemia del Covid-19 se ha utilizado durante toda la etapa de desarrollo un modelo simulado de este drone (Figura 10). Este modelo se empleó en el simulador GAZEBO, el cual está incluido como simulador en el entorno de ROS utilizado y del cual se hablará más adelante.

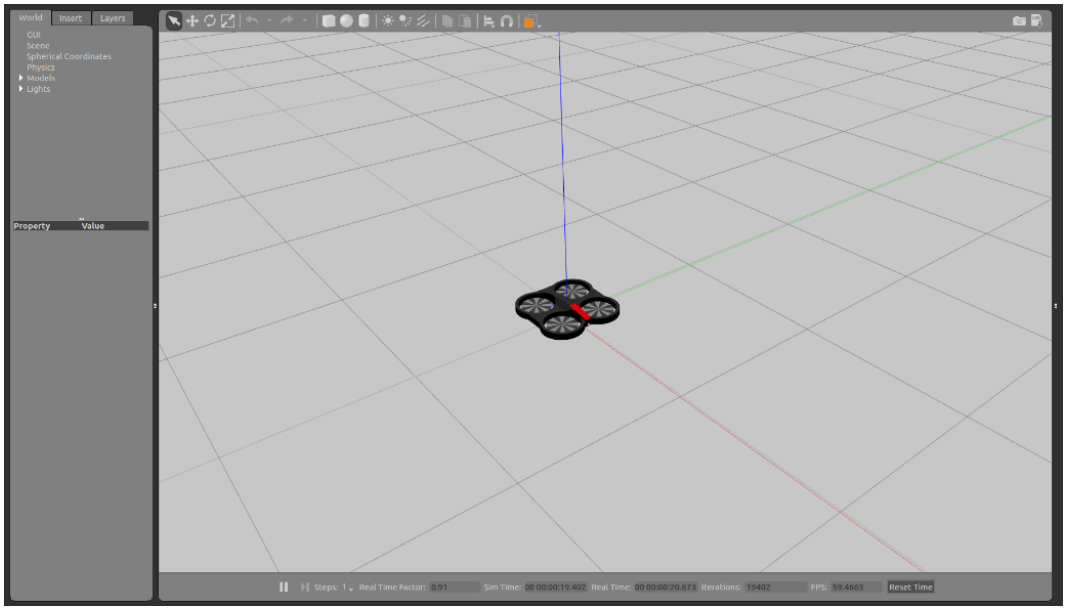

Figura 10. Simulador GAZEBO

3.2.2 Placa de desarrollo ESP32.

Esta placa de desarrollo utilizada cuenta con un chip ESP32 desarrollado por Espressif Systems. Este chip es del tipo SoC (Sistema en Chip por sus siglas en inglés) y hace referencia a que integra todos los módulos en un mismo chip.

En el caso de este (Figura 11), cuenta con un microprocesador Tensilica Xtensa LX6 con una velocidad de 160 MHz. Incluye un módulo dual de WiFi y Bluetooth 4.0. que, junto con una gran cantidad de pines de entrada y salida, denominados GPIO (Entrada y salida de propósito general por sus siglas en inglés), interfaces de tipo UART, I2C y SPI entre otras le aportan gran flexibilidad.

Figura 11. Placa de desarrollo ESP32.

Esta placa se puede programar de forma similar a Arduino utilizando el entorno de desarrollo de este. Esto sumado a su bajo coste, en torno a 10 euros, ha hecho que se haya convertido en un elemento imprescindible en los entornos *maker* para desarrollo de dispositivos de IoT (Internter de las cosas, por sus siglas en inglés).

En nuestro caso, se ha utilizado esta placa para implementar un sistema de sensores de ultrasonidos con el fin de poder detectar objetos cercanos al drone durante el vuelo y evitar colisiones.

3.2.3 Sensor ultrasonidos HC-SR04

Los sensores de ultrasonidos que se han utilizado en esta aplicación son del tipo HC-SR04 (Figura 12. Sensor ultrasonidos HC-SR04.). Estos sensores a pesar de su bajo coste otorgan una medida de distancia con una precisión lo suficientemente alta como para que su uso sea extendido en multitud de proyectos. Por ello, constituye un sensor altamente utilizado con placas de la familia Arduino.

En el caso de la placa ESP32 utilizada la compatibilidad es total. El rango de medida de distancia de estos sensores está entre los 2 y los 400 cm con una resolución de 0.3 cm, aunque en la práctica se ha comprobado que el rango máximo de medida está en torno a los 170cm. El rango de medida angular es de entre 15 y 20 grados en el eje horizontal y en torno a 4 grados en el eje vertical. Este sensor necesita una alimentación de 5 V DC y tiene un consumo menor a 2 mA.

Figura 12. Sensor ultrasonidos HC-SR04.

El funcionamiento de medida de este sensor consiste en el envío de un pulso de alta frecuencia no audible por el ser humano (*trigger*). Este pulso rebota sobre la superficie del objeto del cual se quiere medir la distancia al sensor y es captado de nuevo por el sensor (*echo*). La medida de distancia se obtiene de forma indirecta a partir del tiempo transcurrido entre el instante en el que se envía el pulso y el momento en el que se recibe el rebote.

A partir de la siguiente formula se obtiene la distancia en centímetros a partir del tiempo en microsegundos:

$$d = c \cdot 10^{-4} \cdot \frac{t}{2}$$

Donde, *d* es la distancia en cm, *c* es la velocidad del sonido en m/s a 20˚C, y *t* es el tiempo transcurrido entre que se envía el pulso y se recibe el rebote en µs.

Como se ha mencionado anteriormente, en este trabajo se han utilizado cuatro de estos sensores colocados en el frente, parte trasera y ambos lados formando un anillo de sonar, con el fin de detectar objetos cercanos al drone (Figura 13). Estos sensores han sido conectados a la placa ESP32 compartiendo el mismo pin para enviar el pulso de medida (*trigger*) y utilizando pines de entrada configurados como interrupciones para leer el pulso de rebote (*echo*).

A través de este método de conexionado se consigue por un lado eliminar el número de pines necesarios en la placa, se reduce de ocho a cinco, y por otro lado conseguir una lectura más rápida puesto que se miden los sensores de forma concurrente, en lugar de hacerlo de forma secuencial.

Figura 13. Montaje de sensores y placa ESP32 sobre drone real.

Al igual que con el drone, durante la etapa de desarrollo se ha utilizado una versión modelada de estos sensores, los cuales se incluyeron sobre el modelo del drone en el simulador gazebo (Figura 14). Para ello fue necesario modificar el archivo del modelo 3D del drone e incluir estos sensores con las características de medida anteriormente mencionadas.

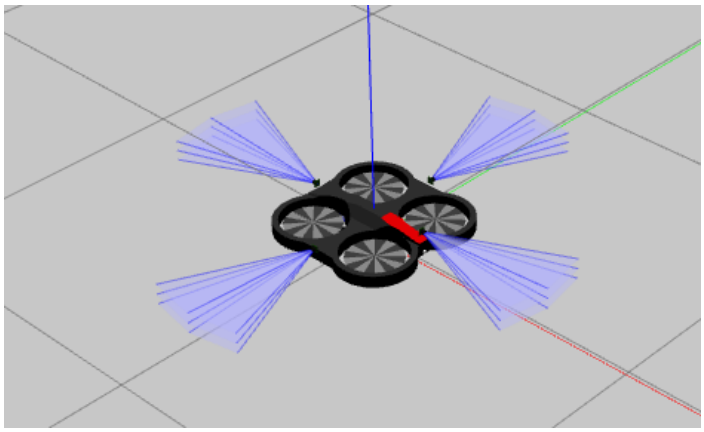

Figura 14. Sensores de ultrasonidos sobre drone simulado.

3.3 Arquitectura software.

La arquitectura software desarrollada en este trabajo consiste en dos elementos fundamentales. El primero de ellos constituido por el sistema de navegación en invernadero y el segundo de ellos por el sistema de reconocimiento y clasificación de las flores. El sistema de navegación está basado en un algoritmo de seguidor de línea utilizando segmentación de la imagen por color para detectar la línea que debe seguir y las diferentes marcas que definen las paradas y la zona de aterrizaje; este tratamiento de la imagen se realizó a través de las

librerías de OpenCV para visión artificial en su versión de Python 2.7. Además de esto, el sistema de navegación cuenta con lazos de control para controlar la posición en el plano horizontal así mismo como el giro y la altura. Adicionalmente se diseñó un sistema capaz de evitar la colisión del drone contra objetos. Todos estos elementos interactúan entre sí para dotar el robot de la capacidad de navegar por un entorno estructurado.

El sistema de reconocimiento y clasificación de flores está compuesto por dos elementos, el primero de ellos es el preprocesamiento de las imágenes tomadas por el drone mediante visión artificial, al igual que para la detección de la línea se utilizó OpenCV en su versión de Python 2.7, el cual consiste en una segmentación de la imagen por color con el objetivo de obtener en qué secciones de la imagen se encuentran flores y extraer recortes de estas secciones que posteriormente se clasificarán como “flor”, “no flor” o “flor madura”. Una vez la imagen ha sido segmentada, se clasifican los segmentos de esta mediante una red neuronal entrenada para dicho propósito. El entrenamiento de la red neuronal y el posterior uso de esta para realizar predicciones se realizó mediante TensorFlow en su versión para Python 3.6 mediante el *wrapper* Keras.

3.3.1 Ecosistema Gazebo/ROS.

El desarrollo del sistema de navegación propuesto en este trabajo y la interconexión con el sistema de reconocimiento y clasificación de imagen se produce sobre el framework de desarrollo ROS (Robotic Operating System). Este framework fue desarrollado en el año 2007 en la Universidad de Standford, con el objetivo de dotar a los usuarios con una serie de librerías, módulos y herramientas con los cuales poder afrontar de forma sencilla y robusta el desarrollo de aplicaciones de robótica en una gran variedad de plataformas. Todo esto con una mentalidad “Open Source”, lo que con el tiempo ha hecho que exista una gran comunidad de desarrolladores e investigadores a nivel mundial.

ROS se encuentra constituido en una estructura de tipo grafo, en la que los diferentes procesos que se ejecutan se denominan nodos. Estos nodos, para intercambiar información entre ellos y poder establecer el grafo de ROS, utilizan los topics, los cuales son buses de comunicación dentro del grafo. Los nodos pueden suscribirse a estos topics para recibir información o publicar para enviar información al topic. De esta manera, ejecutando diferentes nodos sobre un proceso maestro de ROS o roscore, se puede crear la estructura de la aplicación que se pretende desarrollar de forma modular.

En este trabajo, la distribución de ROS utilizada ha sido ROS kinetic, la cual se ejecuta sobre un sistema operativo Linux Ubuntu 16.04.

Como se ha mencionado anteriormente, durante la etapa de desarrollo se utilizó el simulador GAZEBO, el cual está incluido como herramienta dentro de la distribución de ROS anteriormente mencionada. Para realizar esta simulación se utilizaron los paquetes de ROS “ardrone_autonomy”, el cual proporciona todos los drivers

necesarios para el cuadricóptero Parrot utilizado, y el paquete "tum_simulator", el cual contiene la implementación de este drone en el simulador GAZEBO. Adicionalmente, fue necesario realizar un port de este último paquete a la versión kinetic para poder utilizarlo en la aplicación.

La principal ventaja de utilizar un simulador como GAZEBO es que se pueden realizar las pruebas necesarias para comprobar que el sistema está siendo adecuado sin la necesidad de contar con el robot que se pretende utilizar. Esto, por un lado, permite que el desarrollo sea seguro en el sentido de que no se pone en peligro la integridad física y material de los desarrolladores ni del robot, ni de ningún elemento utilizado, al mismo tiempo que permite que el desarrollo se pueda realizar de forma remota. Esto último ha sido de vital importancia para este trabajo debido a la situación de la pandemia del Covid-19.

3.3.2 Navegación del drone en el invernadero.

Como se ha mencionado anteriormente, el sistema de navegación está compuesto por un algoritmo de seguidor de línea mediante visión artificial, así como por los esquemas de control necesarios para el movimiento y de un sistema de evasión de colisiones.

1) Seguidor de línea.

La parte principal del sistema de navegación consiste en un algoritmo de seguimiento de línea basado en visión artificial. Se decidió optar por esta solución tras comprobar los buenos resultados obtenidos por diferentes investigaciones y la robustez que presenta con el sistema de control adecuado [15] [16].

La aplicación consistirá en que el drone, gracias a la cámara cenital con la que cuenta, tomará las imágenes de la línea sobre la que se desplaza y las procesará en tiempo real con el fin de obtener los comandos de velocidad que se deben de aplicar al robot para que este se mantenga sobre la línea. Esta aplicación no ha sido desarrollada desde cero, sino que se ha partido del trabajo realizado por los estudiantes del Instituto Tecnológico de Israel Yakov Korlansky y Shabi Sabatan [17].

El procesamiento de imagen en la aplicación se realiza mediante el uso de funciones que proporciona la librería OpenCV para procesamiento de imagen. Esta es una librería de código abierto ampliamente utilizada para visión artificial tanto en entornos de investigación como industriales. En el caso de este trabajo, se ha utilizado la versión para Python 2.7, aunque se encuentra disponible para otros lenguajes de programación como C++, Visual C o C#. Puesto que el color de los elementos a detectar es la característica principal que los diferencia los unos de los otros, la aplicación desarrollada consiste en una segmentación por color de la imagen con el fin de diferenciar los diferentes elementos.

Dentro de los múltiples espacios de color en los que se puede representar una imagen, el espacio de color HSV (Matiz, saturación y valor por sus siglas en inglés) es el más indicado para realizar segmentación de color. En

este espacio de color, del cual se puede ver una representación gráfica en la Figura 15, a diferencia del espacio de color RGB (Rojo, verde y azul por sus siglas en inglés), los colores vienen definidos por su componente de matiz, representado a través de un ángulo dentro de una rueda de color tomando como referencia el color rojo, el valor de saturación, que indica la lejanía del color al centro de esta rueda de color, y el valor, que representa la cantidad de color negro o blanco en el color [18].

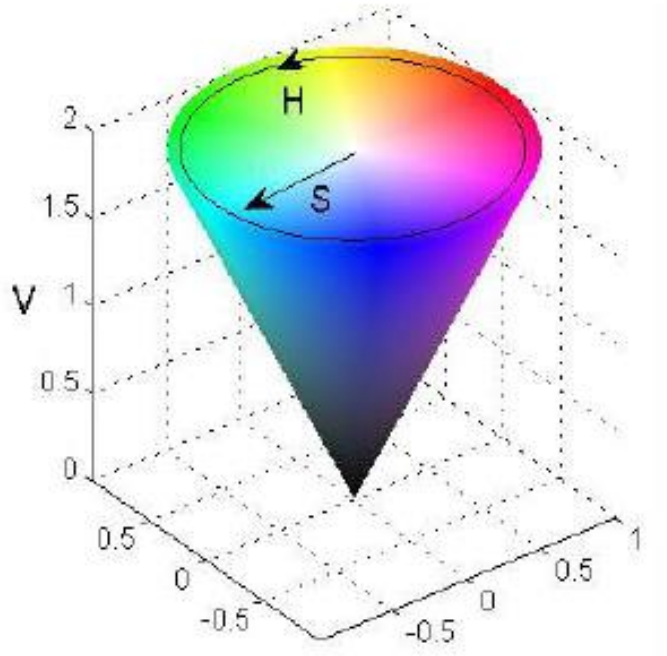

Figura 15. Representación espacio de color HSV.

La principal ventaja de este espacio de color y el motivo por el que se usa en segmentación de color es que, puesto que el color viene definido por una componente angular, se pueden establecer umbrales de detección para una sección concreta del espacio de color de forma sencilla indicando el intervalo angular de matiz para dicho color, así como los valores máximos y mínimos de saturación y valor.

La biblioteca OpenCV proporciona dos funciones con las que se puede realizar la conversión y segmentación de color de forma sencilla. Estas funciones son *cv2.cvtColor* con la directiva *COLOR_BGR2HSV*, para realizar la conversión entre espacios de color, y la función *inRange* para realizar la segmentación entre dos intervalos de color. Dentro de la librería OpenCV, los valores de matiz, saturación y valor están comprendidos entre 0 y 180 para el matiz, y entre 0 y 255 para la saturación y el valor.

En la Figura 16 se puede observar una representación en dos dimensiones de los valores de matiz y saturación según los valores que acepta OpenCV.

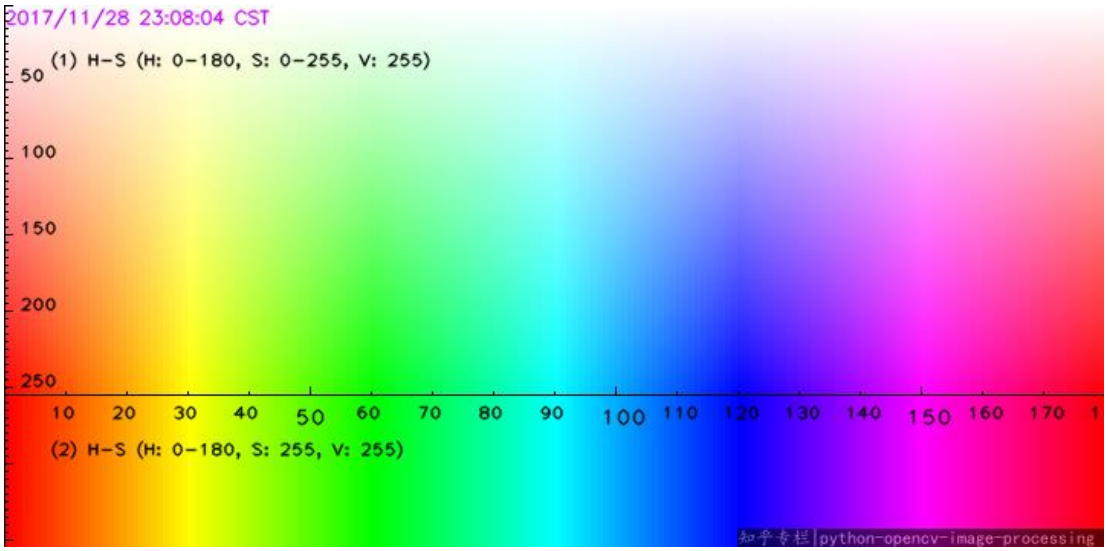

Figura 16. Rango de valores para matiz y saturación en OpenCV.

Una vez realizada la segmentación por color fue necesario obtener la posición y orientación relativa de la línea, y de las marcas de señalización relativas a la posición del drone. Gracias a la función *findContours* se realizó esta tarea. Una vez obtenidos los valores de posición y ángulo que se deben corregir, se puede realizar el control de posición y orientación.

2) Esquema de control

Para asegurar que el drone se encuentra en la posición deseada en todo momento ha sido necesario establecer un esquema de control en lazo cerrado que fuese capaz de asegurar esto. Para ello se tomaron en cuenta dos situaciones de movimiento distintas y se desarrollaron lazos de control adecuadas a cada una de ellas.

La primera situación es aquella en la cual el drone está siguiendo la línea (Figura 17). En esta situación es necesario controlar la posición horizontal en el eje Y y vertical sobre el eje Z, y el ángulo de giro sobre el eje Z (yaw). El control de la velocidad sobre el eje X no es necesario puesto que este valor es el asignado por el usuario y será la velocidad con la cual se desplazará sobre la línea.

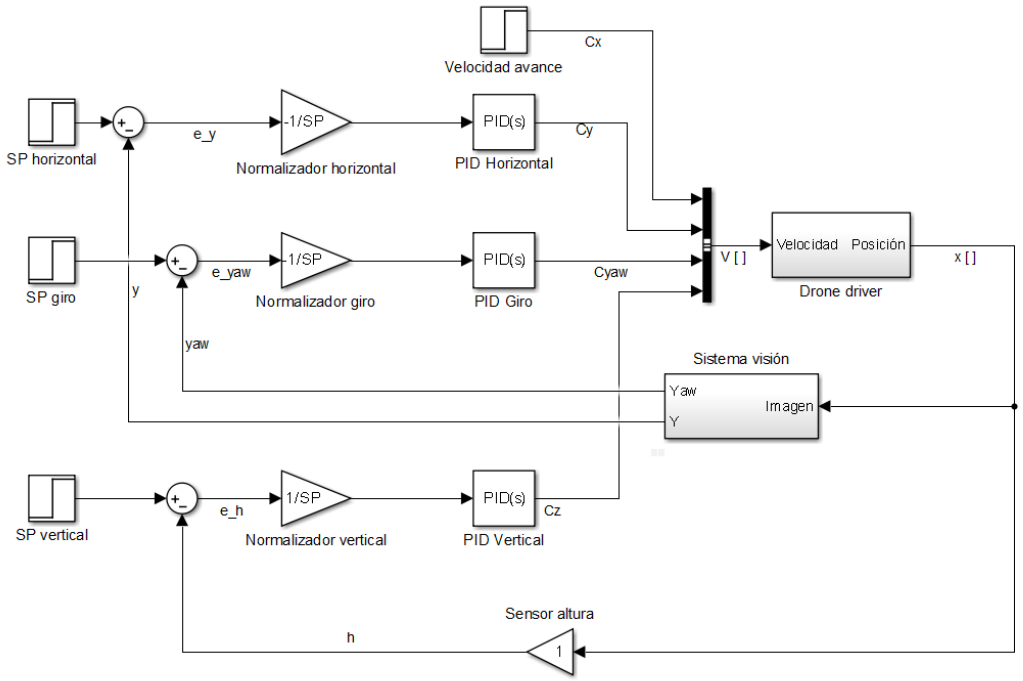

Figura 17. Esquema de control del drone durante el seguimiento de la línea.

Como se puede observar en la Figura 17, se normaliza el valor del error antes de pasar al controlador. Esto es necesario puesto que el controlador del drone acepta comandos de velocidad entre -1 y 1, siendo estos comandos un porcentaje de la velocidad máxima del robot. El signo hace referencia al sentido de la velocidad.

Una vez se ha aplicado el comando de velocidad, el drone modifica su posición de forma conveniente. Para establecer el lazo cerrado es necesario tomar la realimentación negativa midiendo las variables necesarias y calculando el error entre el valor de consigna y estas. En el caso del control horizontal, tanto el valor de consigna como el de posición medida tienen unidades de píxeles. Esto es debido a que se establece como valor de consigna el centro de la imagen en el eje horizontal y como valor medido el centroide del contorno de la línea detectada. De esta manera, el controlador hará que la línea se mantenga en el centro de la imagen. Para el control de giro, el valor de consigna y el ángulo medido estarán en grados. En este caso se establece en la consigna un valor de 0 grados, de esta manera se asegura que la línea detectada se muestre en la imagen como una línea vertical. Para el control de la altura, tanto la consigna como el valor medido es en metros.

La segunda situación de estudio es aquella en la que el drone se encuentra sobre las paradas y debe realizar la toma de fotografías (Figura 18). En esta situación, el esquema de control es muy similar a la situación anterior con la excepción de que también resulta necesario controlar la posición sobre el eje X, puesto que el drone se debe mantener estático sobre la marca.

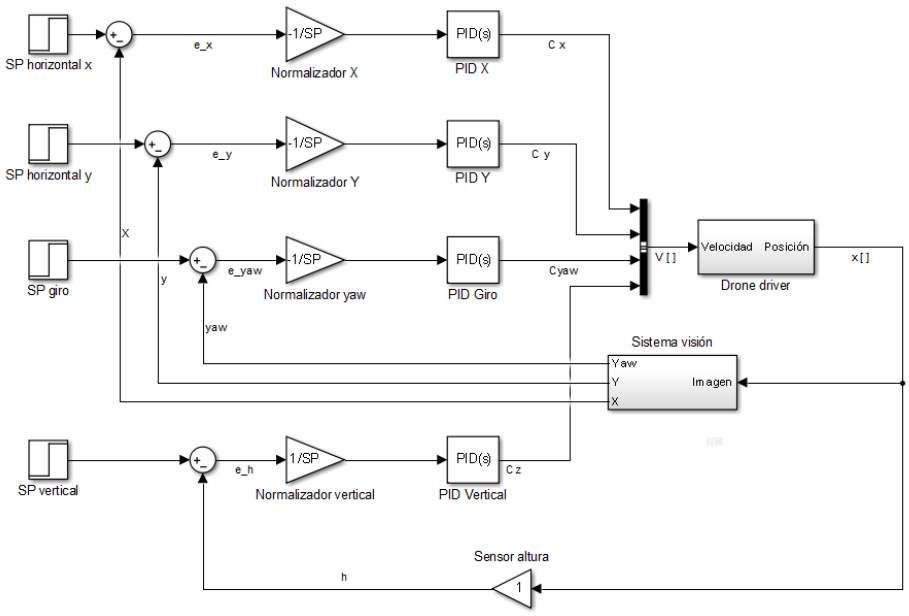

Figura 18. Esquema de control del drone al posicionarse sobre marcas de detección.

En ambas situaciones, el lazo de control de giro y el lazo de control para la altura se mantienen idénticos. Es el lazo de control horizontal el que modifica sus valores de una situación a otra. Aunque el controlador de posición horizontal varíe de una situación a otra, en la segunda situación se presenta el mismo controlador tanto para la posición sobre el eje X como sobre el eje Y. En el siguiente capítulo se justifica la elección de las variables de cada controlador.

3) Mecanismo de seguridad frente colisiones

Con el fin de dotar al sistema de una capa extra de seguridad y así tener la capacidad de evitar colisionar contra un objeto que se encuentre en la marcha, se utilizaron los sensores de ultrasonidos anteriormente mencionados y la placa de desarrollo ESP32. Para la detección de obstáculos se establecieron dos umbrales de detección, el primero de ellos establece el límite en el cual el sistema de control de colisiones toma el control de la velocidad para evitar una colisión; en el segundo umbral, se produce el aterrizaje del drone para evitar una colisión inminente.

Una vez que el sistema toma el control de la velocidad del drone, este lo mantiene a la distancia establecida en el primer umbral hasta que el objeto desaparezca de su trayectoria. Se toma la lectura de los cuatro sensores y, en función de cuales de ellos hayan entrado en la zona delimitada por el umbral superior, se genera el comando de velocidad adecuado para mantener al drone a la distancia de seguridad. Cuando el objeto que se encontraba

en la trayectoria del drone desaparezca de esta, el sistema de navegación vuelve a tomar el control de la velocidad y continua por la línea.

Para obtener la medida de distancia de los sensores tanto en la versión simulada como en la versión real se utilizaron dos técnicas diferentes. Para tomar la medida en la simulación, únicamente fue necesario indicar en los atributos del modelo 3D del sensor en qué *topic* debía publicar la información y el tipo de mensaje.

Para tomar la medida en la versión no simulada de los sensores, primero de todo hubo que programar la placa ESP32 para leer los 4 sensores y que publicase la información en los *topics* correspondientes. Para establecer la comunicación entre la placa ESP32 y el proceso maestro de ROS se utilizó un paquete denominado *rosserial*, el cual tiene dependencias para establecer una comunicación con un sistema de tipo Arduino tanto por conexión serie como por conexión inalámbrica mediante WiFi. Este último es el método de comunicación utilizado ya que la placa de desarrollo empleada permite conexión mediante WiFi. Para establecer la conexión mediante WiFi es necesario, primero de todo, iniciar un nodo de tipo *rosserial* utilizando el protocolo de comunicación TCP/IP sobre un proceso maestro de ROS; una vez este nodo ha sido iniciado se debe indicar en la programación de la placa ESP32 cual es la dirección IP del equipo en el que se está ejecutando el proceso maestro de ROS, así como el puerto de conexión del servidor de *rosserial*. La dirección IP se puede conocer consultando la información de la red WiFi del equipo donde se ejecuta el proceso maestro, y el puerto de conexión está establecido por defecto como el 11411.

3.3.3 Segmentación de imagen por color.

Como paso previo a la clasificación de las imágenes mediante Deep Learning, es necesario extraer de las instantáneas que toma el drone las secciones de la imagen en las que se encuentran las flores. Esto se ha realizado a través de un algoritmo de segmentación de imagen por color mediante el uso de OpenCV.

Al igual que para el reconocimiento del camino que se ha seguir en el sistema de navegación, se ha trabajado con el espacio de color HSV para realizar la segmentación por color. Como se ha mencionado anteriormente, resulta más sencillo establecer umbrales de detección en este espacio de color que en otros como el RGB, puesto que, para un tono de color concreto todas sus variaciones en términos de iluminación y saturación se encuentran sobre el mismo valor de matiz.

Previo a realizar la umbralización de la imagen se aplica un filtro de la media sobre la imagen. Este filtro se aplica con dos objetivos: el primero de ellos, eliminar el posible ruido que esté presente en la imagen; el segundo es difuminar la imagen con el fin de, por un lado, homogeneizar el color en los elementos que se encuentran más cercanos a la cámara en la escena, y por otro lado para difuminar las características de los elementos más alejados en la escena. De esta forma, tendrán más relevancia a la hora de umbralizar los elementos más

cercanos. Este filtro de la media se aplica sobre la imagen mediante una operación de convolución. Para ello es necesario definir cuál será la máscara del filtro a utilizar. En el caso del filtro de la media unidimensional, la máscara se definirá como una matriz de 2 dimensiones de N elementos en filas y columnas donde cada uno de dichos elementos tendrá el valor de la unidad dividido entre N elevado al cuadrado.

La operación de convolucionar la imagen con la máscara del filtro de la media (Figura 19) resulta en que, en un espacio de vecindad para un píxel definido por la dimensión N de la matriz, estando la matriz centrada sobre dicho píxel, se calcula el valor medio de todos los píxeles dentro de ese espacio de vecindad. Dicho valor medio es el valor que tomará el píxel original una vez se haya realizado la operación de convolución. De esta manera es posible eliminar ruido en la imagen puesto que el ruido serán píxeles en la imagen con un valor muy dispar al de sus píxeles vecinos. De esta misma manera, los bordes suponen cambios bruscos en el valor de los píxeles de una imagen donde se encuentran estos bordes, calculando la media en estos, se consigue el suavizado de estos bordes y la difuminación de la imagen [19].

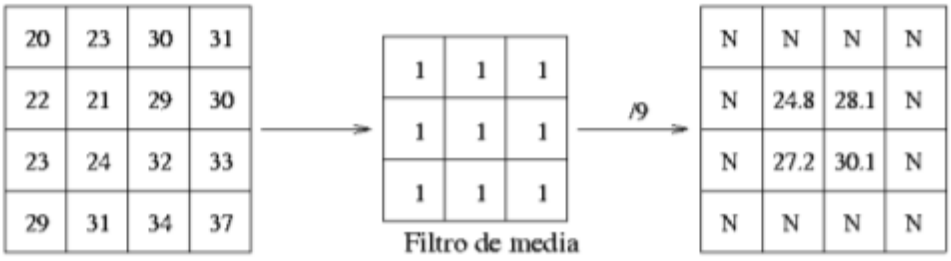

Figura 19. Filtro de la media.

Es necesario resaltar que, al aplicar cualquier técnica de filtrado mediante convolución, las dimensiones de la imagen original se ven reducidas puesto que en los extremos horizontales y verticales de la imagen no es posible aplicar la máscara ya que esta no solapa totalmente una sección completa de la imagen. Como estos píxeles cuentan con un valor indeterminado tras la operación de convolución, son retirados de la imagen.

Una vez se ha realizado el filtrado y la umbralización de la imagen, se procede aplicando dos operaciones morfológicas, una de apertura y otra de cierre. De esta manera, con la operación de apertura se eliminan posibles artefactos que aparezcan en la imagen pertenecientes al fondo. Con la operación de cierre se consiguen unir secciones de la imagen cercanas y que conforman un mismo elemento, pero han sido detectados como dos elementos distintos. Estas operaciones resultan de la combinación de dos operaciones morfológicas denominadas como erosión y dilatación (Figura 20). En la operación de erosión se eliminan píxeles del contorno de un elemento aplicando un elemento estructural sobre el perímetro de este. La operación de dilatación es la

opuesta, se añaden píxeles sobre el contorno cuando este elemento estructural lo recorre. La operación de apertura está conformada por una erosión seguida de una dilatación, mientras que la operación de cierre está compuesta por una operación de dilatación seguida de una de erosión [20].

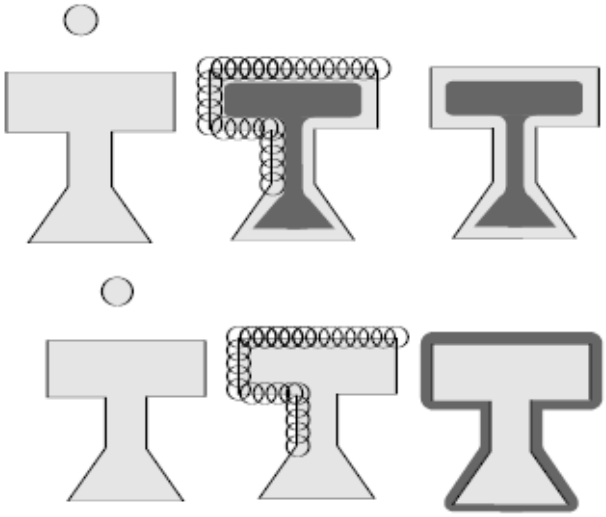

Figura 20. Operaciones morfológicas de erosión y dilatación.

Habiendo realizado todas las transformaciones a la imagen se extraen los contornos de la imagen resultante mediante la función *cv2.findContours* de la biblioteca OpenCV. Teniendo estos contornos, se establece un umbral de detección basado en el área de estos para determinar si es un elemento del fondo que haya podido no ser eliminado en las operaciones anteriores, o es un elemento perteneciente a una flor. Posteriormente se recuadran sobre la imagen original los elementos detectados y se realizan los recortes de estos elementos para posteriormente realizar la clasificación.

3.3.4 Técnicas de Machine Learning usadas.

Como se ha mencionado anteriormente, el modelo Deep Learning utilizado en este trabajo tiene como finalidad clasificar las imágenes dadas en función a unas etiquetas determinadas. Para la realización de esta tarea no se desarrolló un modelo desde cero, sino que se tomó un modelo ya entrenado en una base de datos muy amplia y se aplicó al problema propuesto; esto es lo que se conoce como Transfer Learning. Así mismo, para aumentar la cantidad de imágenes de entrenamiento se recurrió a técnicas de aumento de datos (data augmentation en inglés). Esto consiste en aplicar transformaciones morfológicas aleatorias, tales como rotaciones, translaciones, zoom, etc., a las imágenes con el fin de aumentar la cantidad de estas y así evitar que durante el proceso de entrenamiento el modelo aprenda cual es el orden de las imágenes y dotar de mayor robustez a este.

1) Métodos de Deep Learning.

1. Image Classification:

Como se ha mencionado anteriormente, el método utilizado para el reconocimiento de las flores en las imágenes es mediante una clasificación. Este es uno de los enfoques más utilizados en el reconocimiento de imágenes (como se ha observado en el capítulo 2 de este trabajo), desde el reconocimiento de frutos, hasta la detección de enfermedades pulmonares. La clasificación de imágenes consiste en asignar una de las etiquetas, previamente definidas por el usuario, a una imagen dada al modelo. Esta asignación no se realiza de forma absoluta, sino que el modelo realiza una predicción estadística basada en el valor de los pesos y sesgos de cada una de las neuronas de la RNA.

De esta manera, si el modelo tiene como salida tres posibles etiquetas, la salida producida por una imagen será la probabilidad de pertenecer a cada una de las etiquetas. Con un modelo bien entrenado, la etiqueta a la pertenece la imagen de entrada tendrá un valor de confianza cercano a la unidad, mientras que el resto de los valores probabilísticos tenderán a 0. Para obtener resultados capaces de clasificar las imágenes de forma adecuada, uno de los factores más importantes es la base de datos sobre la que se entrene el modelo. Cuanto más amplia y variada sea esta base de datos, el entrenamiento resultará más efectivo. Sin embargo, cantidad de ejemplos de entrenamiento no siempre está relacionado con unos resultados altos de predicción. Estos datos tienen que ser de calidad y resultar representativos de los objetos a clasificar.

2. Overfitting y data augmentation:

Se conoce como overfitting (sobre entrenamiento en inglés) a la situación en la cual, un modelo que ha sido entrenado a partir de una base de datos presenta una gran disparidad en la precisión entre los datos de entrenamiento y datos que no ha visto antes. Es decir, no se alcanzan los mismos resultados con datos nuevos que con los datos con los que ha sido entrenado el modelo. Esto ocurre cuando los datos de entrenamiento resultan escasos y se utilizan los mismos datos durante diferentes ciclos de entrenamiento. Esto resulta en que el modelo ajusta sus pesos y sesgos internos para producir una salida concreta ante un dato concreto. Es decir, el modelo "memoriza" los datos de entrada. Cuando esto ocurre, el modelo es incapaz de extraer las generalidades que caracterizan a los datos de entrenamiento [21].

Una de las técnicas más populares para solucionar el problema del overfitting en tratamiento de imagen es utilizar la técnica de data augmentation (aumento de datos en inglés). Aplicado esta técnica a imágenes permite, a partir de una imagen generar nuevas imágenes aplicando transformaciones morfológicas tales como rotaciones, translaciones, zoom o reflexión de la imagen. Esto provoca que, aunque a ojos humanos sea la misma imagen, para un ordenador esta sea una imagen totalmente distinta en la que se mantienen todas las características de la imagen original y permite al modelo extraer las generalidades necesarias.

Aplicando estas transformaciones de forma aleatoria durante el entrenamiento tendremos que, a partir de un número reducido de imágenes de entrenamiento se entrene al modelo siempre con imágenes distintas generadas automáticamente [22]. En [23], Taylor et al. demuestran la eficacia del uso de técnicas de data augmentation para mejorar el resultado de diferentes tareas realizadas con Deep Learning aplicadas al tratamiento de imagen.

3. **Transfer Learning:**

   En Deep Learning, el concepto de Transfer Learning hace referencia a la utilización de un modelo pre-entrenado para reconocer patrones en una base de datos para lo que no ha sido entrenado. Estos modelos, al haber sido entrenados en grandes bases de datos y contar con millones de parámetros, resultan muy efectivos para reconocer patrones tales como bordes, color y formas. Esto permite que, modificando la capa de salida de este modelo, con el fin de adecuarla a la aplicación sobre la que se pretende utilizar, y con poco tiempo de entrenamiento, únicamente entrenando esta última capa, se puedan obtener resultados con una precisión muy alta. Así mismo, en estos modelos se pueden reentrenar ciertas capas para mejorar aún más la respuesta. Esto último es lo que se cómo fine-tuning, o ajuste preciso [24]. Para el tratamiento de imagen, gran parte de los modelos pre-entrenados utilizados en transfer learning han sido entrenado sobre la base de datos ImageNet. ImageNet es una gran base de datos de imágenes, con cerca de 50 millones de imágenes distribuidas en más de 20 mil categorías. Durante años, diversos grupos de investigación y expertos en Deep Learning han presentado modelos cada vez con un mayor grado de precisión sobre esta base de datos. En el año 2012, la red neuronal AlexNet revolucionó la competición alcanzando una tasa de error del 15.4% [25]. Esto produjo que en los consecutivos años se hayan desarrollado modelos que presentan una tasa de error menor a esta. Algunos de esto modelos son el Inception y MobileNet, los cuales se usarán en el presente trabajo.

   El uso de Transfer Learning está muy presente en el ámbito de investigación para clasificación de imagen. Un ejemplo de estos es el modelo de detección de cáncer de piel propuesto por Hosny et al. en [26] con una tasa de precisión entre el 97,73% y el 98,61%. Otro ejemplo sería el propuesto por Wang et al. en [27] para la detección de enfermedades pulmonares utilizando el modelo Inception mencionado anteriormente y con una tasa de acierto del 95,41% y el 80,09%.

2) Tensorflow.

Para poder desarrollar el modelo e implementar todas las técnicas vistas anteriormente se ha utilizado la plataforma TensorFlow en su versión para Python3.6. TensorFlow es una plataforma de código abierto que permite la creación de modelos para aprendizaje automático. Originalmente fue desarrollada por Google y actualmente está muy extendido su uso en el desarrollo de aplicaciones de Machine y Deep Learning. TensorFlow permite desarrollar código sobre Python y C++. Además, cuenta con diferentes versiones, como la

de CPU, la versión que utiliza la tarjeta gráfica para el procesamiento en paralelo, y TensorFlowLite, desarrollada especialmente para su uso en dispositivos de bajo rendimiento como puede ser el caso de Raspberry Pi [28].

Así mismo, para desarrollar modelos de Deep Learning sobre Python utilizando TensorFlow, es necesario utilizar la Interfaz de Programación de Aplicaciones (API) Keras. Gracias a Keras se pueden definir, entrenar y realizar predicciones de forma sencilla. A su vez, se podrá realizar *data agumentation* de forma sencilla gracias a la clase *ImageDataGenerator* incluida en las bibliotecas de Keras [29].

Para el caso de realizar *Transfer Learning*, es necesario utilizar *TensorFlow Hub*. Esto es un repositorio de modelos pre-entrenados los cuales carecen de su capa final, lo que facilita la integración de estos en las aplicaciones desarrolladas. Una de las grandes ventajas que aporta el uso de TensorFlow es la posibilidad de guardar el modelo que se haya creado con la posibilidad de utilizarlo sin necesidad de realizar el entrenamiento cada vez, o poder usarlo en otras plataformas.

3.4 Integración de todos los componentes.

Puesto que el sistema de navegación ha sido desarrollado sobre el entorno de ROS, la integración y comunicación de todos los elementos que lo conforman estará establecida en forma de grafo (Figura 21).

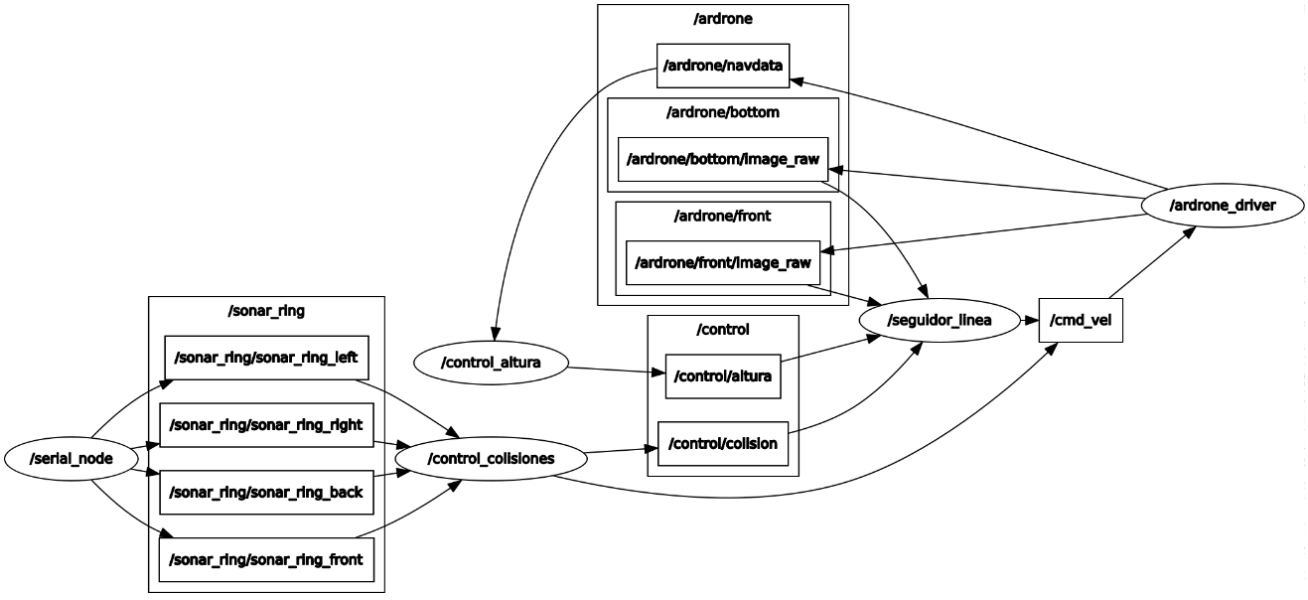

Figura 21. Grafo de ROS de la aplicación.

Los nodos principales de este grafo son: */ardrone_driver* nodo encargado de enviar los comandos de velocidad al drone y de realizar la lectura de las cámaras frontal y cenital, así como del sensor de ultrasonidos para conocer la altura. El otro nodo principal de la aplicación es */seguidor_linea*. En este nodo se encuentra tanto el algoritmo de seguimiento de línea como los lazos de control para el desplazamiento horizontal y el giro. Así mismo, es el nodo encargado de tomar las imágenes de las flores.

Para establecer el control de la altura, se emplea el nodo */control_altura*. Este es el encargado de tomar la lectura del sensor de ultrasonidos que incorpora el drone a través del topic */ardrone/navdata*, de implementar el lazo de control y de transmitir el comando de velocidad al nodo */seguidor_linea* a través del topic */control/altura*.

Por último, para implementar el sistema de seguridad frente a colisiones, se establece la comunicación entre el entorno de ROS y la placa ESP32 a través del nodo */serial_node*. La información de lectura de los sensores de ultrasonidos se publica en los topics de */sonar_ring*. Por último, el nodo */control_colisiones* recibe esta información e implementa el algoritmo para evitar colisiones controlando la velocidad del drone y avisando al nodo */seguidor_linea* de que se va a producir una colisión a través del topic */control/colisión*. Adicionalmente, para tener una referencia visual del camino seguido por el drone en todo momento, se muestra una ventana emergente (Figura 22) con la imagen tomada por la cámara cenital de este y se dibuja sobre ella el contorno de la línea que se está detectando, así como los valores de error de posición y angular.

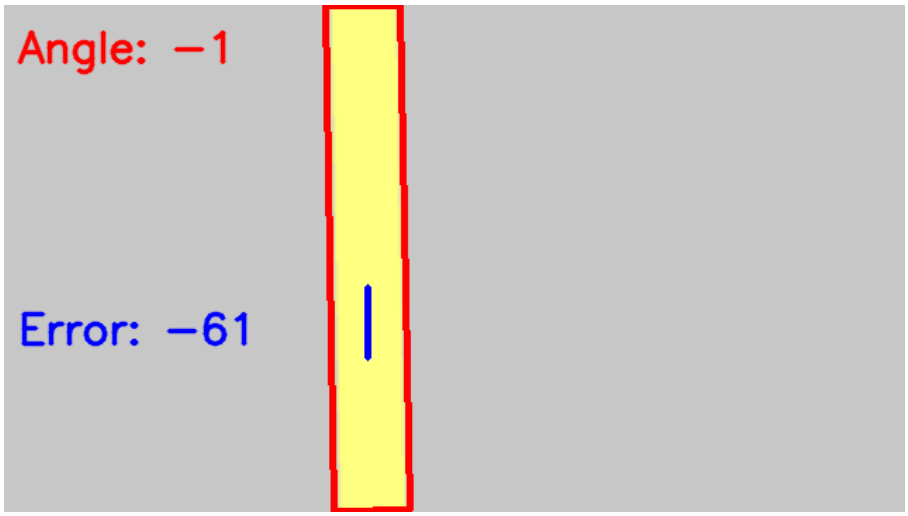

Figura 22. Interfaz de visualización para seguimiento de caminos.

Así mismo, una vez que el drone ha aterrizado tras realizar el recorrido, se imprime por pantalla el valor medio y la desviación típica de los errores de posición y angular con el fin de poder realizar un juicio de la precisión con

la que el robot ha sido capaz de recorrer el entorno. Referente al sistema de segmentación y clasificación de las imágenes, cuando se han extraído las regiones de interés dentro de la imagen, esta se muestra recuadrando los elementos detectados sobre la misma (Figura 23).

Figura 23. Ejemplo de flor detectada y recuadrada.

De forma similar, cuando el modelo realiza una predicción se muestra en pantalla la imagen junto con la etiqueta que tiene mayor valor de predicción junto a este valor (Figura 24).

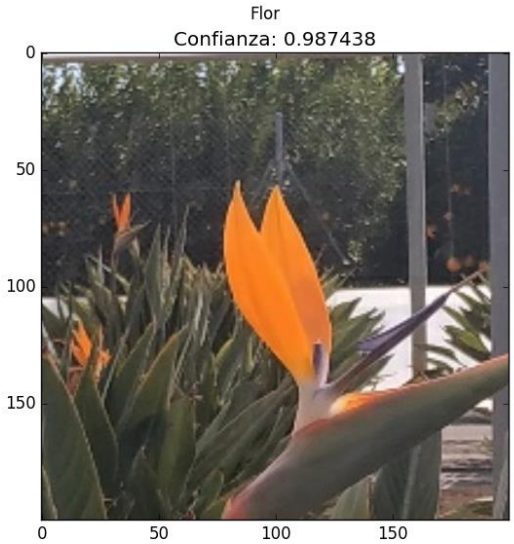

Figura 24. Muestra de clasificación de imagen.

4 Análisis de resultados.

En este capítulo se exponen las pruebas realizadas y los resultados obtenidos para validar el sistema propuesto. De igual forma, se realiza una discusión de los resultados obtenidos comparándolos con los presentados por otras investigaciones.

4.1 Pruebas diseñadas.

Con el fin de evaluar los sistemas desarrollados en el presente trabajo se realizaron una serie de pruebas tanto de forma independiente a cada módulo como de forma conjunta. Estas pruebas abarcan desde el ajuste de parámetros para la umbralización de la imagen por color hasta la comparación de diferentes modelos de Deep Learning, pasando por la sintonización de controladores o el estudio del efecto del *data augmentation* en la respuesta del modelo.

4.1.1 Pruebas de navegación.

A continuación, se exponen las pruebas llevadas a cabo para evaluar el sistema de navegación propuesto:

1) Detección de elementos claves para la navegación.

En esta prueba se evalúa el efecto de los factores ambientales en la segmentación de la imagen por color para la detección de los elementos claves de navegación tales como la línea a seguir, las marcas de las paradas y la zona de aterrizaje.

2) Sintonización de controladores para el movimiento del drone.

Para la sintonización de los controladores se recreó un entorno de prueba dentro del simulador GAZEBO con un recorrido el cual se utiliza para evaluar diferentes propuestas de controlador para el movimiento del robot.

3) Prueba de control de colisiones en entorno simulado.

Se recreó en un entorno simulado la situación de encontrarse con un obstáculo sobre el camino de la marcha para realizar el ajuste de los umbrales de detección de obstáculos.

4) Navegación en entorno simulado y real.

En esta prueba se evalúa la eficacia con la que el sistema propuesto es capaz de navegar por un recorrido determinado realizando las paradas pertinentes y aterrizando en la zona indicada. Se estudia tanto en un entorno simulado como en uno real.

4.1.2 Pruebas de reconocimiento de flores.

Para evaluar la eficacia del sistema de reconocimiento de flores se realizaron las siguientes pruebas:

1) Ajuste de parámetros en la segmentación por color.

En esta prueba se estudia el efecto del tamaño del filtro, el intervalo de umbralización y las operaciones morfológicas de cierre y apertura en la segmentación de la imagen para obtener las regiones de interés.

2) Corrección del overfitting mediante data augmentation.

En esta prueba se observa el efecto de técnicas de transformación de imagen para reducir el efecto del overfitting en la precisión del modelo de predicción.

3) Comparación de diferentes modelos para la clasificación de imágenes.

Para la selección del modelo a utilizar en la clasificación de imágenes se comparan los modelos *mobileNet_V2* e *Inception_V3* evaluando la precisión sobre dos bases de datos, una de ellas con una clasificación binaria entre "flor" y "no flor"; y la otra estableciendo tres posibles etiquetas de salida: "flor", "no flor" y "flor madura".

4) Aplicación del modelo desarrollado a la clasificación sobre imágenes nunca vistas.

Por último, para evaluar la precisión del modelo al clasificar las imágenes, se realizan predicciones con imágenes nunca antes vistas, ni durante el entrenamiento ni durante la validación.

4.2 Resultados.

A continuación, se presentan los resultados obtenidos en las pruebas anteriormente descritas.

4.2.1 Resultados de navegación.

1) Detección de elementos claves para la navegación.

Para la determinación de los umbrales de detección de los elementos claves para la navegación se tomó como punto de partidas las recomendaciones dadas en la documentación de OpenCV para esta tarea [30].

En el caso de querer realizar la segmentación para el color amarillo, el intervalo utilizado para el valor de matiz está comprendido entre 20 y 30. Para los valores de saturación y valor se utilizaron intervalos entre 100 y 255. Se procedió de forma similar para el color azul y el rojo.

Estos umbrales establecidos resultaron muy efectivos para detectar los elementos deseados en la simulación. Sin embargo, en el entorno real fue necesario modificarlos debido a las condiciones de iluminación y las características de la cámara.

A continuación, se muestra un ejemplo de detección de línea en el entorno simulado. En la Figura 25 se puede observar la imagen tomada por la cámara cenital del drone. Una vez que se realiza la segmentación por color se obtiene una máscara binaria (Figura 26) donde se observa en color blanco el resultado de la segmentación por color.

Figura 25. Línea a seguir en entorno simulado.

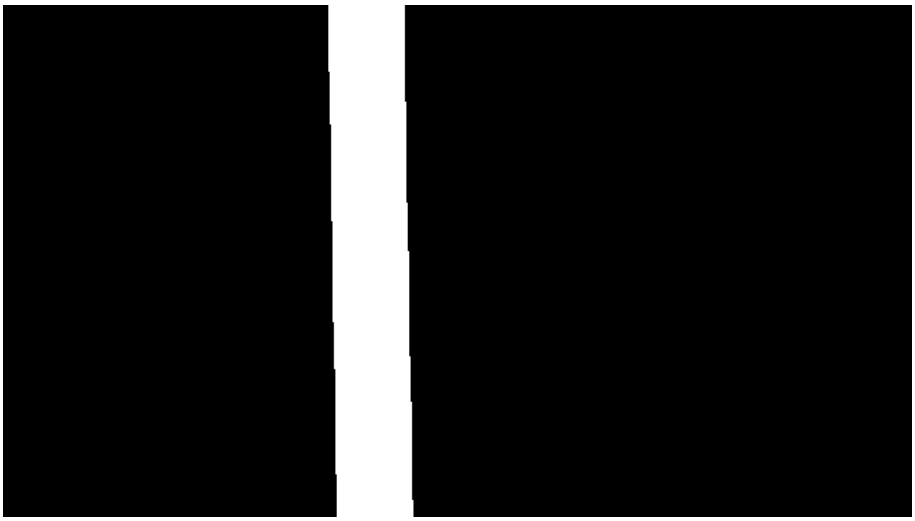

Figura 26. Umbralización de línea en entorno simulado.

Adicionalmente, para comprobar que la detección de color resulte adecuada en el entorno real, se incluyó como textura del suelo de la simulación una foto del suelo del invernadero en el que se realizarán las pruebas (Figura 27). Se puede observar en las Figura 28 y Figura 29 que la detección de la línea no se ve alterada por el entorno.

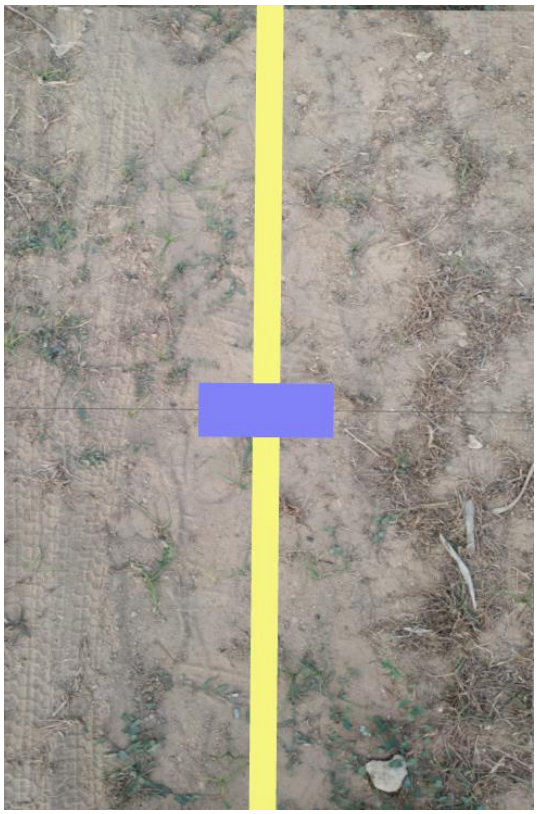

Figura 27. Textura del suelo del entorno real sobre entorno simulado.

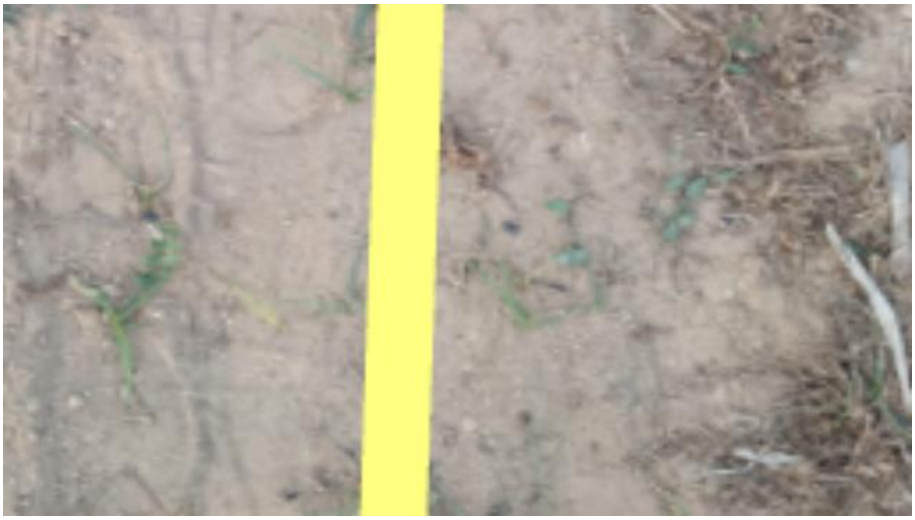

Figura 28. Muestra de línea a seguir sobre entorno simulado con textura de suelo.

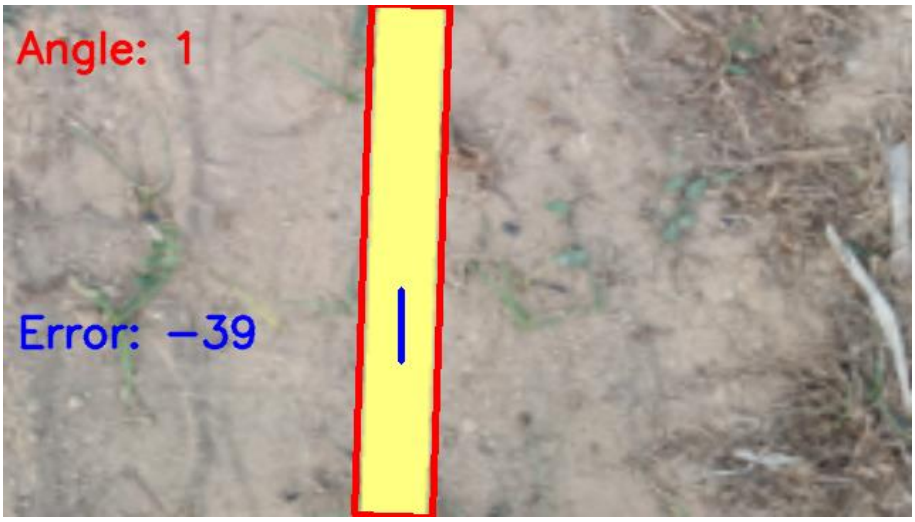

Figura 29. Detección de línea en entorno simulado con textura de suelo.

Aunque el resultado en la simulación fue satisfactorio, las condiciones de iluminación del entorno real requirieron que se modificase el intervalo de detección. Puesto que en la imagen la línea se mostraba con un tono más blanquecino (Figura 30), se redujo el valor saturación y valor de 100 a 20 en el intervalo inferior. De esta forma, la detección de la línea se producía de forma satisfactoria (Figura 31).

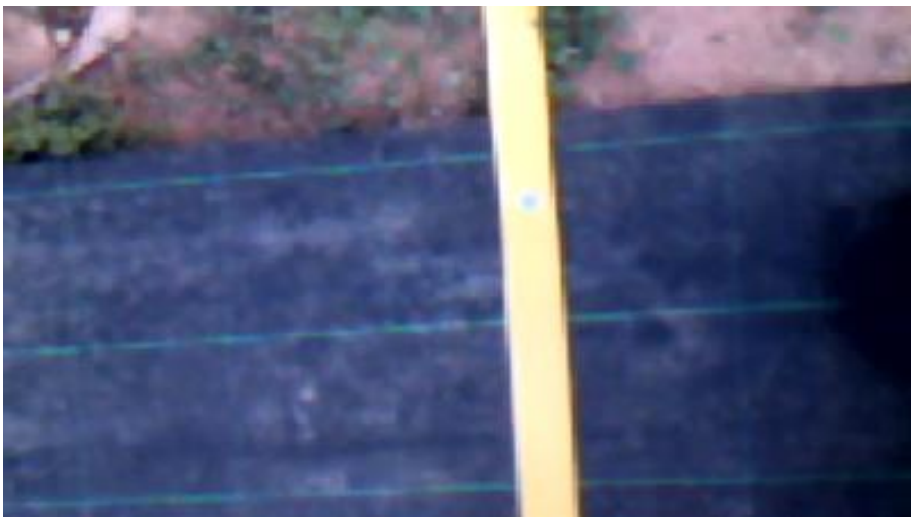

Figura 30. Imagen tomada por el drone de la línea en entorno real.

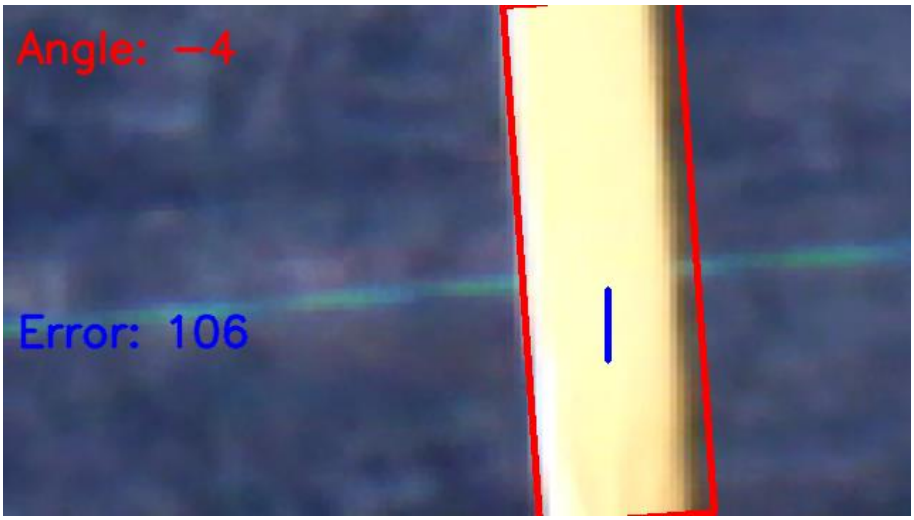

Figura 31. Detección de la línea en entorno real.

Al igual que en el seguimiento de la línea, para detectar las marcas de señalización de las flores se procede de forma similar, la diferencia es que una vez que se detecta una marca (Figura 32), se cambia el control de posición del drone y este pasa a posicionarse sobre la marca girando 90˚ para tomar las fotos (Figura 33). Una vez que el drone se encuentra en la posición requerida, toma la instantánea y continua con la marcha sobre la línea.

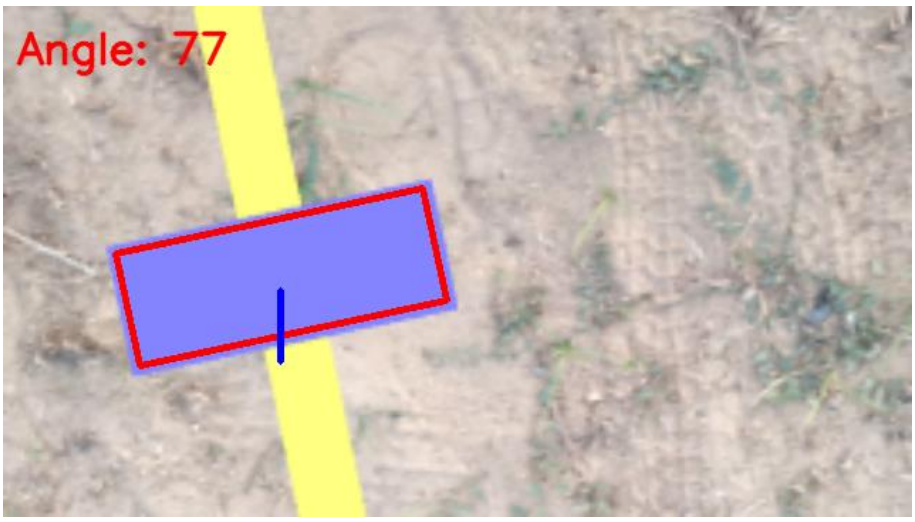

Figura 32. Detección de marca de posicionamiento.

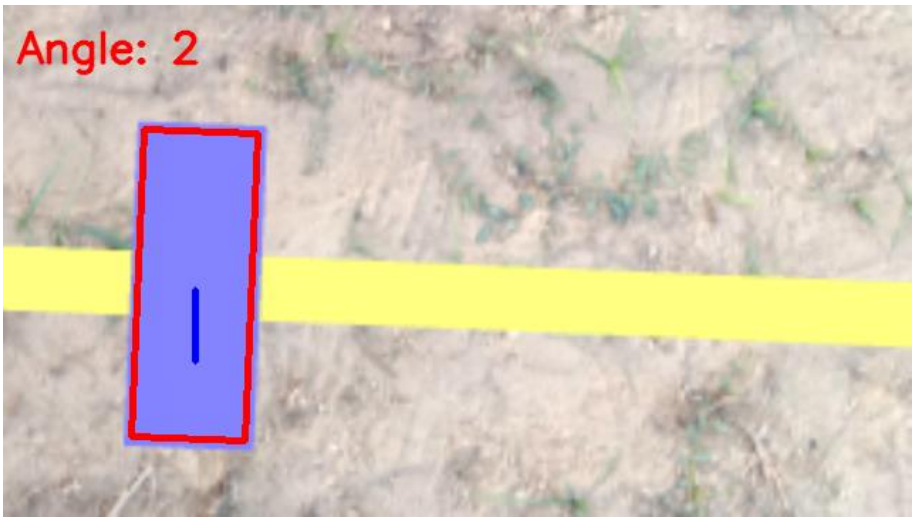

Figura 33. Giro del drone sobre marca de posicionamiento.

Tanto para la detección de marcas azules como de la zona de aterrizaje no se realizaron comprobaciones en el entorno real debido a la imposibilidad de poder controlar el movimiento del drone por causas estructurales.

2) Sintonización de controladores para el movimiento del drone.

Partiendo del trabajo desarrollado previamente por Yakov Korlansky y Shabi Sabatan [17], fue necesario sintonizar el controlador PID encargado de controlar el movimiento horizontal. Sin embargo, para el controlador dedicado al control de giro no fue necesario realizar ningún cambio ya que presentaba un funcionamiento adecuado.

Para realizar las pruebas a los diferentes controladores, se creó un entorno simulado con un recorrido variado para evaluarlos (Figura 34). Para ello se tuvo que modificar el archivo con extensión XML donde se define el mundo que se utiliza y añadir todos los segmentos de línea que conforman el recorrido, así como la zona roja de aterrizaje.

Los resultados se obtuvieron a través de dos enfoques distintos: el primero de ellos de forma gráfica, visualizando el recorrido realizado por el drone en una gráfica a partir de los datos obtenidos de navegación; el segundo de ellos, a través del estudio de la desviación estándar del error de posición.

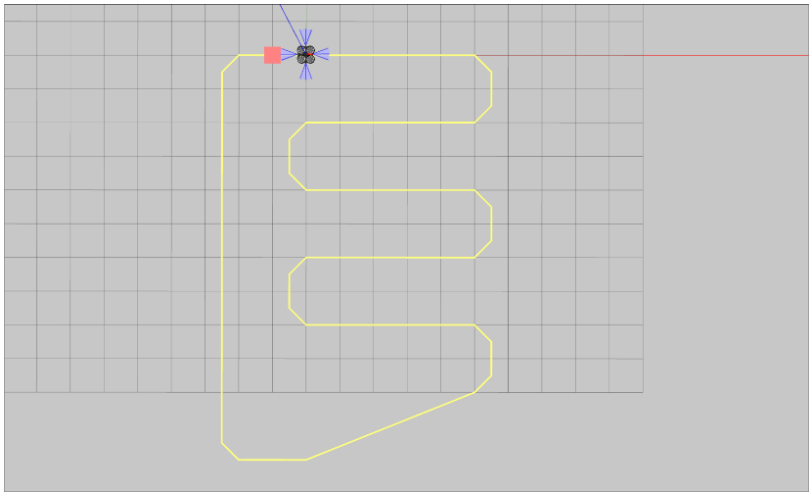

Figura 34. Mundo simulado para pruebas de controladores.

Dentro del entorno simulado que se observa en la Figura 34 cada porción de la cuadrícula representa una distancia de 1 metro.

La sintonización del controlador se realizó de forma manual buscando la oscilación mínima y una respuesta lo suficientemente rápida como para recuperarse cualquier perturbación producida. Para establecer el punto de partida sobre el que se realizará la sintonización se programó un controlador proporcional con ganancia unitaria para comprobar la respuesta del sistema en lazo cerrado sin controlador. La respuesta del sistema de navegación fue la mostrada en la Figura 35: Como se puede observar, el sistema es capaz de seguir la línea, pero en cambio se produce una oscilación de amplitud y frecuencia casi constante. Aun siendo esta oscilación del orden de magnitud de centímetros, es lo suficientemente alta como para provocar que el seguimiento del camino no se realice de forma suave.

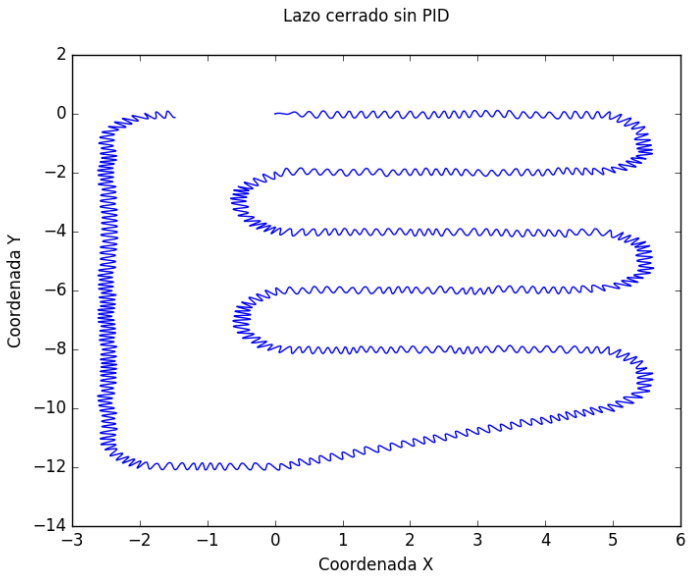

Figura 35. Recorrido de prueba sin controlador

Después de haber obtenido estos datos y a partir de los resultados obtenidos por Shalini et al. [16], se realizó una prueba con un controlador proporcional con una ganancia de 0,112. Esta ganancia está en torno al 10 % de la ganancia con la que se estableció la referencia y debería provocar una respuesta, aunque más lenta, con menos oscilación. En la Figura 36 se observa el gráfico obtenido con este controlador.

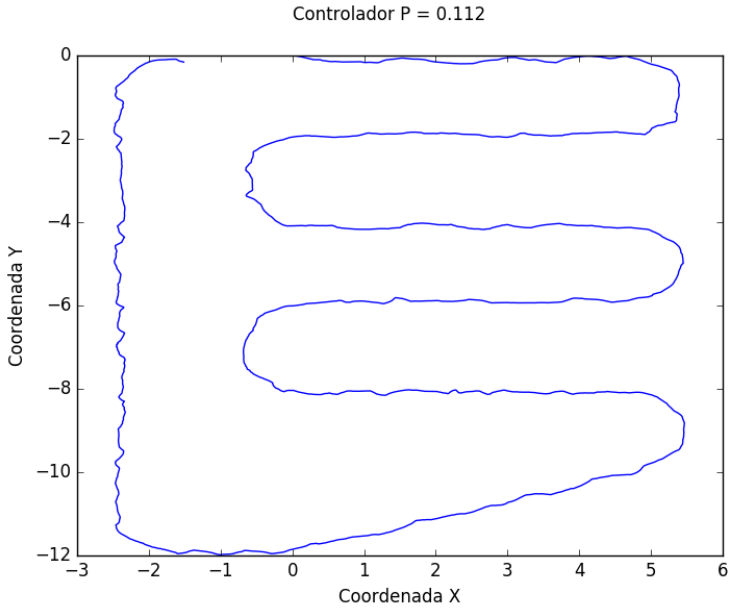

Figura 36. Efecto de controlador proporcional con ganancia 0.112 sobre el recorrido.

Como se puede observar, la respuesta del sistema resulta más suave que la anterior, pero en cambio se observa un error en estacionario, resultando en una ineficiencia de mantenerse sobre el recorrido de forma adecuada. Esto último se puede observar en los segmentos de recta perpendiculares a 0, -2, -4 y -6 en el eje vertical. La desviación estándar del error de posición de este controlador es de 39,05 píxeles.

Tras estos resultados se estableció un controlador con una ganancia 4 veces mayor a la del controlador anterior. Con este controlador proporcional se obtuvo un valor de desviación estándar del error de posición en torno a 16 píxeles. Este controlador tiene una respuesta dos veces mejor que el controlador anterior atendiendo al estadístico mencionado anteriormente. En la Figura 37 se puede observar la respuesta del controlador gráficamente.

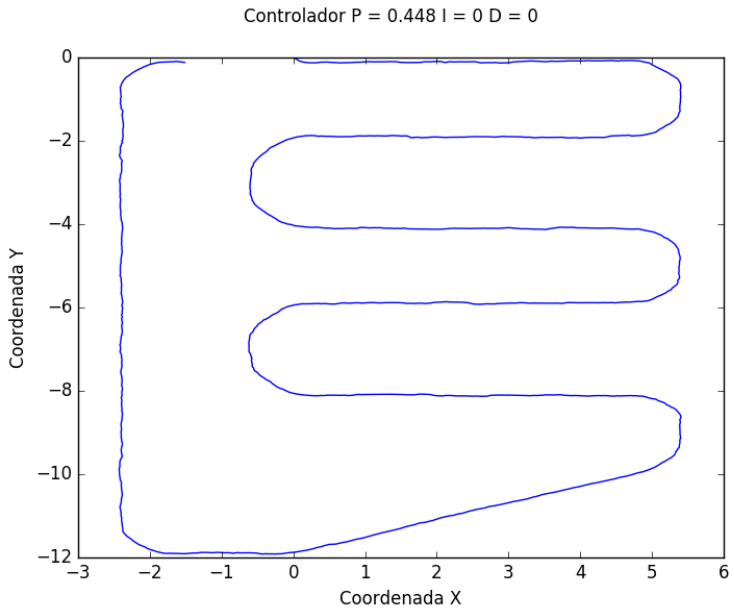

Figura 37. Efecto de controlador proporcional con ganancia 0.448 sobre el recorrido.

Como se puede observar, la respuesta resulta suave y el sistema es capaz de seguir la línea sin problema alguno. En el afán de mejorar la respuesta del controlador, se realizó una tercera prueba con un controlador proporcional de ganancia 0,6. El motivo de esto fue para dotar de una respuesta más rápida al sistema manteniendo la estabilidad. La desviación estándar del error de posición obtenida con este controlador presenta un valor prácticamente idéntico al anterior. Por lo que atendiendo a este estadístico no se produce ningún tipo de mejoría. Sin embargo, al observar la respuesta grafica (Figura 38) se observa una mayor suavidad a la hora de seguir el camino en las secciones de recta. Finalmente se seleccionó como controlador de posición horizontal el tercero de los propuestos con un valor de ganancia proporcional de 0,6. En la Tabla 1 se puede observar una comparación de los resultados obtenidos por los tres controladores.

Controlador	Desviación estándar del error de posición
P = 0,112	39,05 px
P = 0,448	16,60 px
P = 0,6	16,11 px

Tabla 1. Resultados de los tres controladores propuestos.

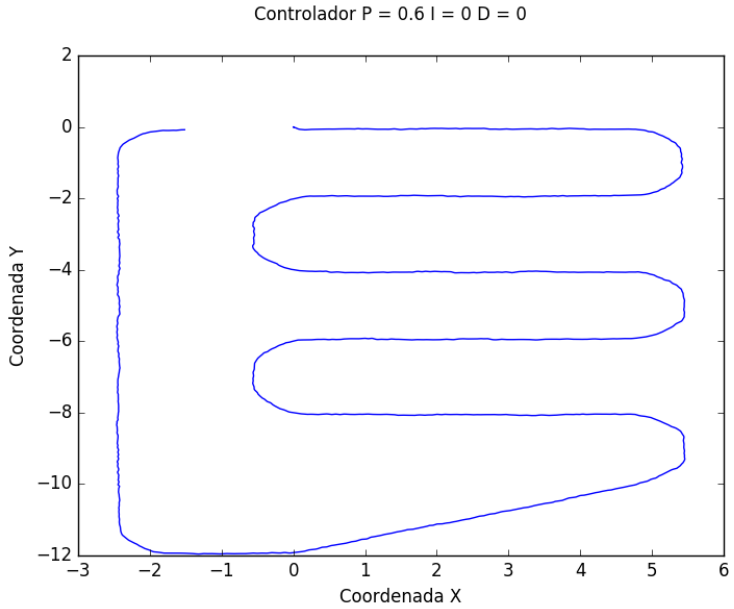

Figura 38. Efecto de controlador proporcional con ganancia 0.6 sobre el recorrido.

Calculando la equivalencia entre píxeles-centímetros en la imagen se puede tener una certeza real de la precisión del controlador propuesto.

A una altura de un metro, a la cual se realizaron las pruebas del controlador, el grosor de la línea detectada era de en torno a 60 píxeles. Teniendo en cuenta que el grosor de esta línea, según está definido en el modelo 3D es de 5 centímetros, se puede establecer una relación de 12 píxeles por centímetro.

En lo referente a la sintonización del controlador para la altura del robot se procedió de forma similar que en el caso anterior. Para establecer el punto de partida se probó la eficacia de un controlador proporcional con ganancia unitaria. A diferencia de lo sucedido en el caso de la posición horizontal, con un controlador proporcional de ganancia unitaria se consigue realizar el control de la altura de forma eficaz.

Esto se puede observar en la Figura 39 donde se muestra el dato de la altura tomado por el drone durante el vuelo cuando el valor de consigna dado fue de un metro. Las unidades de la altitud se encuentran en milímetros.

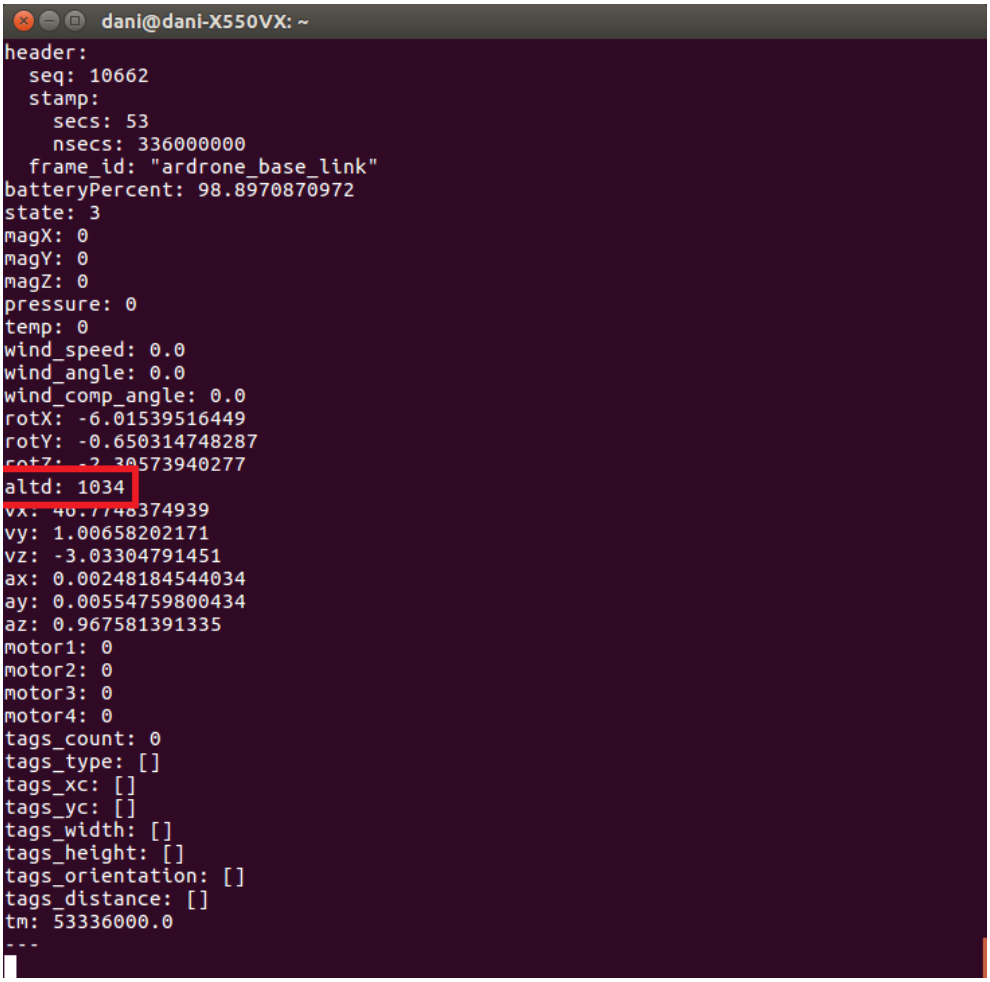

Figura 39. Valor de altitud medido durante el vuelo con valor de consigna de 1 metro.

3) Prueba de control de colisiones en entorno simulado.

Para determinar el umbral de detección a partir del cual el sistema de gestión de colisiones toma el control del drone se colocó un obstáculo que se encuentre a la altura del sensor de forma que cuando el robot recorra la línea se encuentre con el objeto de forma frontal.

Con este método se determinó que para evitar una colisión (Figura 40) el umbral se debía establecer a 40 cm del perímetro exterior del robot. El umbral inferior por el cual se produce el aterrizaje del dispositivo se estableció a una distancia la mitad de la anterior.

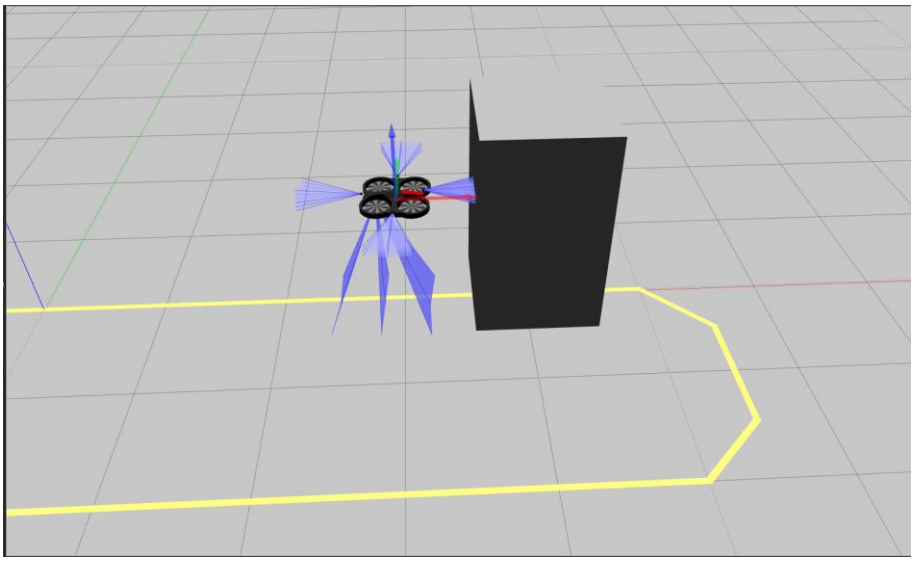

Figura 40. Muestra de efectividad del sistema de evasión de colisiones.

En la Figura 41 se observa cómo, en el momento en el que el obstáculo entra en el campo de detección se envía un comando de velocidad con el fin de alejar al drone del mismo y evitar el choque.

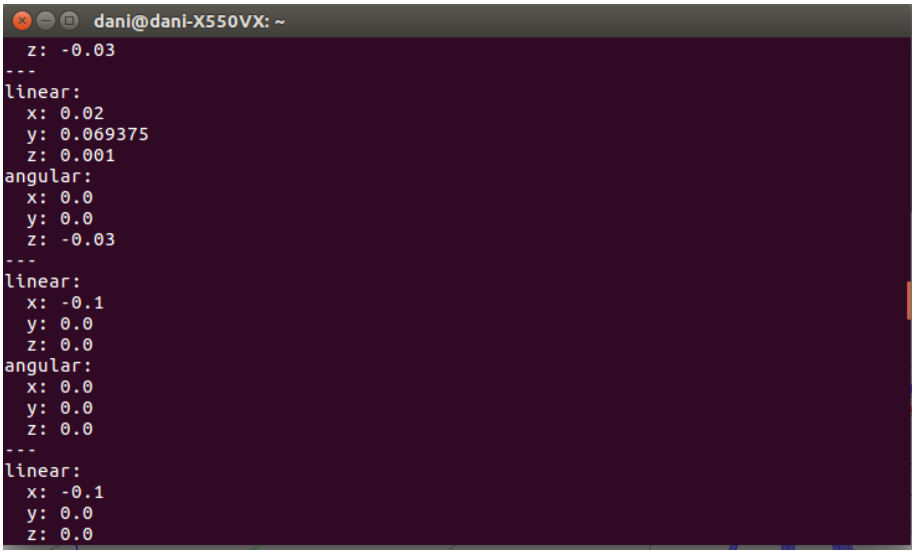

Figura 41. Efecto del sistema de evasión de colisiones sobre los comandos de velocidad del robot.

4) Navegación en entorno simulado y real.

Para la prueba de navegación en el entorno simulado se creó un entorno dentro del simulador GAZEBO donde se encontrasen todos los elementos que forman parte de esta. Este mundo (Figura 42) se creó de la misma forma que se hizo el utilizado en la prueba anterior, pero con la diferencia de que se incluyeron las paradas de color azul.

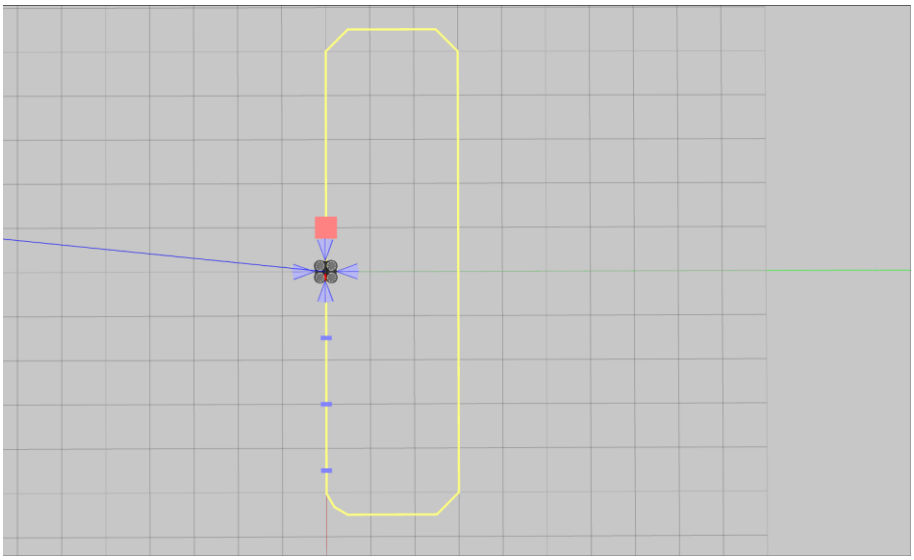

Figura 42. Mundo simulado para evaluar el método de navegación autónoma.

Durante el desarrollo de esta prueba no se observó ninguna anomalía para que el robot siguiese el contorno de la línea. Sin embargo, en los instantes en los que el robot debía posicionarse sobre las marcas azules se observó una oscilación en el movimiento horizontal. Esto fue debido a que se empleó el mismo controlador utilizado para el control de posición horizontal durante el seguimiento de la línea. Este problema se solvento reduciendo a la mitad el valor de la ganancia proporcional. En lugar de una ganancia de 0,6 se estableció una de 0,3. De esta manera el robot fue capaz de mantenerse de forma suave sobre las marcas.

Para realizar la prueba de navegación sobre el entorno real, primero de todo fue necesario acondicionar la sección del invernadero en el que se iba a realizar esta prueba. Las tareas de acondicionamiento realizadas consistieron en retirar de la zona de paso cualquier objeto que pudiese obstruir el movimiento del dispositivo tales como hojas secas o plantas indeseadas. Una vez se realizó esto, se colocó sobre el suelo una línea de color amarillo (Figura 43) de forma similar a la utilizada en el simulador.

Sin embargo, no se pudo completar esta prueba de navegación debido a defectos en el propio drone tales como tener algunas aspas y el chasis torcidos debido a un choque anterior. Esto provocaba que el robot tuviese una deriva hacia el lado izquierdo que no permitía realizar el control adecuado del movimiento.

Figura 43. Recorrido marcado para navegación en entorno real.

4.2.2 Resultados de reconocimiento de flores.

1) Ajuste de parámetros en la segmentación por color.

Como ya se mencionó en el capítulo 3, para obtener la región de interés en las imágenes se aplicaron diferentes algoritmos y técnicas con el fin de extraer únicamente la información necesaria. Primero de todo, fue necesario establecer el intervalo de umbralización para el espacio de color HSV. Para ello se utilizó la imagen de referencia que se muestra en la Figura 44.

Figura 44. Imagen de referencia para ajuste de parámetros en la segmentación por color de la imagen.

La determinación de este intervalo se realizó de forma manual mediante el uso de una herramienta interactiva que permitía modificar en tiempo real los valores de este intervalo y observar los resultados. En la Figura 45 se muestra esto, así como el intervalo de valores escogido.

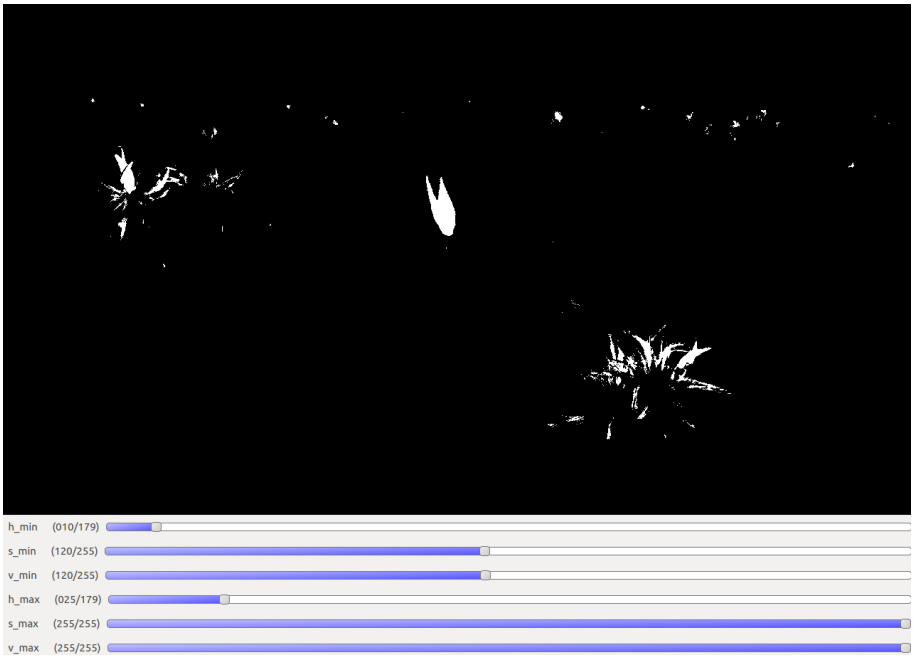

Figura 45. Determinación manual de intervalo de detección para umbralización de imagen.

Como se puede observar en la figura anterior, aparecen gran cantidad de artefactos pertenecientes a las flores del fondo y a secciones de la propia planta. Una de las estrategias que se siguió para mitigar este efecto fue aplicar a la imagen una difuminación, como bien se explicó en el capítulo 3, mediante el uso de un filtro de la media. Este filtro se aplica a la imagen original antes de realizar la umbralización. Para comprobar la eficacia de este método se utilizaron dos tamaños distintos de filtros: Un filtro de la media con tamaño 3x3 y otro de tamaño 7x7. Aplicando el filtro 3x3 a la imagen original (Figura 46) se produce una leve difuminación de la imagen. Al realizar la umbralización de la imagen filtrada (Figura 47) se observa que, aunque bien se han reducido parte de los artefactos del fondo, estos siguen estando bastante presentes en la imagen. Así mismo no se ha conseguido una gran mejora en la homogeneización del color para determinar la región de interés.

Figura 46. Efecto de filtro de la media de tamaño 3x3 sobre imagen de referencia.

Figura 47. Efecto de filtro de la media de tamaño 3x3 sobre umbralización.

Aumentando el tamaño de este filtro a uno de 7x7 (Figura 48) se observa que se produce una mayor difuminación de la imagen eliminando así los bordes de la imagen. A diferencia de en el caso anterior, al realizar la umbralización (Figura 49) se observa que los artefactos del fondo han sido eliminados en su mayoría y se ha conseguido una mayor determinación de la región de interés en las flores al ser el color más homogéneo.

Figura 48. Efecto de filtro de la media de tamaño 7x7 sobre imagen de referencia.

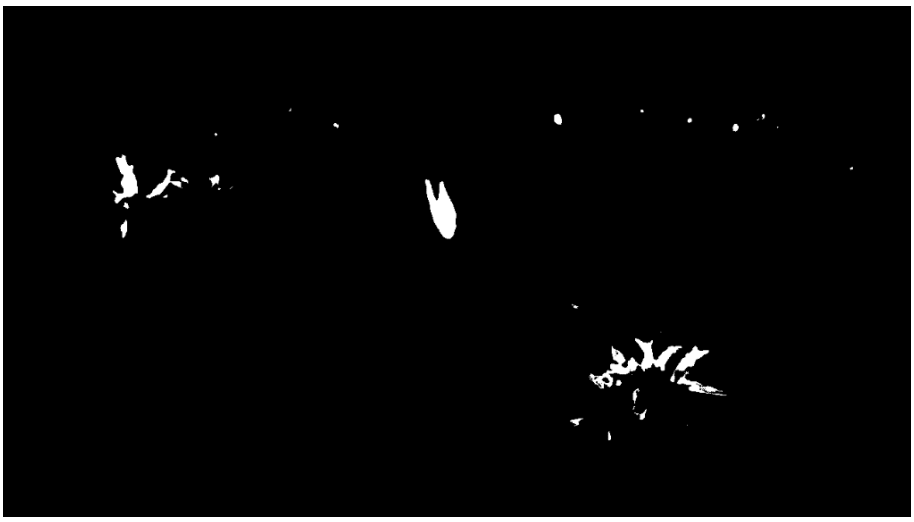

Figura 49. Efecto de filtro de la media de tamaño 7x7 sobre umbralización.

Aunque con esta estrategia se consigue eliminar gran parte de los artefactos del fondo, es posible mejorar esta respuesta aplicando operaciones morfológicas tales como la apertura y el cierre. Tras aplicar la operación de apertura (Figura 50) se puede observar que los artefactos del fondo son cada vez menos presentes y las regiones de interés se encuentran más definidas. Sin embargo, en zonas donde los pétalos de las flores tienen gran

apertura, se detectan como elementos diferentes en lugar de todos ellos pertenecientes a la misma flor. Esto se puede solucionar aplicando una operación de cierre (Figura 51), de esta forma podemos unir contornos cercanos entre ellos y pertenecientes al mismo objeto.

Figura 50. Efecto de operación de apertura sobre imagen umbralizada.

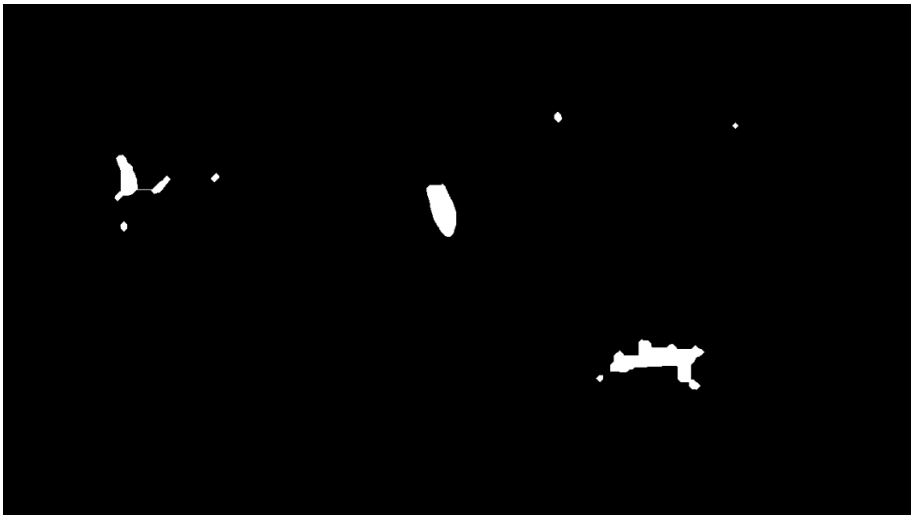

Figura 51. Efecto de operaciones de apertura y cierre sobre imagen umbralizada.

Una vez se han aplicado estas operaciones morfológicas a la imagen es posible obtener las regiones de interés de las flores realizando una umbralización por tamaño de área con la finalidad de eliminar posibles artefactos que hayan escapado al efecto de aplicar las técnicas anteriormente mencionadas.

De esta forma, en la Figura 52 se muestra el resultado de la extracción de las flores sobre la imagen original, habiendo sido recuadradas las flores detectadas por el algoritmo. Posteriormente se realiza el recorte de estas secciones de imagen.

Figura 52. Detección de regiones de interés sobre imagen de referencia.

2) Corrección de overfitting mediante data augmentation.

Como se mencionó en el capítulo 3, el *overfitting* ocurre cuando el número de imágenes de la base de datos no es lo suficientemente alto como para que el modelo sea capaz de extraer las generalidades necesarias y en cambio ajusta sus parámetros internos para memorizar las imágenes de entrada. Esta situación ocurrió durante el proceso de entrenamiento de los modelos que se presentarán en el apartado próximo. Gráficamente, el problema del *overfitting* se puede observar en la Figura 53.

Mientras que con el set de datos de entrenamiento el modelo presenta una precisión muy alta del 100% y una pérdida cercana a 0, la perdida es un indicativo del error cometido por el modelo, ocurre que en el set de datos de validación no se alcanzan los valores del entrenamiento y existe un desfase alto entre estos. Aunque la precisión se mantenga en un valor cercano al 95 %, la pérdida está en torno al 20 %, lo que indica que el modelo presentará un gran error al clasificar imágenes.

Aplicando técnicas de *data augmentation* tales como rotación, traslación, inversión y zoom es posible obtener una cantidad infinita de imágenes a partir de una imagen. En la Figura 54 se muestra una de las imágenes utilizadas para el entrenamiento, y en la Figura 55 5 versiones de esta misma imagen tras haber realizado de forma aleatoria las transformaciones morfológicas mencionadas anteriormente.

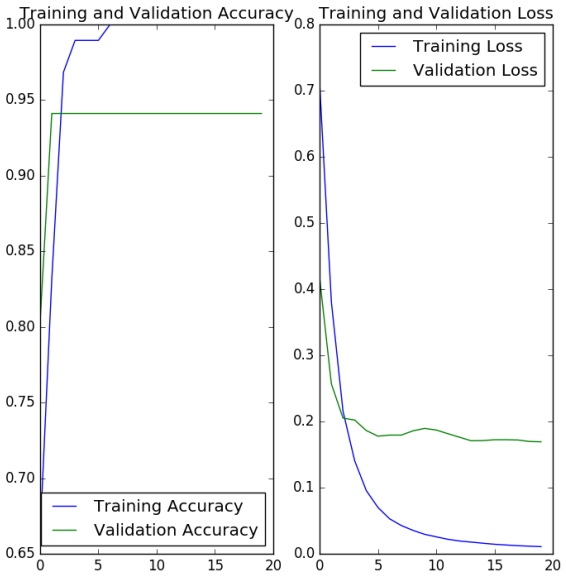

Figura 53. Overfitting en entrenamiento de red neuronal.

Figura 54. Imagen de referencia para data augmentation.

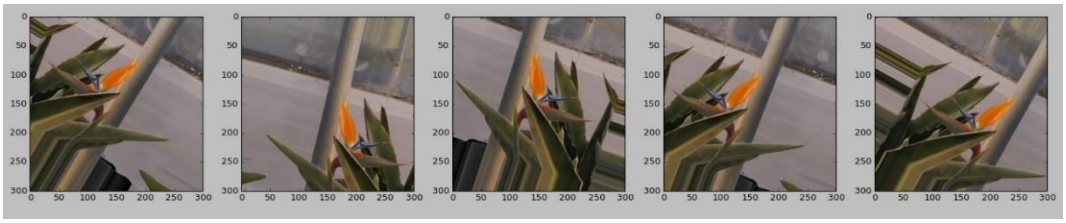

Figura 55. Efecto de aplicar data augmentation sobre imagen de referencia.

La estrategia escogida consiste en utilizar un grupo de imágenes del set de entrenamiento, aplicarle a cada una de ellas estas operaciones morfológicas de forma aleatoria y así generar una nueva tanda de entrenamiento. Este proceso se realiza para cada iteración de entrenamiento. De esta manera, el modelo nunca recibirá como entrenamiento dos imágenes iguales.

Tras aplicar este método se puede comprobar en la Figura 56 que la precisión y pérdida del modelo mejoran de forma significativa sobre el set de datos de validación.

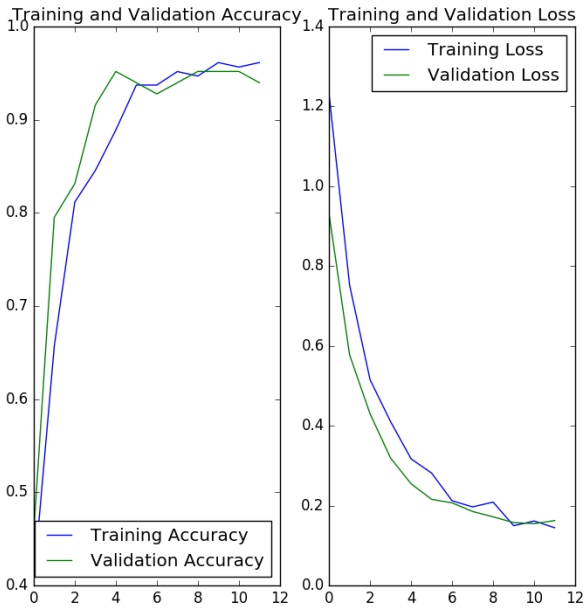

Figura 56. Correción de overfitting mediante el uso de data augmentation.

3) Comparación de diferentes modelos para clasificación de imágenes.

En esta sección se comparan la eficacia de los modelos *Inception_V3* y *mobileNet_V2* para clasificar imágenes de flores. El modelo *Inception_V3* cuenta con 21,802,784 parámetro, mientras que el modelo *mobileNet_V2* cuenta tan sólo con 2,257,984. Pese a esta diferencia, se observó que el modelo *mobileNet_V2* tenía mejores resultados en la clasificación de imágenes de flores. En ambos modelos se utilizaron imágenes con una resolución de 224x224 píxeles tanto para el entrenamiento como para la vadilación. Para poder comparar ambos modelos se realizaron gráficas donde se muestran los resultados de precisión y pérdida para los sets de entrenamiento y validación.

Primero de todo, se realizaron las pruebas con una base de datos en la que se encontraban imágenes de flores y de hojas con el objetivo de tener una clasificación de tipo binario con las etiquetas “flor” y “no flor”. Esta base de datos cuenta con 95 imágenes en el set de entrenamiento y 51 en el de validación. Puesto que la cantidad de imágenes es relativamente pequeña, se aplicaron técnicas de *data augmentation* durante la realización de esta prueba. Posteriormente se utilizó para la segunda prueba una base de datos que contaba con imágenes de tres clases distintas, estas son “flor”, “no flor” y “flor madura”. De esta manera se es posible detectar flores que se encuentren en un grado alto de maduración y no sean aptas para la recolección y posterior venta. Esta base de datos cuenta con 207 imágenes de entrenamiento y 83 de validación. Así mismo se aplicaron técnicas de *data augmentation* al igual que en el caso de la base de datos anterior.

En el caso de la primera base de datos, ambos modelos (Figura 57) obtuvieron una respuesta similar, sin embargo, se observa que en ambos se produce cierto *overfitting* debido a la poca cantidad de ejemplos de entrenamiento.

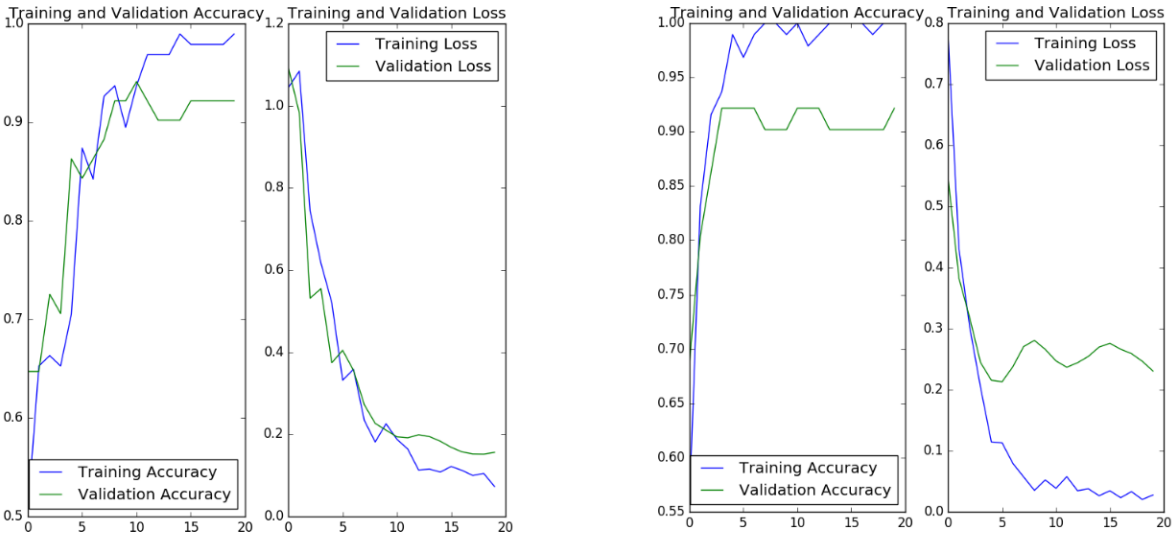

Figura 57. Resultados del modelo Inception_V3 (izquierda) y el modelo mobileNet_V2 (derecha) sobre el dataset de dos etiquetas.

Los valores de precisión y pérdida para el set de validación de los modelos se pueden observar en la Tabla 2.

Modelo	Precisión	Pérdida
Inception_V3	0.9216	0.1560
mobileNet_V2	0.9216	0.2305

Tabla 2. Resultados de precisión y pérdida de los modelos Inception_v3 y mobileNet_V2 aplicados a dataset con dos etiquetas.

Tras 15 iteraciones de entrenamiento, el modelo *Inception_V3* no mejora su respuesta, mientras que en el caso del modelo *mobileNet_V2* basta con 6 iteraciones de entrenamiento para obtener los mismos resultados que el modelo anterior. En el caso de la segunda base de datos (Figura 58), el modelo *mobileNet_V2* presenta una ligera ventaja en la precisión frente al modelo *Inception_V3*, así como un valor de pérdida de la mitad del que presenta el segundo modelo.

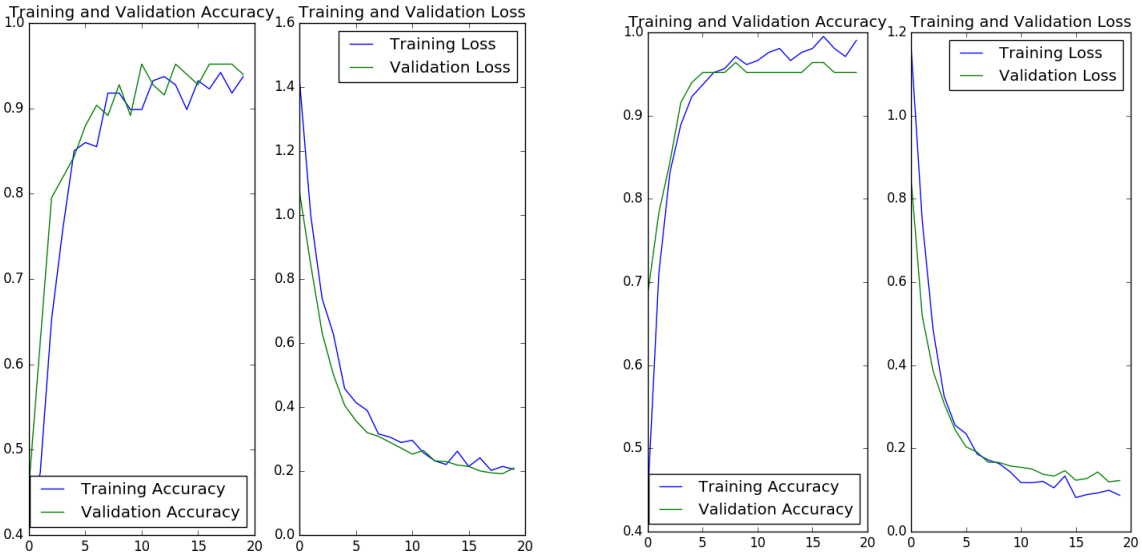

Figura 58. Resultados del modelo Inception_V3 (izquierda) y el modelo mobileNet_V2 (derecha) sobre el dataset de tres etiquetas.

Los valores de precisión y pérdida para el set de validación de estos modelos se pueden observar en la Tabla 3.

Modelo	Precisión	Pérdida
Inception_V3	0.9398	0.2092
mobileNet_V2	0.9518	0.1229

Tabla 3. Resultados de precisión y pérdida de los modelos Inception_V3 y mobileNet_V2 aplicados a dataset con tres etiquetas.

De la misma manera que en el caso anterior, con tan sólo 6 iteraciones de entrenamiento, el modelo *mobileNet_V2* presenta mejores resultados que el otro modelo. Por este motivo fue el empleado para realizar la clasificación de imágenes.

4) Aplicación del modelo desarrollado a la clasificación sobre imágenes nunca vistas.

Una vez establecido el modelo a utilizar, se comprobó la eficacia de este para clasificar imágenes de flores nunca vistas antes, ni en el proceso de entrenamiento ni en el de validación. En este apartado se obtuvieron resultados dispares. Mientras que el modelo fue capaz de clasificar imágenes de las tres etiquetas con una precisión cercana al 100%, hubo otras imágenes que no fue capaz de clasificarlas de forma correcta. Una muestra de esto se puede observar en la Figura 59.

Figura 59. Muestra de clasificación de imágenes de forma correcta y de forma fallida.

4.3 Discusión.

En este trabajo se presenta un controlador para el movimiento de un drone basado en visión artificial a partir del cual se han obtenido grandes resultados en un entorno simulado de la misma manera que el propuesto por Shalini et al. en [16]; si bien en otras investigaciones se han propuestos diferentes tipos de controladores para el movimiento de un drone, como es el caso del presentado por Gatica et al. en [31] o el propuesto por Prayitno et al. en [32].

Así mismo, se ha comprobado que el método de sintonía manual resulta efectivo cuando no es posible realizar una modelización del sistema completo. A modo de justificación de los resultados obtenidos para la sintonía, según el método de Ziegler-Nichols, se puede obtener la ganancia para un controlador de tipo proporcional como la mitad de la ganancia crítica. Esta ganancia crítica es la que provoca en el sistema una oscilación de amplitud y frecuencia constante. Puesto que como se explicó anteriormente, con una ganancia unitaria se producía una oscilación en el sistema casi constante, la ganancia crítica debe estar cercana a la unidad, la elección de una ganancia cercana a la mitad de la unitaria permitirá realizar el control de forma estable.

También se ha comprobado la eficacia de la utilización de técnicas de *data augmentation* para reducir el efecto del *overfitting* como exponían Taylor et al. en [23]. De la misma manera se han confirmado las ventajas, tanto a nivel de precisión como de ahorro de recursos informáticos, de la utilización de modelos pre-entrenados en la clasificación de imágenes como bien expusieron Hosny et al. en [26] y Wang et al. en [27].

Las ventajas que presentan los sistemas propuestos son:

- Se encuentran desarrollados sobre plataformas de código abierto con grandes comunidades detrás, lo que asegura que estos proyectos no se abandonarán a corto plazo, teniendo así soporte para futuros cambios y mejoras.
- Gracias a estar desarrollado dentro del entorno ROS permite la escalabilidad y aplicación de los diferentes subsistemas propuestos a otras aplicaciones sin tener que realizar grandes cambios.
- El sistema está desarrollado sobre elementos de bajo coste, lo que aumenta su aplicabilidad ya que no es necesario realizar una gran inversión.

Po otro lado, las desventajas que presenta el sistema son las siguientes:

- Una gran dependencia de las condiciones ambientales en los sistemas de visión artificial.
- Escasa robustez mecánica y precisión de los elementos utilizados al ser estos de bajo coste.

5 Conclusiones y trabajo futuro.

En este último capítulo se recogen las conclusiones derivadas de la realización del trabajo y de los resultados obtenidos del mismo. De la misma manera, se marcan al final del apartado las futuras líneas de trabajo con el objetivo de mejorar la aplicación planteada.

5.1 Conclusiones.

Tras el desarrollo del trabajo es posible concluir que, debido a la emergencia climática y ambiental en la que se encuentra el planeta en la actualidad, así como el agravamiento de las desigualdades, resulta vital aumentar la inversión destinada a la implementación de la agricultura de precisión, con el fin de cumplir los objetivos propuestos de Desarrollo sostenible de la Agenda 2030 tales como la mejora de la producción utilizando los recursos de forma eficiente y sostenible con el medio ambiente.

En esta línea, la utilización de robots en los procesos productivos favorece esta productividad eficiente dentro de la agricultura. Sin embargo, es de vital importancia fomentar la profesionalización de sectores de la población con poca formación y mejorar las ofertas laborales para compensar la posible pérdida de empleos producida por la inminente nueva revolución industrial. Solamente en España, se prevé que una quinta parte de los empleos desaparezcan con el aumento de la automatización según la Organización para la Cooperación y el Desarrollo Económico (OCDE). De la misma manera se verán afectados casi un tercio de los empleos que serán requeridos de una reestructuración para adaptarse a este cambio. Es por eso que no se debe perder el foco en los problemas laborales y sociales que plantea la automatización.

Otra de las conclusiones a la que se ha llegado tras la realización de este trabajo es del gran avance que supondrá la implementación de tecnologías de Deep Learning más allá del entorno de investigación. Aunque aún sea una tecnología prematura, pasará a formar parte en unos años de nuestra vida cotidiana como en el caso del sistema de navegación autónoma de Tesla o la traducción de textos de Google.

5.1.1 Trabajo futuro.

Debido a las limitaciones presentes en este proyecto, tanto relativas al desarrollo de éste, como a las derivadas de la situación actual del covid-19, u otros factores tales como los defectos estructurales que presenta el dron, que no permitieron la prueba del sistema en el entorno real, resulta necesario continuar con el desarrollo del presente trabajo para así mejorar el sistema propuesto. Para ello, se plantean las siguientes líneas de trabajo futuro:

- Mejora del sistema de seguridad frente a colisiones implementando un algoritmo capaz de evitar los objetos y no sólo mantenerse a una distancia prudencial.
- Aumento de la base de datos para el entrenamiento de la red neuronal debido a que uno de los puntos clave de un modelo de Deep Learning es la calidad y cantidad de los datos con los que se ha entrenado.
- Implementación de técnicas de detección de objetos mediante Deep Learning para eliminar la variabilidad de resultados provocada por el efecto de las condiciones ambientales en el algoritmo de extracción de flores desarrollado mediante visión artificial.
- Solucionar problemas relacionados con la navegación en el entorno real tales como defectos mecánicos en el drone.
- Mejora del sistema de navegación prescindiendo de la necesidad de utilizar una línea indicando el camino y realizando la navegación detectando el contorno del propio pasillo del invernadero.

6 Bibliografía.

[1] E. C. M. S. B. A. R. Rodolfo Bongiovanni, Agricultura de precisión: Integrando conocimientos para una agricultura moderna y sostenible, Montevideo: PROCISUR/IICA, 2006, pp. 15-17.

[2] F. Leiva, «La agricultura de precisión: una producción más sostenible y competitiva con visión futurista,» de *VIII Congreso de la Sociedad Colombiana de Fitomejoramiento y Producción de Cultivos*, Bogotá, 2003.

[3] iGrow.news, «iGrow.news,» [En línea]. Available: https://www.igrow.news/igrownews/tech-magnate-jack-ma-visits-dezhou-greenhouse. [Último acceso: 12 Diciembre 2020].

[4] HortoInfo, «HortoInfo,» [En línea]. Available: http://www.hortoinfo.es/index.php/3343-dron-iknvernadero-060715. [Último acceso: 12 Diciembre 2020].

[5] «Agriculture Robots - Global Market Trajectory & Analytics,» Global Industry Analysts, Inc, 2020.

[6] D. D. K.R. Thorp, «Color image segmentation approach to monitor flowering in lesquerella,» *Industrial Crops and Products,* vol. 34, nº 1, pp. 1150-1159, 2011.

[7] S. A. a. R. N. S. a. S. S. N. a. L. Ganesan, «Fruit Recognition using Color and Texture Features,» *Journal of Emerging Trends in Computing and Information Sciences,* vol. 1, nº 2, pp. 197-200, 2010.

[8] B. V. a. S. S. P. Biradar, «Flower Detection and Counting Using,» *International Journal of Computer Science and Information Technologies,* vol. 6, nº 3, pp. 2498-2501, 2015.

[9] T. B. P. R. P. Tiay, «Flower recognition system based on image processing,» de *Third ICT International Student Project Conference (ICT-ISPC2014)*, 2014.

[10] C. C. Aggarwal, Neural Networks and Deep, Springer, 2018.

[11] T. M. Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math, 1997.

[12] M. O. Horea Mureşan, «Fruit recognition from images using deep learning,» *Acta Universitatis Sapientiae, Informatica,* vol. 10, nº 1, pp. 26-42, 2018.

[13] A. T. H. M. Philipe A. Dias, «Apple flower detection using deep convolutional networks,» *Elsevier,* vol. 99, pp. 17-28, 2018.

[14] G. K. O. H. A. e. a. Farjon, «Detection and counting of flowers on apple trees for better chemical thinning decisions,» *Precision Agriculture,* vol. 21, pp. 503-521, 2020.

[15] A. Brandao, F. Martins y H. Soneguetti, A Vision-based Line Following Strategy for an Autonomous UAV, 2015.

[16] P. Shalini, M. F. Poulopoulou y E. Vela, «Line Detector ROS Node,» *Pendiente de publicación, disponible en línea,* 2019.

[17] Y. S. S. Korlansky, «Line Follower Drone | ROS + OpenCV | Parrot Bebop 2,» 10 Noviembre 2019. [En línea]. Available: https://www.youtube.com/watch?v=cXND8kajHrA&ab_channel=YakovKorlansky. [Último acceso: 9 Octubre 2020].

[18] D. G. F. M. A. J. Martin, «EL COLOR: MODELOS Y TRANSFORMACIONES DE LOS ESPACIOS DE COLOR,» de *Conceptos y métodos en visión por computador.*, pp. 47-76.

[19] J. F.-C. M. V. N. L. Espinosa-Aranda, «OPERACIONES SOBRE EL HISTOGRAMA Y FILTRADO DE LA IMAGEN,» de *Conceptos y métodos en visión por computador*, 2016, pp. 38-42.

[20] J. C. G. R. Á. I. M. E. J. A. Cancelas, «PROCESAMIENTO MORFOLÓGICO,» de *Conceptos y métodos en visión por computador*, 2016, pp. 79-85.

[21] C. C. Aggarwal, «PRACTICAL ISSUES IN NEURAL NETWORK TRAINING,» de *Neural Netoworks and Deep Learning*, Nueva York, Springer, 2018, pp. 25-26.

[22] C. C. Aggarwal, «CONVOLUTIONAL NEURAL NETWORKS,» de *Neural Networks and Deep Learning*, Nueva York, Springer, 2018, pp. 337-338.

[23] L. TAYLOR y G. NITSCHKE, «Improving Deep Learning using Generic Data Augmentation,» *arXiv:1708.06020,* 2017.

[24] C. C. Aggarwal, «AN INTRODUCTION TO NEURAL NETWORKS,» de *Neural Networks and Deep Learning*, Nueva York, Springer, 2016, pp. 63-64.

[25] C. C. Aggarwal, «CASE STUDIES OF CONVOLUTIONAL ARCHITECTURES,» de *Neural networks and Deep Learning*, Nueva York, Springer, 2018, pp. 338-341.

[26] K. M. HOSNY, M. A. KASSEM y M. M. FOAUD, «Skin cancer classification using deep learning and transfer learning,» de *9th Cairo International Biomedical Engineering Conference (CIBEC*, El Cairo, 2018.

[27] C. &. C. D. &. L. H. &. L. B. &. Z. C. &. C. D. &. Z. E. Wang, «Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model,» *IEEE Access,* 2019.

[28] TensorFlow, «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/. [Último acceso: 4 Diciembre 2020].

[29] Jordi TORRES.ai, «Jordi TORRES.ai,» 8 Septiembre 2019. [En línea]. Available: https://torres.ai/data-augmentation-y-transfer-learning-en-keras-tensorflow/. [Último acceso: 12 Octubre 2020].

[30] OpenCV, «docs.opencv.org,» [En línea]. Available: https://docs.opencv.org/master/df/d9d/tutorial_py_colorspaces.html#:~:text=For%20HSV%2C%20hue%20range%20is,need%20to%20normalize%20these%20ranges.. [Último acceso: 12 Octubre 2020].

[31] N. &. M. C. &. S. A. Gatica, «Real fuzzy PID control of the UAV AR.Drone 2.0 for hovering under disturbances in known environments,» de *CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, Pucon, 2017.

[32] A. Prayitno, V. Indrawati y I. I. Trusulaw, «Fuzzy Gain Scheduling PID Control for Position of the AR.Drone.,» *International Journal of Electrical & Computer Engineering,* vol. 8, nº 4, pp. 1939-1946, 2018.

[33] C. &. C. D. &. L. H. &. L. B. &. Z. C. &. C. D. &. Z. E. Wang, «Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model,» *IEEE Access (Pendiente de publicación).*