



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN

GRADO EN INGENIERÍA TELEMÁTICA

**TRABAJO FIN DE GRADO**

# DESPEGUE, BÚSQUEDA Y ATERRIZAJE

Autor: Jorge Vela Peña

Tutor: José María Cañas Plaza

Curso académico 2017/2018

# Resumen

Es cada vez más común el uso de drones en el día a día de las personas. Se puede observar cómo han explotado en estos últimos años como un aparato de entretenimiento. Pero estos tienen mucha historia, los UAV se comenzaron a utilizar hace muchos años con fines bélicos, y poco a poco su desarrollo ha permitido que se utilicen en ámbitos muy diferentes, como la robótica, gracias, por ejemplo, al comportamiento autónomo de este, trabajando también en una navegación en interiores.

Durante este proyecto se ha diseñado y programado un algoritmo con la finalidad de que el dron navegue de forma autónoma y no necesite a ninguna persona diciéndole lo que tiene que hacer en cada momento o controlándole, sino que se pueda confiar en el una vez abandone el punto de origen. Para conseguir esto hay que tener en cuenta los distintos estados en los que se puede encontrar el drone (como puede ser despegando, en periodo de búsqueda o aterrizaje), situándose en cada uno de estos dependiendo de lo que este viendo en cada instante. Para el desarrollo de la visión se ha trabajado en la detección visual objetos, haciendo esto a partir de filtros de color, que nos permite detectar si un objeto es o no el deseado y de esta forma poder dirigirnos a él. Cabe destacar que este tiene que ser un objeto bien elegido y que sea difícil de confundir con los demás, para poder asegurarnos de un correcto funcionamiento. Para toda esta programación nos hemos apoyado en el entorno JdeRobot, utilizando OpenCV para el procesamiento de imágenes.

El algoritmo programado se ha validado experimentalmente, tanto en un dron simulado (en simulador Gazebo), como en un dron real (utilizando el Ardrone2 de parrot). Se ha conseguido un comportamiento satisfactorio funcionando en tiempo real. Todo el software desarrollado y los videos de las pruebas está accesible públicamente.

# Contents

<b>Índice de figuras</b>	<b>III</b>
<b>1 Introducción</b>	<b>1</b>
1.1 Historia de los drones . . . . .	1
1.2 Hardware. . . . .	3
1.3 Software. . . . .	9
1.4 Uso actual de los drones . . . . .	14
1.4.1 Medios Audiovisuales. . . . .	14
1.4.2 Control de distintas zonas. . . . .	14
1.4.3 Desarrollo autónomo. . . . .	16
<b>2 Objetivos</b>	<b>19</b>
2.1 Objetivo principal. . . . .	19
2.2 Metodología. . . . .	20
2.3 Plan de trabajo. . . . .	21
<b>3 Infraestructura utilizada.</b>	<b>23</b>
<b>4 Algoritmo</b>	<b>29</b>
4.1 Diseño. . . . .	29
4.2 Percepción. . . . .	30
4.3 Control. . . . .	41
4.4 Funcionamiento del algoritmo. . . . .	44
<b>5 Experimentos.</b>	<b>46</b>



# List of Figures

1.1	Esta imagen muestra los movimientos que tiene un dron. . . . .	4
1.2	Distintas partes del dron. . . . .	8
3.1	Esta imagen muestra el conjunto de herramientas utilizadas. . . . .	28
4.1	Esta imagen muestra un primer filtro de color. . . . .	32
4.2	Esta imagen muestra el conjunto de herramientas utilizadas. . . . .	33
4.3	Imagen que muestra la suma de las diferentes imagenes creadas por cada objeto. . . . .	39
4.4	Imagen que muestra el diagrama de estados. . . . .	43
5.1	Aterrizaje . . . . .	49

# Chapter 1

## Introducción

Cada vez es mas común el uso de drones para labores que pueden ser muy diversas, como puede ser grabar un plano para una película o mantener vigilado un lugar sobrevolando estas zonas. Pero un area muy importante es la navegación autonoma de estos, conseguir que realicen ciertas tareas sin que haya nadie controlando su ruta ni como se realiza esta. Para ello, hay que contar con los distintos sensores que se le pueden a nadir a un drone y la forma de utilizar estos en beneficio propio para conseguir dicha navegación.

En los siguientes parrafos se va a realizar una breve introducción sobre estos aparatos, la historia que tienen, su hardware y software principal y los distintos usos que hay para ellos actualmente.

### 1.1 Historia de los drones

Los drones son conocidos como UAV(vehículos aereos no tripulados).El primer registro de UAV se trata de un globo aeroestático en un entorno militar, ya que este se podía utilizar para sobrevolar una zona y lanzar bombas desde cierta altura sin necesidad de que hubiera ninguna persona en este y, por tanto, sin arriesgar una vida.

## CAPÍTULO 1. INTRODUCCIÓN

---

Este UAV es muy distinto a lo que vino despues, principalmente porque el motor de éste se trata de una bolsa que tiene un gas mas ligero que el aire, lo que le permite coger altura y jugar con las corrientes de viento para desplazarse en una dirección o en otra.

Mas adelante, ya en la primera guerra mundial, se comenzaron a utilizar para sobrevolar las areas enemigas y hacer fotos de estas para así tener un control de sus movimientos (en este punto nos damos cuenta de que el introducir una cámara en un UAV es algo que se hizo desde los primeros momentos, pero en ello ya profundizaremos mas adelante). Estos vehículos eran aviones tripulados por radiofrecuencia, por lo que se dio un gran salto con respecto al anterior, pues era mucho mas facil su control, por lo que podían manejar su trayectoria con mucha mas facilidad.

También durante la primera, pero mas desarrollado para la segunda guerra mundial, se le dió uso a éstos para utilizarlos como explosivos, ya que podían seguir su trayectoria en todo momento y asegurarse que llegaban al destino correcto. Además de poder seguir a otros vehículos en movimiento del bando enemigo y así hacer que este no llegara a su destino.

Está claro que los inicios de estos tenían solo fines militares y que su desarrollo era exclusivamente para ello. En comparación con estos datos, el avance sobre estos freno en gran medida y ya lo que se hacía era modificaciones para poder dar uso a lo que ya había, pues se utilizaban para vigilancia aérea en zonas de conflictos, lo que llevó a mejorar el sistema de control haciendo así que se pudieran manejar a una mayor distancia.

Destacar que fue alrededor de 1980 cuando se vió que la tecnologá y el software de los UAV eran de gran fiabilidad y se podían asignar a estas tareas de mayor responsabilidad para no jugarse la vida de los pilotos. Decir como dato curioso, que una vez no estaban los pilotos en la cabina del vehículo se podía jugar con mayor libertad a la hora de realizar movimientos, ya que ciertos giros que los pilotos no podian realizar por ser demasiado bruscos para aguantarlos el cuerpo humano, ahora podían hacerlos

## CAPÍTULO 1. INTRODUCCIÓN

---

con la brusquedad que permitiera el sistema.

Tras esto ya en la decada de los 90 se da un avance muy importante, y es que se desarrolla el sistema GPS para el desplazamiento de estos vehículos. Esto permitía no depender de la radiofrecuencia, ya que con ésta vamos a tener un límite en distancia y no revisar los datos para ver en todo momento su situación y dirigir la trayectoria. Con éste sistema se traza una ruta al inicio y el UAV puede trabajar de forma autonoma.

Con todo esto, y teniendo en cuenta que fue en torno al año 2000 cuando se comenzo a dar un uso a los drones tal y como los vemos ahora, no un enfoque tan militar y si mas para uso civil, hay que destacar el gran avance que ha sufrido la robótica aerea, pues en la parte aeroespacial, cada vehiculo innovador que sale mejora con creces el anterior, por cualidades como materiales, diseño o estructura, permitiendo esto una mayor velocidad o resistencia. Y por parte de la robótica ocurre lo mismo, se esta en un continuo desarrollo, y viendo el futuro que tienen los drones muchas empresas y grupos de investigación han decidido centrarse en ellos, pudiendo así mejorar a diario el software de estos, lo que permite un control mas fluido, gracias al envío y procesamiento de información, así como controlar mejor en todo momento el estado que se encuentra el drone(batería, posicionamiento en los distintos ejes o velocidad).

### 1.2 Hardware.

Hay que destacar las partes que tiene un drone y su forma, pues es gran parte lo que lo hace tan especial,permite que tenga una gran libertad de movimientos, ya que puede moverse sin problema desde cualquier punto hacia los ejes X, Y y Z. Lo que ganamos con esto son cosas como poder permitirse un aterrizaje y un despegue totalmente vertical, sin depender de un espacio en el que cojer velocidad para poder levantar el vuelo, e igual con el aterrizaje, pudiendo el dron estando quieto en el aire bajar totalmente en vertical hasta tocar posarse sobre el suelo. Una vez en el aire



## CAPÍTULO 1. INTRODUCCIÓN

---

pueden moverse adelante, atrás, izquierda, derecha, arriba, abajo y combinaciones de movimientos entre ejes, además de los movimientos Roll, Yaw y Pitch y sin necesidad de hacer movimientos bruscos. Sin embargo, en los anteriores UAV solo tenemos el movimiento hacia adelante, teniendo que jugar con Roll, Yaw y Pitch para poder movernos en los distintos ejes.

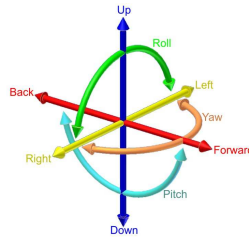


Figure 1.1: Esta imagen muestra los movimientos que tiene un dron.

Explicado esto, un desglose explicando cada una de las partes sería :

**Frame:** También conocido como marco, estructura o chasis. Es la estructura principal sobre la que se sitúan el resto de los elementos. Este variará su forma dependiendo del dron, variando la longitud de las patas o el número de soportes para hélices, por ejemplo. Esta puede estar hecha por diversos materiales, generalmente se trata de algún tipo de plástico, ya que es un material que tiene poco coste y pesa poco. Un ejemplo es el polipropileno, que es ligero y con mucha resistencia, lo que permite colocar sobre ella la batería. Otro material que suele utilizarse es la fibra de carbono, ya que se trata de un material que pesa poco y es muy resistente, aunque puede tener factores negativos como su conductividad. Por último, también nombrar la fibra de vidrio. Este material también es muy utilizado por ser ligero, y tiene características como que no es conductor de la electricidad. Es común ver estructuras híbridas entre distintos materiales, sobre todo juntando los dos tipos de fibra.

**Hélices:** Elemento formado por dos palas montadas de forma concéntrica sobre un eje, que al girar crean un par de fuerzas, permitiendo así el movimiento del dron.

## CAPÍTULO 1. INTRODUCCIÓN

---

**Motores:** Son los encargados de transformar la energía que llega en movimiento sobre el eje en el que se sitúan las hélices, para así permitirles a estas hacer su trabajo. Este a su vez tiene distintos parámetros que serán principalmente los que permitan al dron llevar mayor velocidad.

El número de vueltas que dé por minuto, lo que dependerá de los KiloVoltios. Este suele estar en torno a 800-900kV.

El tamaño que éste tenga. Al mirar las especificaciones de un dron está en un número de 4 dígitos, en el que los dos primeros hacen referencia al tamaño del rotor y los otros dos al tamaño de la bobina.

El empuje, valor que hace referencia al peso que puede levantar el motor.

La corriente, que se trata de la energía (amperios) que se consume cuando el motor está al máximo.

**Batería:** Encargada de proporcionar la energía suficiente para que el dron pueda realizar un vuelo, permitiendo trabajar a la placa controladora y motores. La característica principal de las baterías son los miliamperios, ya que es la que permitirá una mayor capacidad y por lo tanto que el dron tenga un mayor tiempo de vuelo. Existen baterías de muy diversos tamaños, desde los 350 mah en drones de juguete a, por ejemplo, los 4500mah que tiene la batería del dron 3DR solo. También es importante la tasa de descarga, que se trata de la máxima energía que puede entregar y el periodo de tiempo durante el que puede hacerlo. Normalmente los drones traen sistemas de alerta que avisan cuando a la batería le queda poca energía, o que cuando queda un valor menor a cierto porcentaje de carga no permite despegar el dron, evitando así que se quede sin energía a mitad de un vuelo.

**Equipo de transmisión:** Es el encargado de que se comunique el dron con una estación receptora. Este puede variar en función del aparato ya que se pueden usar diferentes tecnologías, pero principalmente se trata de radiofrecuencia o de Wifi. Existen casos, como el modelo 3DR que combina ambas tecnologías, utilizando la radiofrecuencia para la información del movimiento, batería y posicionamiento, y el

## CAPÍTULO 1. INTRODUCCIÓN

---

WiFi para la transmisión de imágenes en directo. Podemos encontrar distintos equipos de sistemas de transmisión, uno de los últimos y más destacables es **Hyperion**, que utiliza un sistema óptico de comunicaciones capaz de transmitir hasta 1Gb por segundo, lo que permite la transmisión de datos mediante la luz directa. La principal característica de éste es que no pierde información cuando no hay contacto directo entre las dos estaciones. Pero vía WiFi es algo muy utilizado en los últimos momentos, pues permite controlar el dron desde una aplicación móvil, por lo que conectando estos dos tendríamos un mando que nos permite cambiar gran parte de la configuración del dron. También existen dispositivos que permiten el control mediante Bluetooth, pero este es menos común ya que tiene mayor restricción de velocidad de datos y distancia.

**Placa controladora:** Es el procesador del dron, el que se encarga de recoger la información del dron y cuando le llega una orden ver qué información tiene que mandar para que ésta se ejecute de forma correcta, así como en caso de haber un problema tratar de evitarlo. Este es básicamente el hardware que utiliza el dron, hay una gama muy amplia dentro de éste, donde cabe destacar **Pixhawk**, pero podemos encontrar varias con gran trascendencia:

- **Pixhawk:** Este es el más utilizado debido a que trabaja con 3DRobotics y Ardupilot. Este sirve para diversos dispositivos como son drones, helicópteros y barcos. Está pensado para cualquier vehículo que tenga movimiento. Se trata de un proyecto hardware abierto, cuyo objetivo principal es proporcionar el hardware de autopiloto a comunidades académicas o gente que tiene esto como un hobby, teniendo así un bajo costo y una alta disponibilidad. Se trata de un piloto automático en tiempo real y muy eficiente, proporcionando un entorno de estilo POSIX. Este es el autopiloto estándar de la industria, y por lo tanto, como veremos a continuación, a partir del cual se han desarrollado diversos autopilotos con distintas mejoras.
- **Pixhawk2:** Es una versión avanzada de la placa anterior. Este tiene mejoras

## CAPÍTULO 1. INTRODUCCIÓN

---

como aislamiento de vibraciones, 3 IMUs para redundancia(3 acelerómetros, 3 giroscopios, 3 magnetómetros y 2 barómetros) y sensor para controlar la temperatura.

- PixRacer: Éste se ha desarrollado para los drones de carreras, aunque también se utiliza en minidrones. Suele tener una mayor memoria flash.
- Navio2: Piloto automático diseñado de Raspberry Pi. Te permite convertir esta en un controlador de drone.
- PXFmini: Se trata de otro piloto automático de Raspberry Pi. Este tiene la electrónica para la mayoría de los componentes que puede utilizar un dron.
- FlytPOD: Este se trata de una placa Odroid XU4 SBC junto con una PixHawk. Puede volar diversos vehículos aéreos y su principal característica es el WiFi que tiene integrado. Existe una placa FlytPOD pro que se trata de una versión extendida de la anterior, teniendo todas sus características, pero con mas sensores y mayor capacidad de almacenamiento.
- U-Pilot: Este hardware se caracteriza por servir para diversos vehículos aéreos, siendo programable para realizar todas las acciones de su camino de forma automática. Su radioenlace con frecuencia en torno a 900Mhz permite controlar el dispositivo a una distancia de 100km.

Hay que destacar un elemento importante como es la **camara**, que aunque no todos los drones la llevan sí que es algo bastante común. Algunos la llevan incorporada (incluso dos camaras, una que apunta hacia la parte de delante y otra que apunta la parte de abajo) y otras que traen soporte para poder incorporar ciertas camaras, normalmente consideradas camaras de acción, para así obtener una mejor calidad, e incluso incorporar adaptadores como puede ser una Gimbal para controlar la parte hacia la que queremos que apunte la camara en cada momento o utilizarlo

## CAPÍTULO 1. INTRODUCCIÓN

---

como estabilizador, para evitar así que afecten a la imagen diversos movimientos, generalmente bruscos, que pueda realizar el dron. Cabe destacar que en muchas ocasiones, utilizado normalmente para carreras de drones, la cámara sirve para integrar la tecnología FPV (First Person View), que es junto a la cámara, el transmisor de vídeo y el receptor de vídeo, poder ver en tiempo real las imágenes sobre una pantalla LCD o utilizando unas gafas de realidad virtual. En este aspecto, en los últimos años se ha visto por otro lado un gran avance de la realidad virtual, y es también hay una gran variedad en este mundo, pues existe una gama que va desde las *cardboard*, que permiten con un trozo de cartón y un par de lentes, poniéndolos de cierta forma y con el uso de un smartphone, tener de forma sencilla unas gafas 3D, hasta gafas para la videoconsola que te permiten entrar en el videojuego, añadiendo una gran calidad de imagen y con un sonido envolvente para entrar de lleno en el ambiente. Pero una de las cosas más impactantes es el conjunto que se ha creado con el dron **FLYBi**, estando éste conectado a unas gafas de realidad virtual que tienen sensor de movimiento, lo que te permite sentir que eres tú el que vuelas y el que estás en el lugar del dron, y con cualquier movimiento que sientan las gafas la cámara del dron lo imitará. En caso de que esto parezca incomodo el dron tiene un joystick con el que enviará la información de los elementos a realizar a la cámara.

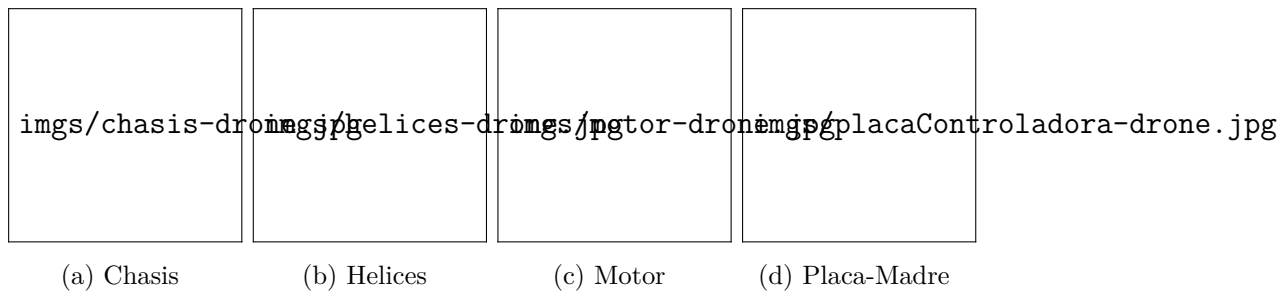


Figure 1.2: Distintas partes del dron.

### 1.3 Software.

Como sabemos, el software es el conjunto de programas que van a permitir realizar ciertas tareas, en este será lo que permita realizar al drone una navegación autónoma. Dicho programa estará instalado en la placa controladora, y por tanto sera el que ejecute para recoger la distinta información de los sensores, procesar la información y enviar las ordenes correctas a los distintos elementos de este. Como se ha comentado anteriormente, el desarrollo de software para este tipo de robots ha evolucionado mucho en los ultimos años debido al uso civil que se le comienzan a dar y no tanto al desarrollo militar. Es importante destacar el software de **ArduPilot**, ya que se trata del software auto-piloto mas avanzado, pero existen también otros que nos permiten el manejo de estos robots:

- **Ardupilot:** Es un sistema OpenSource encargado de recibir la información que se le da y de esta forma enviar las señales correspondientes a los actuadores. Se trata del software mas importante debido a lo completo que es y confiabilidad que proporciona, debido a la gran cantidad de gente que lo utiliza (pilotos de drones profesionales y aficionados) y por el equipo de ingenieros que lo ha desarrollado. Este software se caracteriza por la variedad de dispositivos que puede llegar a controlar, ya que trabaja con diversos dispositivos aereos (aviones, helicopteros, drones, etc) y con dispositivos marinos (como pueden ser los barcos y submarinos). Este ha tenido un gran desarrollo debido a su principal característica: Opensource. Hay mucha gente creando interfaces para este y dichos usuarios comparten sus avances con el resto. A partir de éste han nacido controladores como Ardupilot Mega. El problema que tiene dicho software es que solo permite trabajar con plataformas de los mismos creadores, lo que lleva al siguiente software. Otra característica es la facilidad con la que se le pueden añadir diferentes sensores, como pueden ser modulos GPS o camaras, algo que facilitará la navegación autónoma.

- **Megapirate-NG:** Apareció como desarrollo del anterior. La funcionalidad de uno y otro es prácticamente la misma, con la diferencia de que éste permite trabajar con Hardware de otros creadores. El problema es que siempre depende de Ardupilot, por lo tanto sus funcionalidades, aunque sean mas cómodas para trabajar, puede que esten atrasadas.
- **MultiWii:** Éste se propuso como radiocontrol para drones. Es un sistema que fue creado por los desarrolladores y con los sensores (giroscopios y acelerómetros) de la Nintendo Wii. Es una plataforma basada en arduino, con el factor en contra de tener una funcionalidad bastante limitada.

Estos primeros son los softwares principales que estarían sobre el vehículo, pero también estan los programas que se ejecutarían en otros dispositivos como el ordenador o el teléfono móvil para ver la información que este nos envía. Normalmente el fabricante del drone tiene ya un programa que realiza esta función.

En este punto es importante el protocolo de comunicación que habrá para comunicarse el vehículo con la estación terrena. Aquí hay un protocolo que destaca sobre los demás, el **MAVLink** (Micro Air Vehicle Communication Protocol). Éste protocolo tiene la información contenida en ficheros .xml, lo que permite utilizarlo en diversos lenguajes de comunicación, lo que conlleva una mejora notable en su desarrollo. Al tener el fichero .xml los tipos de mensaje, permite con facilidad añadir nuevos tipos para asignar una tarea nueva a cada uno. Otra ventaja de este es que hay muchos software de drones que lo soportan, como pueden ser Ardupilot, Autopilot, algunos derivados de estos y otros como Gentlenav o Flexipilot, y desde la estación tierra algunos como MAVProxy, Mission Planer o APM planner. Un problema en este protocolo es que los datos no estan encriptados en la comunicación, por lo que es mas facil un ataque y que se manipulen los datos, siendo detectable si se hace perder algun dato ya que se utiliza CRC (codigo de redundancia cíclica para detectar cambios en los datos, este se utiliza en protocolos como TCP). MAVLink utiliza otro software

## CAPÍTULO 1. INTRODUCCIÓN

---

llamado MAVProxy para poder acceder a los datos del vehículo, como la velocidad y las imágenes, lo que nos permitirá también saber que datos mandarle para que funcione de forma correcta.

A parte de todos estos, existen también otras infraestructuras software como puede ser **JdeRobot**. En sí JdeRobot se trata de un software de desarrollo para robótica y aplicaciones de visión por computador. Éste puede trabajar con distintos sensores que le proporcionan información, en caso del drone con la información que le permite Ardrone Server, y gracias a esto puede controlar el dispositivo y ver los distintos datos de este. Con JdeRobot es muy sencillo tomar el control del hardware gracias a su programa de control. Tiene una gran API que permite realizar diversas tareas como trabajar con aparatos reales o simulados, y conectarse a ellos de forma local o a través de la red. Decir que para el trabajo realizado, esta plataforma ha sido muy importante, pues su componente para la cámara, su aplicación para filtros de color y la posibilidad de realizar diagramas de estado permite tener en todo momento datos importantes del dron y poder mantenerlo bajo control sin ningun problema. Destacar que se trata de un sistema OpenSource que permite trabajar con simuladores como Gazebo y con ayuda de librerías como OpenCV.

A parte de todos estos, existen también otras infraestructuras software como puede ser **JdeRobot**. En sí JdeRobot se trata de un software de desarrollo para robótica y aplicaciones de visión por computador. Éste puede trabajar con distintos sensores que le proporcionan información, en caso del drone con la información que le permite Ardrone Server, y gracias a esto puede controlar el dispositivo y ver los distintos datos de este. Con JdeRobot es muy sencillo tomar el control del hardware gracias a su programa de control. Tiene una gran API que permite realizar diversas tareas como trabajar con aparatos reales o simulados, y conectarse a ellos de forma local o a través de la red. Decir que para el trabajo realizado, esta plataforma ha sido muy importante, pues su componente para la cámara, su aplicación para filtros de color y la posibilidad de realizar diagramas de estado permite tener en todo momento datos



## CAPÍTULO 1. INTRODUCCIÓN

---

importantes del dron y poder mantenerlo bajo control sin ningun problema. Destacar que se trata de un sistema OpenSource que permite trabajar con simuladores como Gazebo y con ayuda de librerias como OpenCV.

Ya que este proyecto se ha desarrollado en python, hay que destacar algunas librerias y herramientas muy importantes para la creación de un buen software, lo que permitirá una mejor y más rápida ejecución.

- OpenCV: Se trata de una biblioteca de visión artificial. Esta biblioteca está implementada en C++, pero se puede utilizar en los distintos lenguajes como C++,C, python y Java. Es una biblioteca utilizada por miles de usuarios y tiene funciones que te permiten detectar movimiento, reconocer objetos, reconocimiento facial y trabajar sobre imágenes (modificando datos de éstas), mostrando, guardando y creando nuevas imágenes, cambios de valor sobre pixeles concretos para que sea mas sencillo ver el proceso que se esta realizando. Está disponible para los diversos sistemas operativos y se utiliza mucho para software de visión. Se trata de una librería opensource y por lo tanto su desarrollo puede avanzar a gran velocidad ya que todo el mundo puede contribuir y compartir este.
- PIL: Esta librería permite la edición de imágenes desde python. Tiene una gran variedad de operaciones que permite el realizar diversos cambios como puede ser rotación, escalado o manipulación de píxeles en imagenes, además de cargar y guardar estas para trabajar con ellas o una vez estén modificadas.
- NumPy: Se trata de un paquete en python que permite la computación científica. La principal característica son las matrices multidimensionales que permite hacer y el conjunto de funciones matemáticas que tiene para poder operar. NumPy permite ejecutar a gran velocidad las operaciones deseadas. En ocasiones los datos obtenidos en operaciones con la biblioteca OpenCV son almacenados en matrices NumPy, pues en realidad estas imágenes son matrices de datos a

## CAPÍTULO 1. INTRODUCCIÓN

---

las que se le dan ciertos valores para luego poder representarlas, y acceder a estas estructuras se puede hacer de forma mas sencilla con un bajo coste computacional, además de permitirte interoperabilidad con otros paquetes como `scipy` o `matplotlib`.

- **Scipy:** Esta librería Opensource permite trabajar con diversos algoritmos matemáticos de gran capacidad ya que tiene modulos para la optimización de funciones.

Cabe destacar las diferencias que tiene este lenguaje de programación respecto a otros. La principal diferencia es que se trata de un lenguaje interpretado, por lo que podriamos decir que sus programas hacen una compilación en directo, es decir que mientras compilan ejecutan, lo conlleva que no haya errores de compilación sino que sean errores en ejecución. Este tiene ventajas como la simplicidad para escribir el codigo, ya que te permite hacerlo de forma mucho mas limpia y abreviada que en otros lenguajes. No te obliga a declarar el tipo del que es una variable, simplemente con asignarle un valor ya te deja trabajar con el, a diferencia de otros como C o Java que si hay que declararlo. Todo esto conlleva que el tiempo de desarrollo en un programa se reduce de forma notable, permite que sea mejor y mas fluido el trabajo en equipo ya que si se crea un codigo legible es mas facil de entender y te permite no estar preocupado por la reserva y liberación de memoria. Tiene también factores en contra como pueden ser el rendimiento, pues para hacer la misma operación necesita mas tiempo que otros lenguajes, y emplea mas memoria. Destacar que se esta trabajando en esto ya que se considera una desventaja importante, y se han creado herramientas como **Numba**, que con pocas lineas permite que este acelere su ejecución, hasta veinte veces su velocidad.

### 1.4 Uso actual de los drones

Teniendo ya el contexto de las distintas características que tiene un drone, se va a proceder a describir los distintos usos que se le pueden dar a estos. Cabe destacar que esto comprende distintos campos y todos muy diversos, por lo que al principio se van a comentar los usos que mas se ven en la sociedad en general para terminar viendo usos mas especificos de la robótica aerea, concretamente los que llevan a que este pueda trabajar de manera autónoma.

#### 1.4.1 Medios Audiovisuales.

El drone es un elemento que se ha incorporado ultimamente en este sector debido a la camara que pueden tener. Gracias a esto permite tomar planos de ciertas zonas o fotografias que serían muy difícil de obtener en condiciones normales. También se debe a que el precio de estos es asequible, por lo que se puede acceder a ellos con facilidad, y en un sector tan amplio y vistoso como es este, lleva a un uso cada vez mas común.

#### 1.4.2 Control de distintas zonas.

En este area podemos destacar varios usos distintos, pero su expansión en esto se debe a que con un drone podemos controlar una zona para la que anteriormente necesitabamos varias camaras, y aun asi podian quedar zonas sin vigilar. Esta tecnología nos lleva a poder mover una camara por un lugar amplio sin necesidad de estar alli ni de tener un gran despliegue de elementos, asi como evitar que queden puntos muertos. Algunos ejemplos de esto son los siguientes:

- **Seguridad:** Teniendo un drone en un lugar como pueda ser una nave industrial, podemos hacer que este se desplace grabando en todo momento lo que ve, y si se detecta algo sospechoso en algún lugar el drone se diriga allí en el momento para obtener imágenes de lo que esta pasando.

- **Sector agrícola:** Se debe a la facilidad con la que un drone puede sobrevolar una zona y ofrecer imagenes de alta calidad de esta, obteniendo un buen control de los cultivos en menos tiempo, con menos gasto y al tratarse de un vehiculo electrico al evitar desplazamientos de otros automoviles conlleva un menor impacto ambiental. Además, este permite ver con facilidad el estado de la cosecha, detectar enfermedades o plagas, permiten fumigar desde el aire con mayor precisión, ya que les puedes programar una ruta y que sigan esta. También podrian obtener otros datos como las zonas con mas y menos agua, y obtener las condiciones del terreno y ver si son optimas para esperar cierto resultado.
- **Mantenimiento:** En este apartado podemos englobar diversas actividades, como puede ser mantenimiento de edificios y construcciones, redes electricas o diversas instalaciones industriales como aerogeneradores eólicos y estados de paneles solares. Al igual que en el apartado anterior, aqui podemos recorrer grandes distancias, por ejemplo para comprobar las redes electricas, sin la necesidad de que un operario pierda mucho tiempo recorriendo dicha línea. Con las instalaciones solares por ejemplo, desde un plano superior podriamos observar si todas las placas estan en las condiciones optimas y en caso de existir algun fallo poder identificarlo con facilidad. Para edificios y aerogeneradores lo que hay que tener en cuenta es la altura que estos pueden tener, y a la cual con un dron llegaríamos con facilidad y observaríamos si hay algun problema. En estos casos lo que evitamos, como anteriormente he comentado, es que alguien tenga que ir sitio a sitio perdiendo mucho tiempo. Destacando también que este tipo de actividades se pueden implementar programas que directamente detecten las anomalias, sin necesidad de que haya una persona revisando en todo momento las imágenes, donde también con una inversion inicial, al final ahorrariamos mucho tiempo y dinero.

Por otro lado, pueden aportar gran ayuda en situaciones que no se dan de

## CAPÍTULO 1. INTRODUCCIÓN

---

forma periódica, sino que ocurren de forma esporádica y en lugares muy distintos. Gracias a los drones podemos tener una cámara que nos muestre una imagen de esta zona o que nos permita ayudar allí, y en otro momento llevarlo a otro lado, lo que lleva a no tener un gran despliegue de medios en un lugar que apenas va a ser necesario. Los ejemplos podrían ser los siguientes:

- **Emergencias:** Cuando ocurren ciertas catástrofes naturales por ejemplo, son de gran ayuda debido a la velocidad con la que pueden llevar materiales (médicos o de otro tipo) a la zona afectada.
- **Busqueda de personas:** Al tratarse de medios que pueden sobrevolar zonas obteniendo grandes imágenes, pueden ser de ayuda cuando montañeros o caminantes tienen accidentes en bosques o montañas, quedan incomunicados y se comienza una búsqueda.
- **Incendios forestales:** Los drones pueden estar sobrevolando zonas y obteniendo información de esta, para así poder prevenir posibles incendios o alertar lo antes posible en cuanto uno ocurra.
- **Investigaciones:** Para este apartado pueden ser de distinto tipo, ya que puede recorrer zonas largas de restos arqueológicos y tomar datos de estos, entrar en zonas peligrosas como ciertas cuevas o volcanes y obtener datos para su posterior estudio, e incluso investigaciones biológicas, recreando en un dron la actividad de un ave para así estudiar su comportamiento.

### 1.4.3 Desarrollo autónomo.

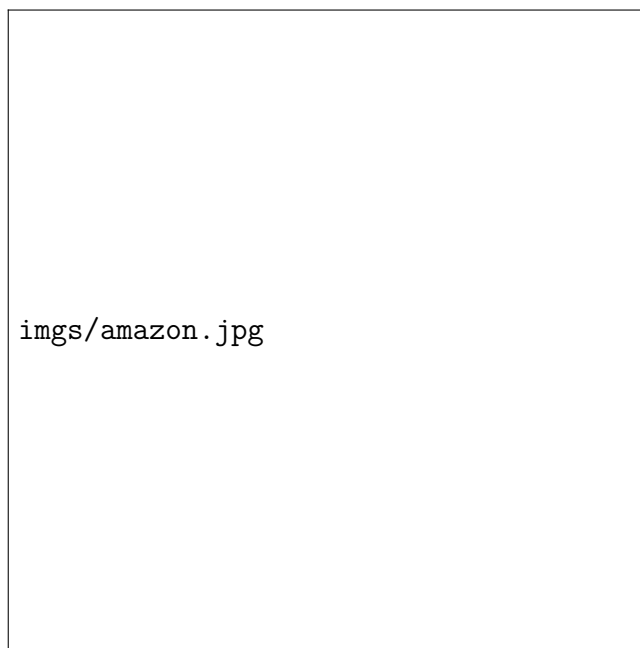
Es muy importante este ámbito, pues determinados grupos y grandes empresas están trabajando en su desarrollo debido al futuro que se viene por delante. Esto se debe a la facilidad que puede llevar esto para realizar grandes tareas, como el control de

## CAPÍTULO 1. INTRODUCCIÓN

---

material en almacenes o transporte de material de un lugar a otro. Una característica importante es la rapidez con la que pueden llegar los drones de un lugar a otro, y acceder a lugares que es difícil por otros vehículos. A continuación se muestran distintos usos que están aplicando sobre estas grandes empresas.

Por un lado, como pionero en este área se encuentra **Amazon**, el cual lleva desarrollando desde 2013 una tecnología que permita el reparto de paquetes mediante drones. Esta tecnología lleva consigo los llamados almacenes aéreos, es decir, un almacén que se mantendría en el aire gracias a dirigibles, el cual tiene paquetes a entregar y drones. El dron obtendría el paquete que se debe entregar y lo llevaría al lugar adecuado. Tras esto volvería a un almacén hasta que se le mande de nuevo al almacén aéreo para el siguiente reparto. Estos drones sabrían en todo momento en el estado y en el punto en el que se encuentran, es decir, que saben a qué lugar deben ir dependiendo de la tarea a realizar.



Por otro lado, también para el reparto de mercancías se encuentra **Google**, este empezó sus pruebas en 2014 y tuvo un programa piloto en Australia, pero

## CAPÍTULO 1. INTRODUCCIÓN

---

no consiguio llevarlo a Estados Unidos. Aun así, consiguio hacer pruebas en una universidad de reparto de comida, un reto que supuso principalmente que la comida llegara rapido a su destino y en buenas condiciones. Además también sirvió para ajustar los sistemas automáticos de vuelo y entrega de la mercancía.

A raíz de estos servicios de entregas, se ha producido otro desarrollo importante, como puede ser el de tener controlados los paquetes dentro de un almacén. Un pionero de esto ha sido un grupo en el **MIT**, desarrollando un sistema que permite a los drones moverse por los almacenes escaneando los codigos de cada paquete, enviando esta información a un servidor y que este pueda tener controlados los paquetes que hay y donde estan situados.

Para el traslado de mercancías en interiores también se ha puesto en marcha la cadena de supermercados estadounidense Walmart, cuyo objetivo es transportar productos de un lugar a otro previamente establecidos. La idea de este proyecto se debe a los grandes almacenes que tienen estos supermercados, y que cuando un cliente no encuentra el producto deseado, avisa a un empleado y este tiene que ir al almacén a buscarlo, perdiendo mucho tiempo entre la distancia recorrida y la búsqueda del producto. Lo que conseguirían con esto, es que en caso de que los empleados esten ocupados, un cliente no tenga que estar a la espera, sino que con un dispositivo podamos pedir el producto y un drone se encargará de ir a por el y traerlo al punto donde nos encontremos. Destacar que se incorporaran en las tiendas controladores aéreos para que los vehículos sigan una trayectoria segura.

# Chapter 2


## Objetivos

En este capitulo se van a explicar los objetivos a conseguir y el metodo utilizado para llegar a ello.

### 2.1 Objetivo principal.

El objetivo propuesto para este trabajo era conseguir que un drone navegara de forma autónoma. Para ello la idea era situar el drone en un punto de inicio, despegara sobre este y una vez se hubiera estabilizado comenzara a desplazarse siguiendo un algoritmo de búsqueda previamente programado. La idea de este es recorrer determinado area sin la necesidad de pilotar el drone. Los movimientos de este drone dependerian de lo que detectara la cámara en cada momento. Esta búsqueda se realizaría con la finalidad de encontrar una baliza sobre la cual aterrizar. En caso de no ver la baliza, el drone se moveria realizando una espiral en busqueda de esta, en el caso de encontrarla el drone se situaría sobre esta y una vez centrado aterrizaría. Esta baliza seria un cuadrado formado por cuatro cuadrados de colores distintos.





imgs/baliza.jpg

Para conseguir este objetivo, hay que tener en cuenta varias partes importantes, de las cuales a partir de una pasamos a la otra para al final llegar a lo que permitirá el correcto funcionamiento del algoritmo: que los movimientos de este dependan de lo que ve la cámara.

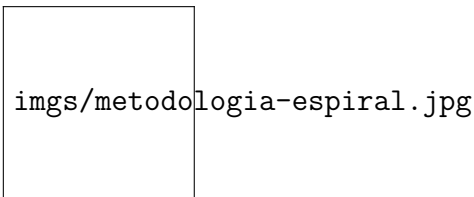
- **Creación de un filtro de color:** Gracias a los colores podemos detectar los cuadrados de la baliza, por lo que realizando este filtro obtendremos estos objetos y los que sean del mismo color, eliminando así información no necesaria de la imagen.
- **Detección de objetos:** Tras el filtro de color y eliminada información no necesaria, podemos obtener distintos objetos en la imagen, viendo si cumplen características como tener un determinado area, para así saber si son los objetos que se desean o no y trabajar con ellos o descartarlos.
- **Movimiento del drone:** Una vez obtenemos información de la imagen y la hemos procesado, en función de esto se pretenderá que el desplazamiento del drone sea uno u otro, trabajando asi de forma fluida y sin movimientos bruscos que puedan desestabilizarlo.

## 2.2 Metodología.

Para poder conseguir todo esto, era necesario saber las plataformas con las que sería posible, donde tiene gran importancia el software JdeRobot, pues gracias a su desarrollo que permite la comunicación con el dron, y de estas forma obtener los datos

de sus sensores, principalmente de la cámara. Para este apartado fue primordial el aprendizaje de sus distintas herramientas, hubo que hacer pruebas con estas y ver su funcionamiento tanto en entornos simulados como en reales.

Una vez se realizaron las primeras pruebas con robots, se propuso un desarrollo en espiral. Para este desarrollo se proponían semanalmente reuniones con el tutor, en las cuales se proponían unos objetivos a seguir en función de lo que se había conseguido hasta el momento. Una vez propuestas estas tareas se evaluaban los distintos riesgos que se podían tomar en función de trabajar de una forma u otra, y los avances a los que se podía llegar por cada camino. Una vez hecho esto se comenzaban a desarrollar los algoritmos para conseguir estos objetivos en función de la forma elegida. Una vez se realizaban estos y se probaban (en simulación o con el robot real, dependiendo del objetivo), se proponía otra reunión para determinar si los avances eran los deseados o no, y en función de esto planear unos objetivos u otros.



Los avances que se iban realizando semana a semana, se subían mediante videos o imágenes a un mediawiki en JdeRobot, quedando reflejados todos los procesos. También se ha utilizado GitHub para subir el código desarrollado.

### 2.3 Plan de trabajo.

El primer punto fue comprender las distintas herramientas de JdeRobot y trabajar con ellas. Creando programas simples y viendo que funcionaban, así como trabajar con distintos robots reales para tener una toma de contacto con ellos y no basarse solo en un trabajo sobre simulador.

## CAPÍTULO 2. OBJETIVOS

---

En segundo lugar hubo que comprender la importancia que tendria la biblioteca OpenCV en este proyecto y ver sus multiples opciones, todo lo que permitia trabajar con imagenes y los cambios que podiamos hacer sobre estas para obtener datos deseados a partir de los cuales mandar información.

Por ultimo quedaba unir estas dos tecnologías y conseguir un buen algoritmo que nos permitiera enviar información a tiempo real a un drone y que este la ejecutara de forma correcta y fluida.

## Chapter 3

### Infraestructura utilizada.

En esta parte voy a hablar sobre los diferentes elementos y aplicaciones para el desarrollo de este proyecto.

En primer lugar hay que hablar del **Parrot AR.Drone 2.0**, dron utilizado para realizar las pruebas y poder comprobar de esta manera que el desarrollo del programa era el correcto.

La *batería* de este era de 1500mah, lo que en algunas ocasiones nos ha llevado a problemas, pues el tiempo de vuelo con esto es unos 12 minutos aproximadamente. Esto conllevaba que algunas pruebas en ocasiones eran muy difíciles de realizar, ya que si no salía el resultado esperado en los primeros intentos muchas veces había que parar la prueba durante bastante rato para continuar. Si las pruebas no eran de vuelo sino que era de imagen con este tiempo era suficiente, ya que el dron no gastaba tanta energía al relizar solo la retransmisión de los datos que obtenía la cámara. Un detalle curioso en este punto es el cambio de funcionamiento que se notaba una vez la batería estaba por debajo del 30/100, pues el despegue se hacía con menos fuerza y de forma más inestable, además las instrucciones de movimiento que se le mandaban las ejecutaba con menos fuerza y por lo tanto de forma más lenta. La única solución posible para remediar este problema fue trabajar a la vez con 3 baterías, pero he de decir que

### CAPÍTULO 3. INFRAESTRUCTURA

---

en momentos que las pruebas eran muy continuas durante mucho tiempo, llegaba un momento que había que parar porque el tiempo de carga no era suficientemente rapido comparado con el de las 3 descargas.

El *alcance* que tiene éste con la estación tierra es de 50 metros, suficiente para las pruebas que han sido realizadas en este proyecto.

Este dron tiene una *placa base* ARM Cortex A8 de 1Ghz y un DSP de vídeo de 8Ghz. También posee una memoria RAM DDR2 de 1GB y 200Mhz. Este chip es el encargado de levantar una red WiFi, a la cual nos podemos conectar mediante el smartphone (hay una aplicacion específica para ello), o como en este caso, mediante el ordenador, y de esta forma poder controlarlo. Esta placa base funciona con Linux.

Además, dicho dron cuenta con dos *cámaras*, una horizontal y otra vertical, lo que nos permite ver diversos planos en todo momento. Estas cámaras están conectadas a la placa base, lo que permite en todo momento la transmisión de imágenes en directo, gracias a las cuales podremos trabajar, sera la información gracias a la cual podremos darle instrucciones al dron. Cabe destacar la diferencia de calidad que hay entre las cámaras, algo que se notaba al realizar pruebas con una cámara, y cuando todo era correcto y querias cambiar de cámara para realizar otra prueba se podía observar como los colores perdian nitidez y se despreciaban algunos detalles, lo que te llevaba a cambiar cosas de los algoritmos que en un principio no se esperaban.

Ya para terminar, la *velocidad* de este drone puede llegar a alcanzar los 18km/h, algo que en este proyecto nunca he llegado a probar, pero sería algo poco aconsejable teniendo en cuenta la distancia de alcance, pues los 50 metros que tiene, si la estación terrena de comunicación esta en un punto fijo, el dron tardaría 10 segundos en salirse de este rango.

Tras este dron, tengo que hablar del dron **3DR solo**. En un principio era este sobre el que iba a funcionar el software realizado en el proyecto, pero al final hubo diversos problemas debido al software del dron, pues no lleva una cámara integrada, sino que es una GoPro que va por otra parte la que se puede añadir para utilizar, se

### CAPÍTULO 3. INFRAESTRUCTURA

---

estuvieron valorando diversas opciones para que este tuviera una cámara de visión que contaré más adelante, pero dado que era un dron con unas muy buenas características y que se trabajaba de forma excelente con él, voy a comentar algunos detalles suyos.

Lo primero es que cuenta con *dos procesadores*, uno en el dron y otro en el mando de control. Destacar que la del mando del control se trata de una Pixhawk de 3DRobotics. Es importante decir que el dron y el mando se comunican mediante radiofrecuencia, y es el mando el que levantaba la red WiFi para poder conectarte al sistema. Esto es otro detalle negativo que tuvimos al trabajar con este dron, pues no podíamos prescindir del mando si en algún momento el proyecto llega a tal punto, lo que lleva al software a tener una limitación de distancia, aunque ésta sea considerable. Desde el mando era el que se le enviaban las instrucciones al dron, pero con software específico se le podía enviar las instrucciones al mando y ya este comunica con el dron.

la *imagen en directo* la podíamos ver a través del teléfono a gran calidad (añadiendo la GoPro) pero situando la colocación de ésta antes del despegue.

Otro punto muy importante era la *batería*, esta tenía 5.200mAh de capacidad, que siempre nos fue suficiente para las pruebas que se realizaron, pero permite un tiempo de vuelo de 20 minutos. Decir que con este dron hubo menos tiempo de vuelo que con el anterior.

Y por último hay que hablar sobre la potencia que tiene el *motor* que permite alcanzar los 88.5km/h. Un inconveniente de tan alta potencia, son las corrientes internas que puede generar. En exteriores esto no supone nada ya que cuando se levanta aproximadamente un metro sobre el suelo, es capaz de estabilizarse él solo. Sin embargo, en interiores puede desestabilizarlo bastante, y como ocurrió en una prueba, ir dirección a la pared y sufrir algún daño como el romperse una hélice.

Dejando de un lado los drones, otro material con el que se trabajó fue la **Intel Compute Stick** es un dispositivo del tamaño de la palma de la mano que nos permite llevar un ordenador encima en todo momento (dispone incluso una pequeña pestaña para poder añadirlo al llavero) y que podemos conectar a cualquier pantalla

### CAPÍTULO 3. INFRAESTRUCTURA

---

con conexión HDMI. Además tiene un puerto USB 3.0, al cual podemos conectar un hub para poder añadir un teclado, ratón u otro dispositivo de almacenamiento de datos. Éste necesitaba un cable de alimentación para poder iniciarse. Su uso muy favorable, pues gracias a él podríamos llevar el software realizado encima en todo momento, con todas las necesidades para poder conectarlo a un dron y de esta forma el algoritmo pudiera ejecutarse sin problemas, ya que gracias a su antena WiFi podrá conectarse al dron para poder comunicarse. Por otro lado, se nota que no está preparada para grandes tareas de ejecución (tiene 2Gb de memoria RAM), y por tanto, a la hora de ejecutar programas pesados, la velocidad de éstos se ve notablemente disminuida, factor negativo teniendo en cuenta lo importante que es la velocidad de procesamiento de la información y transmisión de las instrucciones.

También ha sido muy importante para el proyecto el poder probar los distintos avances en el simulador **Gazebo**, el cual nos posibilita hacer pruebas en cualquier momento y sin preocuparnos de daños materiales. Gazebo permite la creación de entornos de manera precisa y poder probar los robots en distintos tipos de ambientes. Se trata de un programa OpenSource, lo que ha permitido su expansión con facilidad, y muchos añadidos como plugins o repositorios con robots comerciales para poder acceder a su uso. Esta plataforma ha sido en la que se ha trabajado principalmente durante todo el proyecto, pues existe en ésta un simulador de ArDrone, el cual se controlaba a través de las herramientas de JdeRobot. El escenario principal aquí utilizado consistía en el dron y un coche con una baliza, el cual tenía que encontrar, centrarse sobre ella y aterrizar. Cabe destacar que había una gran diferencia en los movimientos comparando con el dron real. Esto ha sido un problema en los periodos que se ha ido desarrollando el algoritmo en las dos partes a la vez, pues un mínimo retoque en Gazebo podía suponer un gran problema en el real, aunque para probar como podían ser los distintos movimientos ha sido de gran ayuda, así como la parte de visión, ya que era muy sencillo comprobar cualquier modificación, ver si algo funcionaba sobre un entorno perfecto y de esta forma haber mejorado esta parte continuamente.

Como software principal se ha trabajado con **JdeRobot**. De aquí se han trabajado con diversas herramientas, las cuales nos han permitido seguir unos pasos firmes desde un inicio hasta el final del desarrollo. En primer lugar se utilizó *color filter* para ver el funcionamiento de los filtros de color. Esta herramienta nos permite, a partir de una imagen o vídeo, trabajar en los distintos espacios de color (como puede ser RGB, HSV, HSI...) y poder ir variando los parámetros máximo y mínimo (entre 0 y 255) de cada uno para ver que colores cumplen las características y cuales no, viéndose de esta forma, en otra imagen, sólo los colores que pasan este filtro dejando el resto como un fondo negro. Una vez utilizada esta herramienta y realizamos pequeños algoritmos para ver que funcionaba igual un filtro propio que este filtro, pasamos a trabajar con otra llamada *follow turtleboot*. Ésta nos permitía manejar el dron, teniendo un cuadro que nos permite el movimiento en los ejes X e Y, una barra para realizar un cambio en la altura y un pequeño círculo para que pudiera rotar sobre si mismo. Por otro lado tenía un botón para aterrizaje y despegue y unos botones que permiten iniciar un algoritmo creado, de forma que podemos probar si lo programado funciona. Esta herramienta puede sacar tres cuadros diferentes, uno para la cámara, el cual nos da una imagen en directo de lo que está transmitiendo el dron y nos da la opción de elegir la cámara que queremos ver. El segundo, nos muestra las distintas propiedades del dron en ese momento como la orientación, altura o inclinación. Por último, el último cuadro es para el filtro de color. Éste funciona cuando el algoritmo programado está en marcha. En un primer momento como indica su nombre, nos permite trabajar con el filtro de color que habíamos practicado antes sobre la otra herramienta, pero al final utilizamos éste para realizar marcas sobre la imagen de las cosas que nos interesaban de esta, como puede ser encuadrar la baliza, la cruceta o pintar sobre esta. Por último, a todo esto le hemos añadido un diagrama de estados, gracias a una pequeña aplicación desarrollada a partir de *visualHFSM*. nos permitía crear un dibujo de los diferentes estados por los que debería pasar el dron, según lo que estuviera viendo en cada momento o la acción concreta que estaba realizando. De esta forma se marca en el diagrama en el estado en



## CAPÍTULO 3. INFRAESTRUCTURA

el que se encuentra, pudiendo saber así cual sería el siguiente estado esperado y viendo que funciona de forma correcta.

En la siguiente figura podemos observar un ejemplo de este conjunto, pudiendo ver el diagrama de estados y la imagen con los datos interesantes marcados:

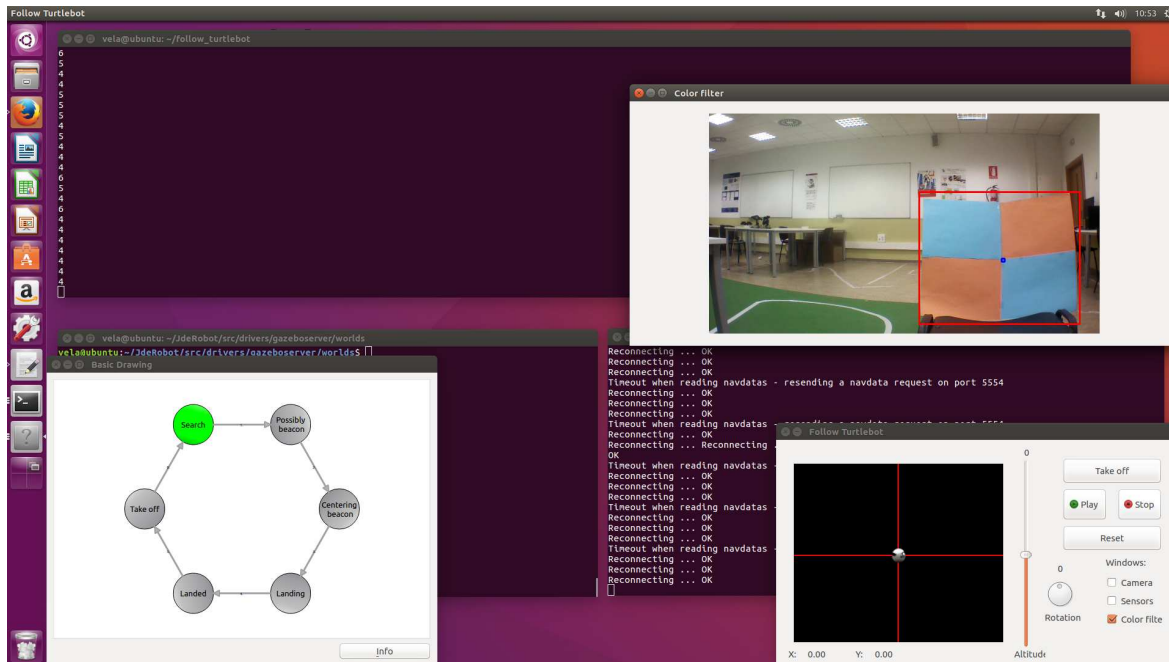


Figure 3.1: Esta imagen muestra el conjunto de herramientas utilizadas.

Ya por último, decir lo utilizado para programar el algoritmo. Este se ha desarrollado utilizando Python, gracias al cual hemos podido utilizar bibliotecas como PIL o OpenCV principalmente para el manejo de imágenes. Nos permitían obtener la imagen transmitida por el dron con facilidad y aplicarle los filtros de color deseados en cada momento, así como realizar distintas operaciones morfológicas (erosión, dilatación, apertura y cierre) sobre los fotogramas en cada momento, obteniendo imágenes más claras sobre las cuales podemos trabajar y con sus funciones, como puede ser drawcontours, obtener un código más compacto y eficiente. Esto nos permite un mejor rendimiento del algoritmo, lo que nos llevará a unos mejores resultados.

# Chapter 4

## Algoritmo

A lo largo de este capítulo voy a contar las distintas fases por las que se ha pasado hasta llegar al algoritmo final. Se va a poder ver como se han producido diversos cambios a lo largo del proyecto. Esto se debe a que debido a las pruebas que se iban realizando, veíamos los fallos e imperfecciones y tratábamos de arreglarlos. Además, según el punto en el que nos encontráramos nos interesaba más fijarnos en unos u otros detalles, por lo que no solo había cambios en la base del algoritmo principal sino también en el GUI (Interfaz Gráfica de Usuario). Para la explicación de esto, se abordará el tema por partes. En primer lugar se hará una explicación del diseño del algoritmo. Tras esto se explicarán las partes principales que tiene el procesamiento de la información (percepción, control, estados) para finalmente terminar explicando como sería una realización completa del conjunto.

### 4.1 Diseño.

El diseño de este algoritmo se trata de un ciclo continuo. Se trata de un proceso basado en adquisición-procesado-envío de datos. La parte de adquisición se realizará gracias a los sensores del dron, los cuales obtendrán cierta información.

Esta información será transmitida al dispositivo que la procese, y una vez hecho esto el dispositivo enviara instrucciones al dron, que seran las que indiquen la velocidad,dirección y sentido en el que este se tiene que desplazar. Tenemos los sensores del dron, de los cuales la camara es el mas importante para este trabajo. Lo primero que hace el algoritmo, siendo esto la parte de adquisición, es obtener la imagen. Dependiendo del punto del algoritmo en el que se encuentre, querra obtener una u otra información de la imagen,pudiendo clasificar esta parte como procesado de la información, para lo que siempre utiliza un filtro de color. Pongamonos en el caso de que el dron esta buscando una baliza en la que aterrizar(cuadrado que consta de dos colores), filtrara estos dos colores en la imagen obteniendo así los objetos de interes. En caso de ver que hay un objeto que posiblemente sea la baliza, se eliminaran los objetos que sean menores a un determinado area, para asi evitar posible ruido que se haya introducido en la imagen u objetos que sabemos que no son los que buscamos. Tras esto el algoritmo realizara las operaciones que seran explicadas mas adelante para asegurarse de que se trata de una baliza.Llegando en este punto al envio de datos. En caso de tratarse de una baliza, el algoritmo enviara las instrucciones correspondientes, indicandole los movimientos a realizar comportarse frente a esta. En caso contrario, el algoritmo enviara las instrucciones para que continúe con la busqueda.

### 4.2 Percepción.

En este apartado se tratará sobre el procesamiento de la imagen obtenida. Este puede ser el punto mas importante de todos, pues es gracias al cual el dron sabe en que punto del algoritmo se encuentra y que información enviar. Las primeras pruebas no eran muy complejas, obteniamos una imagen RGB que era la que se transmitia, la convertiamos a HSV para que fuera mas facil su interpretación, debido a que no tenemos tres colores a los que tratar sino tres parametros( Matiz, saturación y valor). Lo que se conseguia con esto era que un color no dependiera tanto de las condiciones del

medio, pues variando poco los valores H y S no hay una gran modificación en el tono, y por lo tanto la intensidad de la luz influirá de menor manera. Tras esto se realiza un filtro de color, eligiendo los valores de H,S y V entre 0 y 255, quedandonos solo con los objetos que nos interesaban. Una vez estabamos en este punto, si se trataba de una imagen perfecta, como pueden ser algunas de simulador no había problemas, pero en caso contrario, debido a factores como luces, sombras y reflejos, un color podia tratarse de distinta forma según el momento o que pasaran el filtro puntos de la imagen que no eran de las características deseadas. Por ello, era necesario el uso de operadores morfológicos, los cuales explico brevemente para que así se entienda su uso:

- **Erosión:** Dada una imagen y un elemento estructural, la erosión es el conjunto de los elementos  $x$  para los cuales el elemento estructural trasladado por  $x$  está contenido en la imagen.  
Aplicación: Cuando un pixel que parece pasar el filtro, pero los elementos de su alrededor(en concordancia con el elemento estructurante) no lo pasan, este pasa a ser parte del fondo.
- **Dilatación:** Transformación dual a la erosión. El resultado de esta es el conjunto de elementos tal que al menos algún elemento del conjunto estructurante esta contenido en  $x$ , cuando el elemento estructurante se desplaza sobre  $x$   
Aplicación: Pixeles que parecen de fondo, pasan a ser de la figura si estan cerca de pixeles que pasan el filtro.
- **Cierre:** Se trata de realizar una dilatación en la imagen seguida de una erosión.
- **Apertura:** Se trata de realizar la erosión en una imagen seguido de la dilatación.

Pues bien, gracias a esto se puede hacer un pre-procesado de la imagen(técnica mediante la cual se mejoran y realzan las características de esta para así facilitar las posteriores operaciones a realizar) y asi obtener una mejor imagen con la qe trabajar. Hay que destacar que las mas utilizadas durante la realización de este trabajo han sido

## CAPÍTULO 4. ALGORITMO

---

la erosión y la apertura. Pues con la erosión evitábamos que se colaran píxeles de fondo como parte del objeto, por lo que obteníamos solo la parte requerida, y por otro lado con la dilatación realizada en la apertura, tras eliminar el ruido de fondo, si en algún objeto se quedaba un píxel en negro conseguimos que este pasara a formar parte de la figura, por lo que con estos métodos habíamos conseguido el objetivo: eliminar el ruido de fondo y obtener el objeto de forma mas compacta. En la siguiente imagen podemos observar un ejemplo de esto:

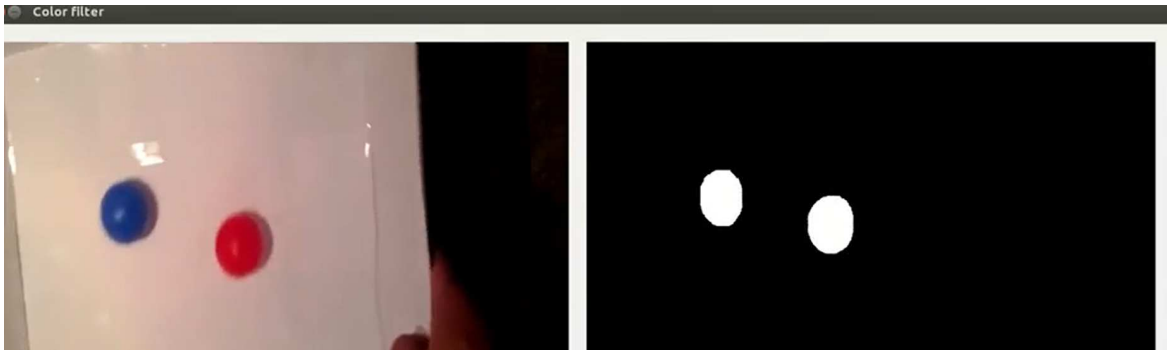


Figure 4.1: Esta imagen muestra un primer filtro de color.

A continuación pongo como ejemplo una parte de código, el cual coge una imagen, la convierte a HSV, crea un filtro de color y pasa la imagen a través de este, obteniendo una imagen resultante:

```
hsv = cv2.cvtColor(input_image, cv2.COLOR_BGR2HSV)
lower_orange = np.array([100,100,80], dtype=np.uint8)
upper_orange = np.array([150, 255,255], dtype=np.uint8)
maskOrange = cv2.inRange(hsv, lower_orange, upper_orange)
maskRGBOrange = cv2.bitwise_and(input_image,input_image, mask= maskOrange)
```

Destacar que con la última línea de este código, conseguiríamos que la imagen resultante no quedara en blanco y negro, sino que los objetos que pasan el filtro se mostrarán en su color original.

## CAPÍTULO 4. ALGORITMO

---

Con esto se obtuvo una primera buena señal, y era que sobre imagenes reales(con imperfecciones) conseguimos realizar los filtros que queriamos, por lo que el siguiente paso fue decidir la forma que tendría la baliza y comenzar a trabajar en ella y sus características. Esta baliza constaria de un cuadrado formado por 4 cuadrados de dos colores:

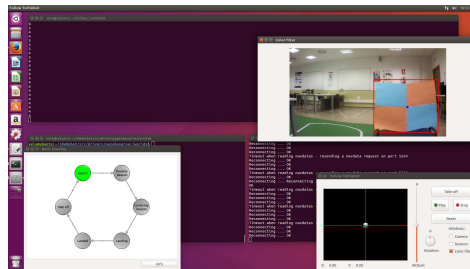


Figure 4.2: Esta imagen muestra el conjunto de herramientas utilizadas.

Con la imagen de simulador, lo que tuvimos que hacer fue cambiar los valores al filtro de color anterior, con lo que ya conseguimos obtener los colores de la baliza, pero sobre la baliza real se estuvo mas tiempo trabajando, pues al trabajar con ella en distintos entornos había muchos cambios de luminosidad, y aunque HSV minimizara este problema, según el entorno de trabajo seguía dando problemas. Al final ampliando el rango de colores, un mejor pre-procesado de esta imagen y el procesado para el algoritmo, se consiguió que esto no fuera un problema.

Pero en realidad en este punto solo teniamos el primer paso, que la baliza pasara el filtro de color, ahora teniamos que conseguir que nuestro algoritmo detectara eso como una baliza para poder posicionarse sobre ella. Para este punto, se ha pasado por distintos puntos y algoritmos, los cuales se han ido mejorando o cambiando según los fallos que se veían, hasta llegar a un algoritmo final.

Lo primero que hicimos fue sobre el simulador. Se obtenia la posición de los pixeles donde se encontraba el objeto que pasaba el filtro, y a partir de esto se calculaba su pixel central. Esto se hacia gracias a una función de OpenCV que retornaba el area

## CAPÍTULO 4. ALGORITMO

---

que no se consideraba fondo, y luego de este area calculaba su centro. Un ejemplo del codigo python que realiza esto, a partir del codigo anterior, es el siguiente:

```
momentsOrange = cv2.moments(maskOrange)
x_center = int(momentsTot['m10']/momentsTot['m00'])
y_center = int(momentsTot['m01']/momentsTot['m00'])
```

Una vez hemos obtenido el centro del objeto, hay que calcular el centro de la imagen. En este caso, para calcularlo, hemos hecho es que la primera vez que se ejecute nuestro algoritmo(teniendo en cuenta que este está en una continua ejecución) llame a una función a la que se le pasa la imagen de entrada, obtiene su tamaño y calcule a partir de este cual es el centro de la imagen. La razón de hacer esto es que tanto el dron real como el del simulador tienen dos camaras, que obtienen distintas imagenes con distintos tamaños, de esta forma se calcula el tamaño la imagen que se va a utilizar de manera automática antes de comenzar cualquier proceso. Pues bien, para hacer las pruebas iniciales, teniendo el centro de la imagen y el centro del punto al que queremos llegar, basta con hacer la resta de estos y multiplicarlos por un valor(en función de las velocidades del dron) para que este se mueva, centrando su centro y el del objeto.

Tras conseguir, con un filtro y unas funciones, que el dron pudiera situarse sobre un objeto filtrado, pasamos a mejorar la detección de la baliza. La idea principal de la detección es detectar donde esta la cruz que forman los 4 cuadrados de colores, o lo que es lo mismo, detectar la intersección de los cuadrados de la baliza, así obtendremos el punto sobre el cual situarnos para podernos centrar.

Un primer algoritmo fue realizar los dos filtros de color, uno por cada color de la baliza, obteniendo así por separado los colores verde y naranja cuando trabajabamos en el simulador. Una vez obteniamos las imagenes por separado realizamos una dilatación de ambas imágenes y luego una suma, lo que nos daba como resultado otra vez la baliza inicial pero con las intersecciones de otro color, de forma que filtrando por este nuevo color obtenido obteniamos la cruceta de la baliza. En este punto en el

## CAPÍTULO 4. ALGORITMO

---

*color filter* de la herramienta veíamos los dos filtros por separado y la intersección de ambos:



En este punto, cuando el dron detectaba un objeto en la figura de la izquierda, lo trataba como baliza, por tener los dos colores principales juntos, pero aun quedaba mucho camino por recorrer, pues no detectabamos el centro de la intersección. Por otro lado, fue en este punto donde decidimos que las ventanas que mostraba el *color filter* ya no era tan necesaria, pues habíamos obtenido lo que queríamos, y se decidió trabajar hacia una ventana que mostrara la imagen completa de lo que estaba viendo el dron y marcara de alguna forma el contorno y cruceta de dicha baliza. Para esto fueron muy útiles las funciones *findContours* y *drawContours* de OpenCV, las cuales nos permitían encontrar los bordes de los objetos de las imágenes y marcar estos. Lo que hacíamos en este proceso era obtener los bordes de las imágenes que habían pasado el filtro y después pintar sobre la imagen original. Para poder utilizar esta función necesitábamos pasar la imagen a escala de grises, como podemos ver a continuación:

```
imgray1 = cv2.cvtColor(maskRGBOrange,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray1,255,255,255)
_,contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(input_image, contours, -1, (0,255,0), 5)
```

Conseguido esto, para el simulador teníamos un algoritmo principal que hacía lo que buscábamos en situaciones ideales, pero seguía teniendo varios fallos. El



primero a arreglar era que si en lugar de 4 cuadrados la baliza solo tenia dos, al dilatar y filtrar obtendriamos de igual forma otra figura, aunque en lugar de una cruceta se tratara de una linea. La forma de enfrentarse a esto fue contando el numero de objetos que había en la imagen. Si la imagen tenía dos objetos de cada color y un objeto que fuera la intersección de los otros, se trataria de una baliza, en caso de no ser de esta forma no se trataria como tal. En la parte de visión estuvimos mucho tiempo trabajando con este algoritmo, pues para el simulador funcionaba bien. Pero en este punto, en la parte de visión, detección de colores y realización del filtro, comenzamos a trabajar con el dron real, haciendo tareas muy simples, por lo que mejorar la robustez de esto se quedo un poco de lado.

Para comenzar a probar el filtro de colores en el dron real se hizo una baliza como la que teniamos en el simulador, pero al ir a realizar las primeras pruebas estaba el problema de que en la realidad había muchos colores parecidos en las distintas direcciones, por lo tanto el dron confundia muy facil los colores. Para arreglar esto se empezo trabajando sobre un color y mejorando el algoritmo, de tal manera que el rango de colores fuera mas reducido, y aunque esto era un problema porque al trabajar con la luz real, un mismo color variaba mucho su rango segun la situación que estuviera, trabajando en HSV se consiguio, y tras esto en el pre-procesado se realizó una erosión con mas iteraciones de forma que eliminaba mejor los pixeles de fondo, pues en estas pruebas el problema era que uno de los colores de fondo era muy parecido al color de la cartulina. También añadimos que el dron solo hiciera caso a los objetos con un area mayor a determinado valor, con lo que conseguimos que si algún objeto de fondo seguía pasando el filtro lo obviara.

Despues de esto ya quedaba la ultima parte por parte del filtro de colores aunque la mas dificil: *conseguir situar una baliza en condiciones reales*. La primera complicacion de esto es el detectar dos colores muy distintos, el problema de esto era que en cualquier entorno que estuvieramos era muy probable tener de fondo un color muy parecido a cualquiera de los dos, pero como anteriormente, fue reforzar el filtro

y pre-procesado para conseguir esto. Una vez conseguimos las pruebas eran sencillas: hacer despegar al dron y poner la baliza delante. Una vez que movieramos la baliza el dron debía moverse en el mismo sentido, prueba que, aunque costosa, se realizo con exito. Destacar que también en este momento la imagen que mostraba *color filter* se volvio a modificar, pues se mostraba la imagen real y sobre ella se pintaba, si era posible baliza, los bordes y la cruceta en verde. En el momento que se trataba de la baliza real, los bordes se pintaban en rojo (valor RGB 255,0,0) y la cruceta en verde (valor RGB 0,255,0). De esta forma era muy facil saber, con solo visualizar la imagen, en que punto se encontraba el algoritmo y si actuaba correctamente.

Ya con un algoritmo de detección con el que se podía trabajar correctamente, todavía le faltaba algo de robustez, pues si situabamos un objeto del mismo color que alguno de los colres de la baliza lo detectaba como un objeto mas, lo que llevaba a que el numero de objetos de un color fuera mayor, y daba problemas como el que el número de objetos de un color fuera mayor, el area era mayor que el deseado y en caso de calcular el centro del area no lo situaba donde debía. Por lo tanto, se llego a una solución que, aunque tardaba mas en procesar la imagen, es mas robusta. El proceso es el siguiente:

1. Como anteriormente, pasar los filtros de color y obtener solo los objetos de los colores deseados, pasando el resto a ser píxeles de fondo. De aqui obtenemos dos imágenes, una por color.
2. De cada imagen, obtenemos el número de objetos, sus areas y sus contornos. Vamos recorriendo los objetos de las imagenes, en caso de tener un area mayor a un valor determinado, se crea una imagen negra del mismo tamaño que la imagen de entrada, y se pinta sobre esta el contorno del objeto con un valor determinado, ponamos como ejemplo que cada contorno tiene un valor RGB (0,1,0). De forma que, si por ejemplo, el numero de objetos tras pasar los dos filtros era 5, obtendremos 5 imagenes.

3. Por cada imagen obtenida, dilatamos lo obtenido (por lo que el borde se ensancha) y lo sumamos a una imagen final, de forma que en cada imagen, donde se sitúe el contorno se sumará a cada píxel un valor  $\text{RGB}(0,1,0)$ . De forma que el punto que sea la intersección de dos objetos, el valor será  $(0,2,0)$  y donde sea la intersección de 4 objetos el valor será  $(0,4,0)$ . El punto donde interseccionen 4 objetos será la cruceta (ejemplo de la suma de imágenes en figura 4).
4. Una vez tenemos esta imagen final, le podemos aplicar un filtro RGB que permita pasar los píxeles con valor  $(0,4,0)$ , de forma que de la cruceta solo obtendremos el punto donde interseccionan los 4 cuadrados.
5. Ya con la imagen que solo muestra los píxeles de intersección, podemos utilizar la función `drawcontours`, que a parte de permitirnos marcar el centro sobre la imagen original, nos retorna el valor de los píxeles que pinta, de donde obteniendo su centro, podemos obtener con exactitud los valores X e Y donde se sitúa la cruceta.
6. Para finalizar este proceso, se realizó de nuevo un cambio en *color filter*, en el cual con la función `rectangle` de OpenCV marcamos la cruceta de la baliza (como se puede ver en la figura dos de la página 17) y con un cuadrado rojo los objetos que son posibles balizas, marcando de esta forma la baliza real.

A continuación inserto un fragmento de código, el cual a partir de una imagen que ha pasado un filtro, pinta los contornos de cada objeto, cada uno sobre una nueva imagen negra, y las guarda en un array. Tras esto dilata y suma las imágenes obteniendo la imagen final.

```
f = []  
i=0  
imggray2 = cv2.cvtColor(maskRGBOrange,cv2.COLOR_BGR2GRAY)
```

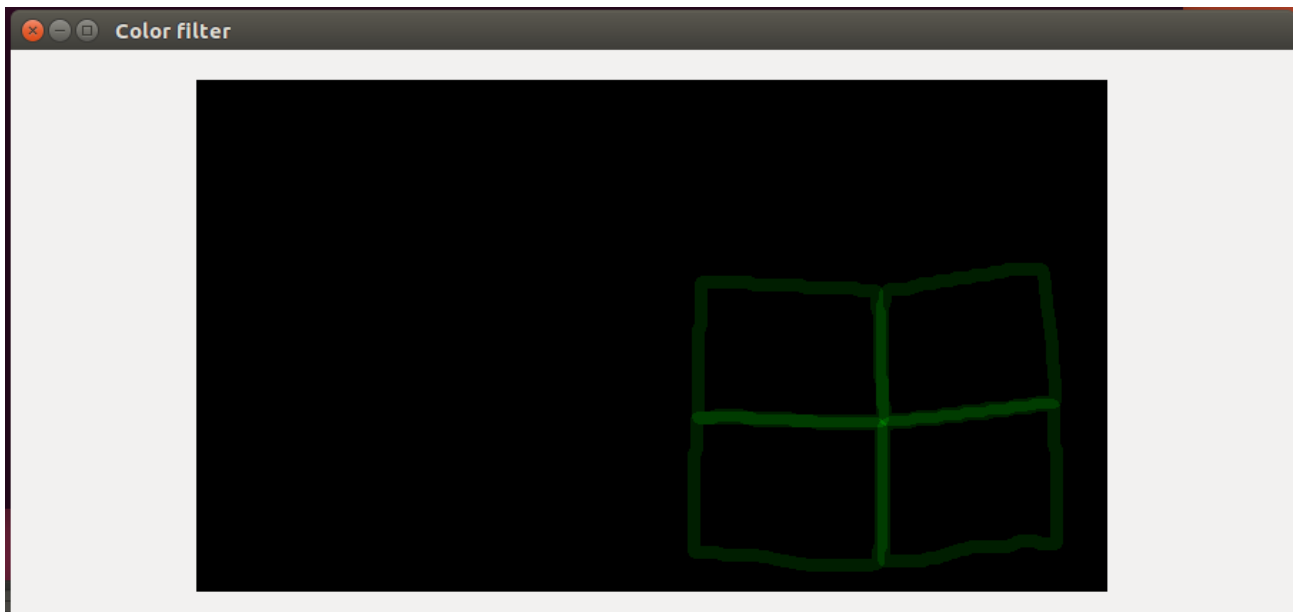


Figure 4.3: Imagen que muestra la suma de las diferentes imagenes creadas por cada objeto.

```
ret,thresh = cv2.threshold(imgray2,255,255,255)
_,contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
areas = [cv2.contourArea(c) for c in contours]
for extension in areas:
    if extension > 100:
        img = np.zeros((y_img*2,x_img*2,3), np.uint8)
        actual = contours[i]
        approx = cv2.approxPolyDP(actual,0.05*cv2.arcLength(actual,True),True)
        cv2.drawContours(img,[actual],0,(0,30,0),12)
        f.append(img)
        i=i+1

kernel = np.ones((3,3),np.uint8)
```

```
if(len(f)>0):
    f[0] = cv2.dilate(f[0],kernel,iterations = 4)
    show_image2=f[0]
    for k in range(len(f)-1):
        f[k+1] = cv2.dilate(f[k+1],kernel,iterations = 4)
        show_image2=show_image2+f[k+1]
```

Por otro lado, voy a mostrar un fragmento de código en el cual, a partir de la imagen que muestra el borde de la baliza, la cruceta y el centro en determinados valores, obtengo los pixeles sobre los que se centra la cruceta. La razón de que se inicien a un valor negativo es para que al retornar estos valores cuando se llama a la función, el algoritmo sepa que no se ha detectado la intersección entre los cuatro cuadrados de la baliza.

```
lower_green = np.array([0,80,0], dtype=np.uint8)
upper_green = np.array([0, 255,0], dtype=np.uint8)
maskSHI = cv2.inRange(show_image2, lower_green, upper_green)
show_image2 = cv2.bitwise_and(show_image2,show_image2, mask= maskSHI)

compare_image = np.zeros((y_img*2,x_img*2,3), np.uint8)
diff_total = cv2.absdiff(compare_image, show_image2)

imagen_gris = cv2.cvtColor(diff_total, cv2.COLOR_BGR2GRAY)
_,contours,_ = cv2.findContours(imagen_gris,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

positionX=-1
positionY=-1
for c in contours:
```

```
if(cv2.contourArea(c) >= 0):  
    posicion_x,posicion_y,ancho,alto = cv2.boundingRect(c)  
    cv2.rectangle(show_image,(posicion_x,posicion_y),(posicion_x+ancho,posicion_y+  
    , (0,0,255),2)  
    positionX= (posicion_x+posicion_x+ancho)/2  
    positionY= (posicion_y+posicion_y+ancho)/2
```

### 4.3 Control.

Una vez terminado el filtro de color, hablemos del algoritmo de movimiento del dron. Las primeras pruebas sobre simulador se realizaban de forma simple: El dron despegaba, y mientras no detectara baliza sobre la que centrarse, subia dando vueltas sobre si mismo, en el momento que encontraba la baliza obtenía el centro de esta e intentaba que coincidiera con el centro de la imagen. Tras esto se probó que la baliza se desplazara y que el dron fuera capaz de seguirla. Prueba que también funciona sin problemas. Al depender la velocidad de la resta de los centros, provocaba que al estar mas lejos la velocidad de movimiento fuera mayor y fuera disminuyendo conforme se centraba, evitando de esta forma que el dron frenara de forma brusca y disminuir los balanceos y turbulencias. Aun así, se detectaban ciertas imperfecciones debido a que se trataba de un controlador proporcional, el cual funcionaba bien pero no lo suficiente, por lo que se decidió añadir una componente derivativa, con el objetivo de mantener el error al minimo, evitando que este se incremente y por lo tanto el dron sufra cada vez mas oscilaciones al situarse sobre un punto. Este control se basa en derivar el error con respecto al tiempo y multiplicarlo por una constante. Dado que nuestro algoritmo ejecuta una vez cada cierto tiempo y no esta continuamente pasando por este punto, podemos considerar que se trata de un sistema en tiempo discreto, y por tanto en lugar de trabajar con derivadas trabajaremos con sumatorios. De esta forma, para añadir un control derivativo realizabamos una operación que depende de la velocidad anterior y

## CAPÍTULO 4. ALGORITMO

---

la que tenemos ahora:

$$v_{derivativa} = 1 - (v_{anterior} - v_{nueva})/50$$

En nuestro algoritmo, en caso de que la operación de un valor menor a 0.1 lo igualamos a este, con lo que conseguimos que nunca sea un valor muy cercano a 0 y por lo tanto no se quede en el sitio. Este resultado lo multiplicamos a la velocidad final, y lo que conseguimos es que si el valor entre dos velocidades continuas es muy alto este se atenúe de forma que el dron no cambie mucho y vaya oscilando, sino que lleve una velocidad mas continua, de cara a acelerar o frenar su velocidad.

Por otro lado, el dron tiene un algoritmo de busqueda, en el que se pretende que se mueva realizando una espiral continua, haciendo así que recorra toda la zona de su alrededor. Para ello se le envía la información de dos velocidades que fueran variando con el paso de las iteraciones, una que dependía de la rotación sobre si mismo, y otra velocidad que se le pasaba para que hiciera en su eje X hacia delante. De esta forma, con el paso de las iteraciones disminuía la velocidad de rotación, al mismo tiempo que aumentaba la velocidad hacia delante, con lo que conseguimos un algoritmo de busqueda en espiral, pudiendo recorrer de esta forma un determinado area con facilidad. Un ejemplo de esto lo podemos ver en el siguiente fragmento de código.

```
self.cmdvel.sendCMDVel(1.8+wSearch,0,0,0,0,1.5 - wSearch)
numVuelta=numVuelta+1
if(numVuelta==100):
    timerW=timerW+(timerW/8)
    numVuelta=0
    if(wSearch<1):
        wSearch=wSearch+0.2
```

Por ultimo, otra herramienta que se utilizo fue el diagrama de estados gracias a la herramienta visualStates. Esta parte es sencilla, al arrancar la aplicación se pueden

## CAPÍTULO 4. ALGORITMO

---

crear distintos estados haciendo que unos apunten a otros, de tal forma que puedes ver el punto en el que se encuentra el algoritmo. Cada vez que pasas por una parte del código, indica en que punto se encuentra y este se marca con color verde sobre el diagrama. En un principio este diagrama constaba de tres estados, despegue, búsqueda y aterrizaje(que como veremos mas adelante, son las partes principales del algoritmo), aunque para tener mas datos modificamos para tener despegue, búsqueda, posible baliza, centrando sobre la baliza, aterrizando y aterrizado. Este nos fue de gran utilidad, pues cuando en un inicio planteamos los posibles estados que había y sus distintas transiciones vimos que había multitud de posibilidades, dependiendo de que parte viera de la baliza, si solo veía un color y al final encontraba dos o simplemente era otro objeto que coincidía, si por alguna casualidad perdía la baliza y tenía que comenzar el algoritmo desde el principio, o si todo iba correctamente. Gracias a que se mostraba este diagrama, era todo mucho mas sencillo.

Para poder añadirlo lo unico que se tuvo que hacer es añadir dos paquetes del GUI de JdeRobot, uno que permitía abrir otra ventana y otro que permitía añadir estados y transiciones entre ellos. Para marcarlo, siendo el numero que aparece el numero del estado que queremos marcar, valía con la siguiente linea de código:

```
self.machine.setStateActive(2, True)
```

Visualmente, durante la ejecución obteníamos esto:

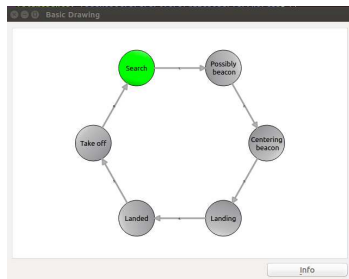


Figure 4.4: Imagen que muestra el diagrama de estados.



## 4.4 Funcionamiento del algoritmo.

En realidad, todo lo contado del algoritmo sirve para satisfacer estas tres partes.

**El despegue** del dron. Para estas pruebas se ha estimado que el despegue constara de entre 5 y 10 segundos dependiendo el entorno en el que nos encontremos. Este varia principalmente por el espacio disponible, pues esta parte lo que pretende es levantarse del suelo y alcanzar cierta altura. Aunque es la parte mas sencilla, en un principio pensamos que todo ocurriria en perfecta forma, sin embargo al comenzar a trabajar con el ArDrone nos dimos cuenta de que bien por las corrientes internas que se forma en espacios cerrados, o por la inestabilidad de este simplemente por el paso del tiempo y que su estructura ya no esta en como en un inicio, una vez en el aire no mantenía la posición, sino que se desplazaba cuando se le enviaban instrucciones de quedarse en el sitio(en mi caso,con el dron que trabajaba una vez se levantaba comenzaba a ir hacia atrás). Debido a esto, decidimos que el despegue se controlara también mediante visión. Lo que pretendíamos con esto es que una vez el dron estuviera en el aire, detectara una baliza de color naranja, calculara su centro e intentara situarse sobre este, asegurandonos así realmente que el dron permanecia en el sitio. Trabajando en esta idea nos dimos cuenta que el dron tenía un par de segundos en el despegue en los que trabajaba de forma automática y no es capaz de realizar las acciones que se le estan enviando, lo que lleva a estar los dos segundos iniciales sin control sobre este. Durante este periodo se marca el estado "Take off" en el diagrama de estados.

Una vez tenemos el dron en el aire llegamos a la parte mas compleja, **la busqueda** de la baliza. En este momento se marca el estado "Search" en el diagrama de estados, y el dron comienza a moverse en forma de espiral mientras busca algún objeto que tenga los colores de la baliza. En el momento que encuentra uno de los dos colores marca el estado "Possibly beacon" e intenta centrarse sobre el objeto, esperando que se trate de una baliza. En caso de no tratarse de la baliza, el dron se aparta de esta y continua con su algoritmo de busqueda, pero en caso de serlo

## CAPÍTULO 4. ALGORITMO

---

marca sobre el diagrama de estados "Centering beacon" haciendo que coincidan sus centros. Aquí ya no se fija en el área del objeto como en el caso anterior, para centrarse sobre el centro de este, sino que al haber obtenido ya donde está la cruceta, tiene sus valores desde un principio, por lo que es sobre estas coordenadas sobre las que tiene que situarse. Destacar que pueden darse factores externos por los cuales el dron pierda parte de la baliza y no la detecte como tal, pero si siga detectando objetos, en este caso tratará a tales objetos como la baliza e intentará situarse de nuevo sobre ellos durante un periodo de tiempo. En caso de tratarse de algún otro factor y que ya no exista baliza en este punto continuará buscando. Una vez el centro de la imagen y la cruceta están a una distancia menor de un determinado número de píxeles, aparte de centrarse comienza su descenso, hasta que la baliza en la imagen tiene un determinado área en el que se considera que ya está suficientemente cerca y deja de descender.

Es en este momento cuando comienza la parte del **aterrizaje**, esto se debe a que el aterrizaje en sí también es un proceso que se realiza un poco a ciegas, pues en el momento que envías la instrucción de "land" al dron este comienza a descender hasta que llega al suelo y para sus motores. Este momento se produce cuando el dron está prácticamente centrado sobre la baliza y a una distancia cercana, que nos podamos asegurar que en el periodo de aterrizaje no se va cruzar ningún obstáculo ni se van a producir accidentes. Se marca en el diagrama de estados "Landing" y el dron comienza a descender en vertical hasta encontrar el sitio donde posarse. Una vez aterrizado para sus motores y se marca el estado "Landed", habiendo finalizado de esta forma el algoritmo.

# Chapter 5

## Experimentos.

Durante este capítulo voy a contar las distintas pruebas que se han realizado durante todo el proceso. Cabe destacar que en un principio las pruebas fueron muy variadas y no todas se han utilizado en la idea final. Esto se debe a que en un principio teníamos diversas opciones e ideas, y fuimos investigando en ellas, pero por diversos factores no se llevaron a cabo.

En un principio la idea era hacer funcionar todo sobre el **3DR solo drone**, debido a sus muy buenas características anteriormente explicadas. Para ello estuvimos, junto a otro compañero, un tiempo estudiando su funcionamiento y como podíamos acoplar este a la tecnología JdeRobot. Fue un proceso costoso y donde pudimos detectar sus virtudes y defectos, pues tras probar el dron, manejándolo con su mando, veíamos que sus increíbles características como podían ser su potencia a la hora de los movimientos y la gran estabilidad que tenía al realizar movimientos bruscos con él. Tras esto, al comenzar a trabajar con las herramientas nos dimos cuenta del primer problema, el que levantaba la red WiFi no era el dron sino el mando que venía con él, el dron levantaba la red de radiofrecuencia y con esta se comunicaba por el mando, lo que conllevaba que la navegación del dron para ejecutar un algoritmo dependía de la distancia de este al mando. Buscando encontramos que la empresa estaba trabajando

en una solución a esto, pero que tardaría meses en llegar. Por otro lado se juntaba que el dron no tenía una camara incorporada, ya que funcionaba con una GoPro, para la cual JdeRobot no tenia soporte. A todo esto habia que buscarle una solución, así que se penso en añadir una camara para la visión de otra forma. Gracias a la potencia de este dron se le podian añadir materiales sin que esto le impidiera volar, como era una *intel compute stick* y una *camara externa*. Realizamos pruebas con la intel computer stick y el ArDrone, el cual funcionaba sin problemas, y mas adelante probando la comunicacion de la intel con este dron levantando el servidor y por otro lado el ordenador comunicando con la intel y enviandole las instrucciones para que las realizara el dron, prueba que se realizo con exito. Al igual que funciono esto, también funcionaron las pruebas con el dron 3DR Solo, pudiendo manejarlo con las herramientas JdeRobot y viendo sus datos cuando este realizaba movimientos. Tras esto, como la idea del algoritmo era que funcionara también en interiores, por ejemplo en grandes naves industriales para el control y desplazamiento de mercancia, nos pusimos con esta prueba, lo que nos hizo darnos cuenta de que debido a su GPS, hasta que no encontraba una señal lo suficientemente fuerte no te dejaba despegar, protocolo de seguridad por si habia algún problema que pudiera utilizar su modo “return home“. Pero utilizando el protocolo MAVLink desde el ordenador habí un modo que te permitia evitar las restricciones de seguridad, haciendo asi que el dron despegara y pudiera ser utilizado en cualquier lugar. Fue al probar esto cuando la potencia del dron nos jugó una mala pasada, pues genero una cantidad de corrientes internas de aire que, en el momento de su despegue, en el cual hay unos segundos donde no se tiene control sobre el, no lo hiciera totalmente en vertical sino con una desviación hacia su izquierda, lo que llevo a chocar con distintos obstáculos. Por ultimo e intentando utilizar varios materiales juntos, probamos a unir la intel computer stick al ArDrone 2.0, para ver si tenía suficiente fuerza como para volar con el, pero el ArDrone no consiguió levantarse del suelo.

Tras la realización de las pruebas y alguna mas que surgió a raíz de estas, nos

dimos cuenta de que habíamos obtenido grandes avances que sin haber realizado un trabajo conjunto podrían habernos llevado mucho mas tiempo, pues esto nos aportó muchos datos que nos servirían mas tarde, a parte de aprender el funcionamiento de estos drones, los materiales que los componían y como realizar la comunicación dron-ordenador para el envio de información y recibo de datos. Por todo esto y por los frentes que se iban abriendo para trabajar sobre el dron, fue cuando dividimos un poco mas nuestras tareas, ya no investigando un conjunto de cosas sino dividiendo las tareas y semanalmente poniendo en comun los avances que habíamos realizado, para ver que íbamos encaminados hacia el objetivo. En este punto fue cuando quedo marcada mi tarea principal para poder realizar este proyecto. **La detección de balizas y algoritmo de movimiento.**

Ya se ha explicado antes como fue el desarrollo del algoritmo, pero la verdad es que las distintas pruebas realizadas son las que de verdad permitieron su desarrollo.

Se comenzo trabajando con las camaras del ArDrone server en un espacio pequeño y cerrado, donde se consiguió un filtro para los dos colores de la baliza. Tras esto se quiso probar el dron en un espacio amplio, debido a que en espacios pequeños, al despegar el dron no se mantenía en el eje Z sino que lo hacía en diagonal, por lo que se chocaba con obstaculos como podían ser las mesas que había, y ya se aprovechó para que así pudiera realizar todo el algoritmo. Pero a la hora de detectar los colores se notaba la luz que entraba en todos los momentos y como iba oscureciendo con el atardecer, lo que llevo a que la prueba con el dron real no se hiciera sobre dos colores sino centrandonos solo en el naranja. Una vez conseguido el filtro, el fondo del pabellón en el que estábamos, la pared era de un marron muy claro que se podía confundir con la baliza, por lo que se fijo mas la idea de obviar los elementos pequeños, y realizar una mayor erosión, y entonces salio una primera prueba de gran valor, cuando nos situabamos delante del dron con la baliza , este aterrizaba:

Despues de esta prueba en el mismo lugar, decidimos probar el algoritmo de busqueda, sin encontrar la baliza, solo ver que una vez que despegaba se desplazaba



Figure 5.1: Aterrizaje

realizando una espiral, prueba que quedo grabada y también se realizo con éxito.

Ya teniendo el algoritmo de busqueda y un primer filtro de color, las pruebas se centraron en que el dron consiguiera centrarse sobre una baliza tras despegar. En un principio fue complicado, pues el dron que se tenía para realizar las pruebas al despegar, en los dos segundos que no teniamos control sobre el, en lugar de subir verticalmente lo hacía en diagonal hacia atras, lo que nos llevo a comprobar que distancia recorria en este tiempo para poder jugar con ella, colocando el dron unos metros delante de la baliza, para que en el momento que se tomaba el control se situara sobre esta. Los primeros controles no eran muy fluidos, pues el algoritmo que había para detectar el color e indicar sobre el GUI eran un poco pesados, y añadiendo a eso que solo se tenía un control progresivo, oscilaba mucho, llegando un momento que perdía la baliza. Debido a esto se mejoro sobre todo el control, aunque también el algoritmo, utilizando mas funciones de OpenCV, por ejemplo. Con todo esto, el dron se mantenía sobre la baliza aunque continuaba oscilando mas de lo que debiera. Poco a poco se fue ajustando para que la oscilacion fuera mínima.

Como con un color ya teniamos grandes avances, se volvio a trabajar sobre la baliza real, lo que nos llevo a ajustar de nuevo un poco la parte del filtro de color y la información que se enviaba debido a este, pues en cada prueba prácticamente habia que modificar parte del pre-procesado debido a las condiciones del lugar. Pero una vez se vio que era algo robusto se intento trabajar de nuevo en lugares de menor

tamaño, y destacar que las pruebas salieron bien, pues habiendo calculado la distancia que recorría en diagonal, al contar con esta distancia en el despegue, cuando se tomaba control del dron, este veía la baliza que se le tenía puesta, por lo tanto ejecutaba las instrucciones que se le enviaban y no se desviaba mas de lo debido. El problema de comenzar a trabajar en este entorno de trabajo fue que el suelo era de un color muy parecido al que teníamos en la baliza, y por tanto se tuvieron que ajustar de nuevo los filtros de color. Decir que de nuevo se volvian a producir mas oscilaciones de las esperadas , así que para evitar esto finalmente se añadió lo llamado banda muerta, de forma que si la desviación es minima no intente centrarse sino que se quede en el sitio, y si la desviación se encuentra dentro de unos limites se corrija muy brevemente, consiguiendo asi que el dron este situado sobre cierto area, que era lo que se pretendia, aunque no sea exactamente el centro.

Como se puede observar, todas estas son las pruebas realizadas con materiales reales, pero destacar que todas estas, antes de probarlas así fueron realizadas en el simulador, ya que era lo mas comodo para trabajar calculando areas, centros y la utilización de operadores morfológicos. Contar las pruebas realizadas sobre este sería repetir en cierta forma el proceso, pero si me gustaría destacar la primera prueba que se hizo con el controlador PD, pues en videos vistos a posteriori se puede comprobar la diferencia de oscilación, pues en el real cuando oscilaba mucho el dron, perdía la baliza y se le mandaba la instrucción de quedarse en el sitio para que pudiera aterrizar, en el simulador se veía como giraba bruscamente, perdía la baliza y volvía a su posición, donde volvía a ver la baliza lo que le llevaba a girar bruscamente de nuevo, así durante varias iteraciones hasta que se centraba y se paraba ese movimiento brusco. Sin embargo cuando se añadió este controlador se veía como por el cambio de velocidad tenía un punto de frenado debido a la diferencia entre velocidades y poco a poco volvía a acelerar en una dirección o frenar, según lo requerido en ese momento.

# Chapter 6

## Conclusiones

Para finalizar, hay que realizar una evaluación del trabajo realizado, lo aprendido durante este periodo, los objetivos cumplidos y a los que no hemos llegado de la forma deseada. En un primer aspecto, podemos decir que el trabajo ha sido satisfactorio, en primer lugar porque se ha conseguido un algoritmo que desarrolle un despegue-busqueda-aterrizaje como se deseaba en un primer momento, además que para llegar a ello se han superado distintos retos que han ido surgiendo a lo largo de su desarrollo, pero profundizando un poco más en todo esto:

- En primer lugar comentar todo lo que ha supuesto trabajar con un dron. En realidad creo que esto me ha aportado bastante para trabajos futuros, pues el trabajar con un robot real te lleva a darte cuenta de como puede cambiar algo en función de factores externos, como pueden ser viento, luz o lluvia, o incluso internos como el desgaste del robot, ya que no son todas condiciones ideales en las que superando ciertos tests sabes que el programa va a funcionar, sino que hay que hacer muchos test y en condiciones muy distintas. Además, también cuando se da un inesperado problema y tienes que dejar de trabajar en la línea que lo estabas haciendo durante un tiempo determinado, uno se da cuenta que el trabajar en la realidad y no con simuladores puede conllevar esto, y tener



que cambiar durante este tiempo la linea de trabajo sin tener claro como se van a juntar luego ambos caminos. Todo esto creo que son factores negativos que aportan datos positivos que ayudan a madurar en la forma de trabajo.

- En segundo lugar, entrando en el objetivo, yo creo, mas complicado de este trabajo, es el conseguir un filtro de color. Al final se ha conseguido un filtro deseado, pero tras muchos cambios. Esta parte ha llevado un gran trabajo, pues he tenido que aprender a utilizar herramientas nuevas y consultar muchas dudas que iban surgiendo, trabajar con distintos espacios de color y en diferentes condiciones de luminosidad. Pero al final este objetivo se ha conseguido, algo que ha resultado de gran satisfacción puesto que se empezo prácticamente al principio y no ha dejado de sufrir cambios hasta el ultimo momento.
- Ya por ultimo, hablando de la parte de control del dron, decir que en el dron real ha sido mucho mas costosa de lo esperado, es una parte que sobre el simulador ha funcionado sin problemas cada vez que se añadía algo, pero al querer trabajar sobre el dron real era muy complicado y cualquier cambio de un valor pequeño suponía gran cambio en la realidad. Es este punto también uno se da cuenta de lo difícil que es dejar de lado las condiciones ideales. Aun así ver el algoritmo de busqueda y el aterrizaje cuando encontraba la baliza ha sido muy fructífero.

También, viendo la wiki realizada durante el proyecto, en la que se han ido documentando las cosas importantes que se iban haciendo, y algunos datos que tenía guardados de diversas pruebas realizadas, puedo ver grandes avances en lo realizado, y darle un gran uso a las nuevas tecnologías.

Es importante destacar que estos campo de la robótica y la visión se esta produciendo un importante crecimiento, y añadiendo lo aprendido en el trabajo y como se puede trabajar en el creo que sería interesante continuar la linea de este proyecto para continuar con el desarrollo y aprendizaje de estas técnicas.