



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN

GRADO EN INGENIERÍA TELEMÁTICA

**TRABAJO FIN DE GRADO**

**DESPEGUE, NAVEGACIÓN Y  
ATERRIZAJE VISUALES DE UN DRONE  
USANDO JDEROBOT**

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 los UAV tienen mucha historia, 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.

Durante este proyecto se ha diseñado y programado un algoritmo con la finalidad de que el drone navegue de forma autónoma, desde una baliza inicial hasta otra baliza final, de posición desconocida, y no necesite a ninguna persona diciéndole lo que tiene que hacer en cada momento o controlándole. Para conseguirlo se han tenido en cuenta los distintos estados en los que se puede encontrar el drone (despegando, búsqueda o aterrizaje). También se ha desarrollado la percepción visual de las balizas, utilizando filtros de color y otras técnicas, como operadores morfológicos, permite detectar si un objeto es o no el deseado y de esta forma poder dirigirnos a él. Las balizas se han elegido para que sea difícil de confundir con los demás, y facilitar un correcto funcionamiento. Para toda esta programación nos hemos apoyado en el entorno JdeRobot, utilizando OpenCV para el procesado de imágenes.

El algoritmo programado se ha validado experimentalmente, tanto en un drone simulado (utilizando el simulador Gazebo), como en un drone real (utilizando el Ardrone2 de Parrot). Se ha conseguido un comportamiento satisfactorio funcionando en tiempo real. Todo el software desarrollado y los vídeos de las pruebas están accesibles públicamente.

# Contents

<b>Índice de figuras</b>	<b>VI</b>
<b>1 Introducción</b>	<b>1</b>
1.1 Historia de los drones . . . . .	1
1.2 Aplicaciones actuales . . . . .	3
1.2.1 Medios Audiovisuales . . . . .	3
1.2.2 Seguridad . . . . .	4
1.2.3 Sector agrícola . . . . .	5
1.2.4 Inspección . . . . .	5
1.2.5 Emergencias . . . . .	6
1.2.6 Prototipos de investigación . . . . .	7
1.3 Hardware de drones . . . . .	9
1.4 Software de drones . . . . .	15
1.5 Robótica aérea en RoboticsLab de la URJC . . . . .	17
<b>2 Objetivos</b>	<b>21</b>
2.1 Objetivo principal . . . . .	21
2.2 Requisitos . . . . .	22
2.3 Metodología . . . . .	22
2.4 Plan de trabajo . . . . .	23
<b>3 Infraestructura utilizada</b>	<b>25</b>
3.1 Parrot AR.Drone 2.0 . . . . .	25
3.2 Simulador Gazebo . . . . .	26
3.3 Entorno JdeRobot . . . . .	27

3.3.1	Driver ArDroneServer . . . . .	27
3.3.2	Plugin modelo de cuadricoptero Gazebo . . . . .	27
3.3.3	Herramienta UAVviewer . . . . .	28
3.3.4	Herramienta ColorTuner . . . . .	28
3.3.5	Práctica choca-gira con autómatas. . . . .	29
3.3.6	Práctica follow_turtlebot. . . . .	30
3.4	Biblioteca OpenCV . . . . .	30
<b>4</b>	<b>Algoritmo de Navegación autónoma</b>	<b>32</b>
4.1	Diseño . . . . .	32
4.2	Percepción . . . . .	35
4.2.1	Pre-procesado . . . . .	36
4.2.2	Detección de la baliza . . . . .	38
4.3	Control . . . . .	41
4.3.1	Despegue . . . . .	42
4.3.2	Búsqueda . . . . .	42
4.3.3	Aterrizaje . . . . .	46
4.4	Arquitectura software de la implementación . . . . .	47
<b>5</b>	<b>Experimentos</b>	<b>50</b>
5.1	Pruebas de percepción de la baliza . . . . .	51
5.2	Experimentos de despegue . . . . .	52
5.3	Experimentos de búsqueda . . . . .	54
5.4	Experimentos de aterrizaje . . . . .	56
5.5	Ejecución típica del algoritmo completo . . . . .	58
<b>6</b>	<b>Conclusiones</b>	<b>61</b>
6.1	Líneas futuras . . . . .	63



# List of Figures

1.1	Drone grabando para la publicidad de una marca de coches. . . . .	4
1.2	Empresa Prevent Security Sistems utiliza drones para videovigilancia en grandes superficies . . . . .	4
1.3	Empresa Novadrone utiliza drones para la gestión de las explotaciones agrícolas . . . . .	5
1.4	Desarrollan en la universidad de Holanda un drone que lleva incorporado un desfibrilador . . . . .	7
1.5	Drone detectando código RFID. . . . .	9
1.6	Diferentes movimientos que puede realizar un drone. . . . .	10
1.7	Distintas partes del drone. . . . .	14
1.8	TFG Alberto Martín. . . . .	18
1.9	TFG Arturo Velez. . . . .	18
1.10	TFG Manuel Zafra. . . . .	19
1.11	TFG Daniel Yagüe. . . . .	19
1.12	TFG Jorge Cano. . . . .	20
2.1	Método de desarrollo en espiral . . . . .	23
3.1	Mundo Gazebo. . . . .	27
3.2	UAV viewer . . . . .	28
3.3	Diagrama de estados. . . . .	29
3.4	Follow Turtlebot. . . . .	30
3.5	OpenCV . . . . .	31
4.1	Esquema que muestra la adquisición, proceso y envío de órdenes . . . . .	34

4.2	Baliza utilizada para el drone real. . . . .	34
4.3	Esquema de percepcion del algoritmo . . . . .	35
4.4	Ejemplo de un filtro de color y uso de operadores morfológicos para eliminar ruido. . . . .	37
4.5	Procesamiento de imagen de la baliza . . . . .	41
4.6	Esquema de control del algoritmo. . . . .	42
4.7	Diagrama de estados utilizado para el algoritmo. . . . .	43
4.8	Esquema de control durante el algoritmo de búsqueda . . . . .	46
4.9	Interfaces que se abren al iniciar la aplicación. . . . .	49
5.1	Drone detectando objeto en escenario simulado. . . . .	51
5.2	Detectando baliza real. . . . .	52
5.3	Despegue sobre la baliza del coche. . . . .	53
5.4	Experimento de despegue . . . . .	53
5.5	Búsqueda en espiral en escenario simulado . . . . .	55
5.6	Búsqueda en el escenario real . . . . .	56
5.7	Experimento de aterrizaje en simulación . . . . .	57
5.8	Experimento de aterrizaje en el drone real . . . . .	58
5.9	Algoritmo completo de navegación en escenario simulado. . . . .	59
5.10	Algoritmo completo sobre el drone real. . . . .	60

# Chapter 1

## Introducción

Cada vez es más común el uso de drones para labores muy diversas, como puede ser grabar un plano para una película o mantener vigilado un lugar sobrevolando estas zonas. Una parte en la que se dan grandes avances es la robótica aérea, y una funcionalidad concreta es la navegación autónoma de éstos, conseguir que realicen ciertas tareas sin que haya nadie controlando su ruta. Para ello hay que contar con los distintos sensores que se le pueden añadir a un drone y la forma de utilizar éstos en beneficio propio para conseguir dicha navegación.

En los siguientes párrafos se va a realizar una breve introducción sobre estos aparatos, la historia que tienen, los distintos usos que hay para ellos actualmente, su hardware y software disponible para ellos. También se describirán varios proyectos de RoboticsLab-URJC que forman parte del contexto cercano de este Trabajo de Fin de Grado.

### 1.1 Historia de los drones

Los drones son conocidos como UAV (vehículos aéreos no tripulados). El primer registro de UAV fue un globo aerostático en un entorno militar en el año 1849, que

## CAPÍTULO 1. INTRODUCCIÓN

---

se podía utilizar para sobrevolar una zona y lanzar bombas desde cierta altura sin necesidad de que hubiera ninguna persona en éste y, por tanto, sin arriesgar una vida. Este UAV es muy distinto a lo que vino después, principalmente porque el motor de éste se trata de una bolsa que tiene un gas más 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.

Ya en la primera guerra mundial se comenzaron a utilizar para sobrevolar las áreas enemigas y hacer fotos para así tener un control de sus movimientos (el introducir una cámara en un UAV es algo que se hizo desde los primeros momentos). Estos vehículos eran aviones tripulados por radiofrecuencia, por lo que se dio un gran salto con respecto al anterior, pues era mucho más fácil su control, por lo que podían manejar su trayectoria con mucha más facilidad.

También durante la primera guerra, pero más desarrollado para la segunda guerra mundial, se le dio 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 en los inicios tenían sólo fines militares y que su desarrollo era exclusivamente para ello. Tras la segunda guerra mundial, el avance sobre estos frenó en gran medida y 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.

Fue alrededor de 1980 cuando se vio que la tecnología y el software de los UAV eran de gran fiabilidad y se les podía asignar a estas tareas de mayor responsabilidad para no poner en riesgo la vida de los pilotos. 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 podían realizar por ser demasiado bruscos para el cuerpo humano, ahora podían hacerlos con la brusquedad que permitiera el sistema.

## CAPÍTULO 1. INTRODUCCIÓN

---

Ya en la década de los 90 se da un avance muy importante, con el desarrollo del sistema GPS para el desplazamiento de estos vehículos. Esto permitía no depender de la radiofrecuencia, ya que con ésta se tiene un límite en distancia y no revisar los datos para ver en todo momento su situación y teledirigir la trayectoria. Con este sistema se traza una ruta al inicio y el UAV puede trabajar de forma autónoma.

Cabe destacar el gran avance que ha sufrido la robótica aérea e los últimos años. En torno al año 2000 y en adelante, se ha profundizado en el uso civil de los drones y no tanto militar. Por un lado en la parte aeroespacial, cada vehículo innovador mejora con creces al anterior debido a diseño, estructura o materiales, que permiten mayor velocidad y resistencia. Y por parte de la robótica ocurre lo mismo, está 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 más 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 Aplicaciones actuales

En esta sección se barre de manera breve las aplicaciones y distintos usos que se le dan hoy en día. Se ven en la sociedad en general para terminar viendo varios prototipos más específicos de la robótica aérea en los que se está investigando, concretamente los que llevan a que este pueda trabajar de manera autónoma.

#### 1.2.1 Medios Audiovisuales

El drone es un elemento que se ha incorporado últimamente en este sector debido a la cámara que pueden tener. Gracias a esto permite tomar planos de ciertas zonas o fotografías que serían muy difícil de obtener en condiciones normales. También se debe

## CAPÍTULO 1. INTRODUCCIÓN

---

a que el precio de éstos es asequible, por lo que se puede acceder a ellos con facilidad, y en un sector tan amplio y vistoso como es éste, lleva a un uso cada vez más común.



Figure 1.1: Drone grabando para la publicidad de una marca de coches.

### 1.2.2 Seguridad

Teniendo un drone en una nave industrial por ejemplo, se puede hacer que éste se desplace grabando en todo momento lo que ve, y si se detecta algo sospechoso en algún lugar el drone se dirija allí en el momento para obtener imágenes de lo que está pasando.



Figure 1.2: Empresa Prevent Security Sistems utiliza drones para videovigilancia en grandes superficies .

## CAPÍTULO 1. INTRODUCCIÓN

---

### 1.2.3 Sector agrícola

El uso de drones en este sector se encuadra en la agricultura de precisión. Se debe a la facilidad con la que un drone puede sobrevolar una zona y ofrecer imágenes de alta calidad, obteniendo una buena monitorización de los cultivos en menos tiempo, con menos gasto, y al tratarse de un vehículo eléctrico al evitar desplazamientos de otros automóviles conlleva un menor impacto ambiental. Además, permite ver con facilidad el estado de la cosecha, detectar enfermedades o plagas, posibilita fumigar desde el aire con mayor precisión, ya que se les puede programar una ruta y que la sigan. También podrían obtener otros datos como las zonas con más y menos agua, y obtener las condiciones del terreno y ver si son óptimas para esperar cierto resultado. En este tipo de drones, aparte de la cámara son de importancia otros sensores, como los de temperatura y humedad, o infrarrojos para captar el espectro infrarrojo de las plantas.



Figure 1.3: Empresa Novadrone utiliza drones para la gestión de las explotaciones agrícolas .

### 1.2.4 Inspección

En este apartado se engloban diversas actividades, como puede ser mantenimiento de edificios y construcciones, redes eléctricas o diversas instalaciones industriales como

## CAPÍTULO 1. INTRODUCCIÓN

---

aerogeneradores eólicos y estados de paneles solares. Al igual que en el apartado anterior, aquí se pueden recorrer grandes distancias, por ejemplo para comprobar las redes eléctricas, sin la necesidad de que un operario pierda mucho tiempo recorriendo dicha línea. Con las instalaciones solares por ejemplo, desde un plano superior se podría observar si todas las placas están en las condiciones óptimas y en caso de existir algún fallo identificarlo con facilidad. Para edificios y aerogeneradores lo que hay que tener en cuenta es la altura que pueden alcanzar, y a la cual con un drone llegaríamos con facilidad y observaríamos si hay algún problema. En estos casos lo que evitamos, como anteriormente se ha dicho, 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 anomalías, sin necesidad de que haya una persona revisando en todo momento las imágenes, donde también con una inversión inicial, al final se ahorra mucho tiempo y dinero. Un ejemplo de ésto se da en la empresa Iberdrola, la cual ha incorporado drones para el mantenimiento e inspección de sus infraestructuras, utilizando drones por ejemplo para el mantenimiento de las palas de los aerogeneradores. Unión Fenosa también ha incorporado los drones para la inspección de los tendidos eléctricos.

### 1.2.5 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. Un ejemplo de esto es el *Angel Drone*, proyecto desarrollado por la universidad de Sidney, que puede llevar materiales quirúrgicos o plasma sanguíneo.

## CAPÍTULO 1. INTRODUCCIÓN

---



Figure 1.4: Desarrollan en la universidad de Holanda un drone que lleva incorporado un desfibrilador .

### 1.2.6 Prototipos de investigación

Determinados grupos y grandes empresas están trabajando e investigando y construyendo prototipos con drones para distintas labores. Esto se debe a la facilidad que pueden ofrecer los drones para realizar tareas como el control de 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 para otros vehículos.

Como pionero en este área se encuentra **Amazon**, el cual lleva desarrollando desde 2013 una tecnología que permita el reparto de paquetes mediante drones. La idea cuenta con doce prototipos, debido en parte a los distintos tipos de UAV que tienen. 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 drone 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.

## CAPÍTULO 1. INTRODUCCIÓN

---

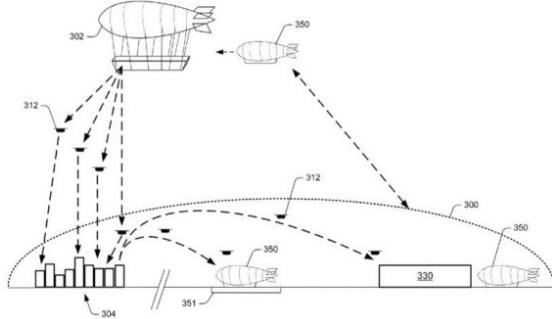


Figure 1.5: Esquema de como será el proceso para la entrega de paquetes.

El 7 Diciembre de 2016, Amazon Prime Air (servicio de drones repartidores) realizó la primera entrega de una compra real en la región de Cambridge, Reino Unido. Se puede encontrar este vídeo, además de otros, en su página web.<sup>1</sup>

Por otro lado, también para el reparto de mercancías se encuentra **Google**, llegando a tener un programa piloto en Australia en el año 2014, pero no consiguió llevarlo a Estados Unidos. Aun así, consiguió hacer pruebas de reparto de comida en una universidad, un reto que supuso principalmente que la comida llegara rápido 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, 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 códigos de cada paquete, enviando esta información a un servidor y que éste pueda tener controlados los paquetes que hay y dónde están situados, como se puede ver en la figura 1.5.

---

<sup>1</sup><https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011>

## CAPÍTULO 1. INTRODUCCIÓN

---



Figure 1.6: Drone detectando código RFID.

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 éste tiene que ir al almacén a buscarlo, perdiendo mucho tiempo entre la distancia recorrida y la búsqueda del producto. Lo que conseguirían es que en caso de que los empleados estén ocupados, un cliente no tenga que estar a la espera, sino que con un dispositivo pueda pedir el producto y un dron se encargará de ir a por él y llevarlo al punto donde el cliente se encuentre. Destacar que se incorporarán en las tiendas controladores aéreos para que los vehículos sigan una trayectoria segura.

### 1.3 Hardware de drones

Hay que destacar las partes que tiene un dron 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 se gana con esto son funcionalidades como poder permitirse un aterrizaje y un despegue totalmente vertical, sin depender de un espacio en el que coger velocidad para levantar el vuelo, e igual con el aterrizaje, pudiendo el dron estando quieto en

## CAPÍTULO 1. INTRODUCCIÓN

---

el aire bajar totalmente en vertical hasta posarse sobre el suelo. Una vez en el aire pueden moverse adelante, atrás, izquierda, derecha, arriba, abajo y combinaciones de movimientos entre ejes, además de los giros 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.

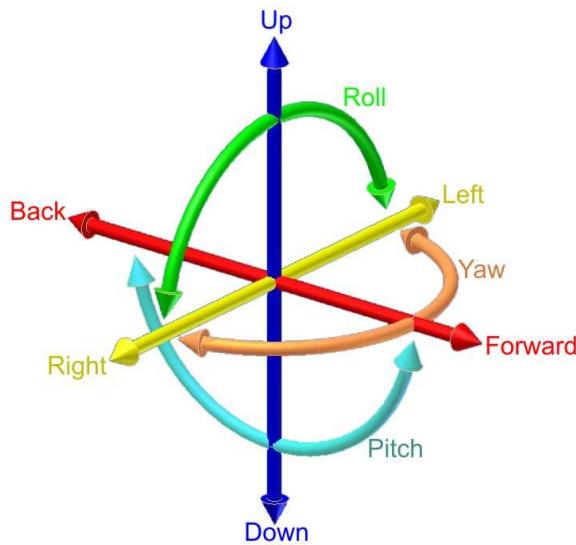


Figure 1.7: Diferentes movimientos que puede realizar un dron.

Un desglose explicando cada una de las partes sería:

**Marco:** También conocido como estructura o chasis. Es la estructura principal sobre la que se sitúan el resto de los elementos. Cambiará su forma dependiendo del dron, variando la longitud de las patas o el número de soportes para hélices, por ejemplo. 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 él la batería. Otro material que suele utilizarse es la fibra de carbono, ya que pesa poco y es muy resistente, aunque puede tener factores negativos como su

## CAPÍTULO 1. INTRODUCCIÓN

---

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 drone.

**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 éstas hacer su trabajo. Éste, a su vez, tiene distintos parámetros que serán principalmente los que permitan al drone llevar mayor velocidad:

- El número de vueltas que dé por minuto, lo que dependerá de los KiloVoltios. Suele estar en torno a 800-900kV.
- El tamaño que el drone tenga. Al mirar las especificaciones de un drone 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, 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 drone pueda realizar un vuelo, permitiendo trabajar a la placa controladora y motores. La característica principal de las baterías son los miliamperios hora, ya que es la que permitirá una mayor capacidad y por lo tanto que el drone 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 drone 3DR Solo. También

## CAPÍTULO 1. INTRODUCCIÓN

---

es importante la tasa de descarga, que es 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 drone, evitando así que se quede sin energía a mitad de un vuelo.

**Equipo de transmisión:** Es el encargado de que se comunique el drone con una estación receptora. Puede variar en función del aparato ya que se pueden usar diferentes tecnologías, pero principalmente usa radiofrecuencia o Wifi. Existen casos, como el modelo 3DR, que combina ambas tecnologías, utilizando la radiofrecuencia para la información del movimiento, batería y posicionamiento, y el WiFi para la transmisión de imágenes en directo. Se pueden encontrar distintos equipos de sistemas de transmisión, uno de los últimos y mas 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. Su principal característica es que no pierde información cuando no hay contacto directo entre las dos estaciones. La vía WiFi es muy utilizado, pues permite controlar el drone desde una aplicación móvil, por lo que conectando estos dos tendríamos un mando que nos permite cambiar gran parte de la configuración del drone. También existen dispositivos que permiten el control mediante Bluetooth, pero es menos común ya que tiene mayor restricción de velocidad de datos y distancia.

**Placa controladora:** Es el procesador del drone, el que se encarga de recoger la información del drone y cuando le llega una orden ver que información tiene que mandar para que ésta se ejecute de forma correcta, así como en caso de haber un problema tratar de evitarlo. Hay una gama muy amplia:

- Pixhawk<sup>2</sup>: Es el más utilizado debido a que trabaja con 3DRobotics y Ardupilot.

Sirve para diversos dispositivos como drones, helicópteros y barcos. Está pensado

---

<sup>2</sup><https://pixhawk.org/>

## CAPÍTULO 1. INTRODUCCIÓN

---

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. Es un piloto automático en tiempo real y muy eficiente, proporcionando un entorno de estilo POSIX. Éste es el autopiloto estándar de la industria, y por lo tanto, como veremos a continuación, a partir él se han desarrollado diversos autopilotos con distintas mejoras.

- Pixhawk2: Es una versión avanzada de la placa anterior. Tiene mejoras como aislamiento de vibraciones, 3 IMUs para redundancia (3 acelerómetros, 3 giróscopos, 3 magnetómetros y 2 barómetros) y sensor para controlar la temperatura.
- PixRacer: 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 ésta en un controlador de drone.
- PXFmini: Es otro piloto automático de Raspberry Pi. Éste tiene la electrónica para la mayoría de los componentes que puede utilizar un drone.
- FlytPOD: Es 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 es una versión extendida de la anterior, con más 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.

## CAPÍTULO 1. INTRODUCCIÓN

---

Hay que destacar un elemento importante como es la cámara, que aunque no todos los drones la llevan sí es bastante común. Algunos la llevan incorporada (incluso dos cámaras, una que apunta hacia la parte de delante y otra que apunta la parte de abajo) y otras que traen soporte para incorporar ciertas cámaras, normalmente consideradas cámaras de acción, para obtener una mejor calidad. Algunos drones incorporan una Gimbal para controlar la parte hacia la que queremos que apunte la cámara en cada momento o utilizarlo como estabilizador, para evitar así que afecten a la imagen diversos movimientos, generalmente bruscos, que pueda realizar el drone.

En 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 esta linea uno de los sistemas más impactantes es el conjunto que se ha creado con el drone **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 drone, y con cualquier movimiento que sientan las gafas la cámara del drone lo imitará. En caso de que esto parezca incómodo tiene un joystick con el que también se pueden ordenar los movimientos realizar a la cámara.



Figure 1.8: Distintas partes del dron.

## 1.4 Software de drones

El software es el conjunto de programas que permiten realizar ciertas tareas, por ejemplo al drone una navegación autónoma. Dicho programa estará instalado en la placa controladora, y por tanto será el que ejecute para recoger la distinta información de los sensores, procesar la información y enviar las órdenes correctas a los distintos elementos. El desarrollo de software para este tipo de robots ha evolucionado mucho en los últimos años debido al uso civil que se le comienzan a dar y no tanto al desarrollo militar. Existen varios entornos software que permiten el manejo y la programación de estos robots:

- **Ardupilot<sup>3</sup>:** Es un sistema OpenSource encargado de recibir la información que se le da y de enviar las señales correspondientes a los actuadores. Se trata del software más importante por lo completo que es y la 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 han desarrollado. Este software se caracteriza por la variedad de dispositivos que puede llegar a controlar, ya que trabaja con diversos dispositivos aéreos (aviones, helicópteros, drones, etc) y con dispositivos marinos (como son los barcos y submarinos). Éste ha tenido un gran desarrollo debido a que es código abierto. Hay mucha gente creando interfaces para él 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 sólo permite trabajar con plataformas de los mismos creadores. Otra característica es la facilidad con la que se le pueden añadir diferentes sensores, como pueden ser módulos GPS o cámaras, algo que facilita la navegación autónoma.
- **Megapirate-NG:** Apareció como desarrollo del anterior. La funcionalidad de

---

<sup>3</sup><http://ardupilot.org/ardupilot/index.html>

## CAPÍTULO 1. INTRODUCCIÓN

---

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 más cómodas para trabajar, puede que estén atrasadas.

- **MultiWii:** 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 son los entornos software principales que estarían sobre el vehículo, pero también están los programas que se ejecutarían en otros dispositivos como el ordenador o el teléfono móvil para ver la información que éste 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 comunicar el vehículo con la estación terrena. Aquí hay un protocolo que destaca sobre los demás, el MAVLink (Micro Air Vehicle Communication Protocol). Este protocolo tiene la información contenida en ficheros .xml, lo que permite utilizarlo en diversos lenguajes de comunicación, y 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. Otra ventaja es que hay muchos software de drones que lo soportan, como pueden ser Ardupilot, Autopilot, algunos derivados de éstos 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 están encriptados en la comunicación, por lo que es más fácil un ataque y que se manipulen los datos. Además detecta si se pierde algún dato ya que utiliza CRC (código de redundancia cíclica para detectar cambios en los datos, este se utiliza en protocolos como TCP). MAVLink utiliza otro software llamado MAVProxy para acceder a los datos del vehículo, como la velocidad y las imágenes, lo que permite también saber qué

## CAPÍTULO 1. INTRODUCCIÓN

---

datos mandarle para que funcione de forma correcta.

Destacar ROS (Robot Operating System), meta sistema operativo de código abierto mantenido por la Open Source Robotics Fundation(OSRF). ROS tiene librerías y herramientas que permiten desarrollar software para robots. Es el software para robots mas extendido en el mundo, lo que conlleva que sea el que mas aplicaciones tiene para la robótica. También ofrece mucho software para el manejo de drones. En mayo de 2010 mostró el primer vídeo en el que un drone conseguía volar utilizando ROS. En esta demostración el drone se movía con trayectorias agresivas, para mostrar su correcto funcionamiento. Desde entonces ha seguido sacando herramientas para ver el estado del drone conectando con los sensores, imágenes de las camaras o GPS. PIXHAWK también integró su sistema con ROS, lo que hizo que éste todavía creciera mas. También tiene interfaz para el control del ArDrone. Por otro lado, en protocolos de comunicación, MAVLink lanzó también un software para la compatibilidad con ROS. Como se puede observar, al igual que este software es el mas extendido en la robótica en general, ha conseguido una gran importancia también en la robótica aérea.

A parte de todos éstos, existen también otras infraestructuras software como puede ser **JdeRobot**, que es la que hemos usado en este trabajo y que se describirá con mas detalle en el capítulo 3.

### 1.5 Robótica aérea en RoboticsLab de la URJC

En un contexto mas cercano, este proyecto se inició tras otros realizados anteriormente en el laboratorio de robótica de la universidad. Los ejemplos mas destacables son los siguientes:

Alberto Martín <sup>4</sup> [1] trabajó en el seguimiento de objetos con un drone utilizando su cámara. Tenía un controlador reactivo PID, y el drone tenía que seguir una pelota de color rosa, consiguiendo que le siguiera en un espacio 3D, según muestra

---

<sup>4</sup><http://jderobot.org/Amartinflorido-tfm>

## CAPÍTULO 1. INTRODUCCIÓN

---

la figura 1.8. El drone en caso de no encontrar la pelota rosa se quedaba parado donde estuviera. Adicionalmente en este proyecto se desarrollo el driver para el ArDrone del Parrot real, que es el mismo que hemos utilizado en este proyecto.

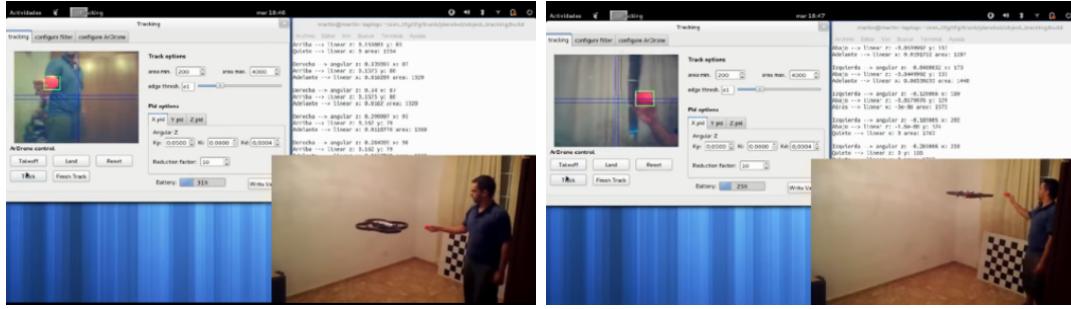


Figure 1.9: TFG Alberto Martín.

Arturo Velez<sup>5</sup> [2] trabajó en un seguimiento visual pero sin filtro de color, en el cual un drone debe seguir una textura que se mueve por el suelo detectando los puntos de interés como se muestra en la figura 1.9. En caso de que el drone pierda la referencia de la figura en la imagen, realiza un algoritmo de búsqueda.

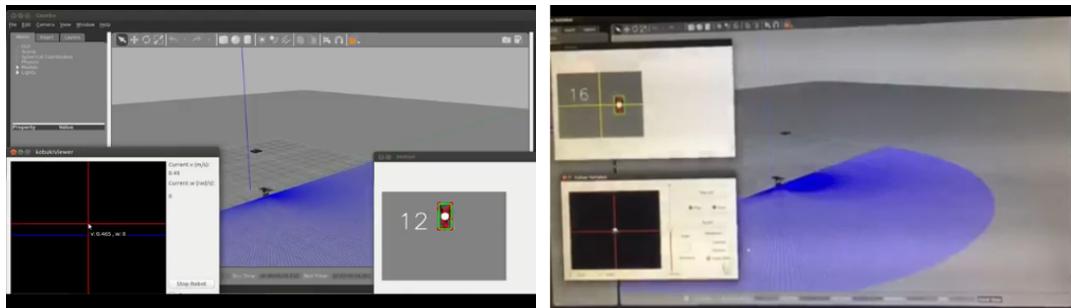


Figure 1.10: TFG Arturo Velez.

Manuel Zafra<sup>6</sup> [3] trabajó en la localización del drone mediante April Tags. El objetivo de éste era recorrer autónomamente una ruta tridimensional en interiores,

<sup>5</sup><http://jderobot.org/Avelez-tfg>

<sup>6</sup><http://jderobot.org/Mazafra-v-pfc>

## CAPÍTULO 1. INTRODUCCIÓN

---

como secuencia de puntos 3D para lo que necesitaba estar localizado. Gracias a las April Tags que detectaba el drone podía detectar el punto en un mapa 3D en el que estaba situado como se muestra en la figura 1.10.

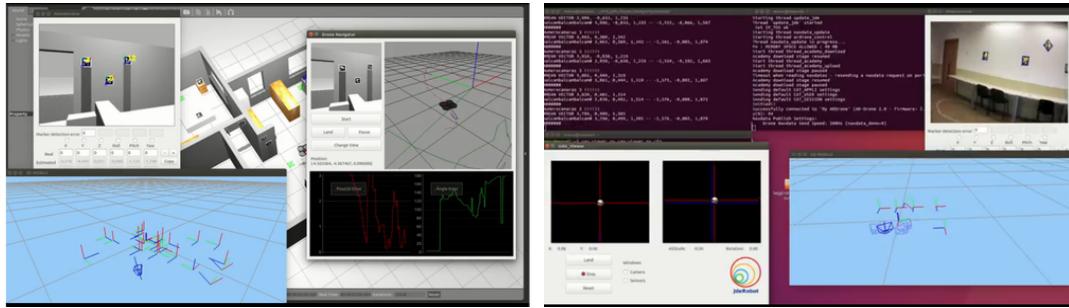


Figure 1.11: TFG Manuel Zafra.

Daniel Yagüe<sup>7</sup> [4] trabajó con el ArDrone sobre Gazebo, probando distintas funcionalidades y escenarios. Por un lado trabajó en el driver para el cuadricoptero simulado, consiguiendo un plugin para el ArDrone que proporciona los datos de los sensores a bordo y que acepta y ejecuta órdenes como despegar, aterrizar y los distintos movimientos de vuelo. Por otro lado sobre simulador programó aplicaciones de control visual, haciendo que el drone siguiera una linea, una carretera o que un drone siguiera al otro, como se muestra en la figura 1.11.

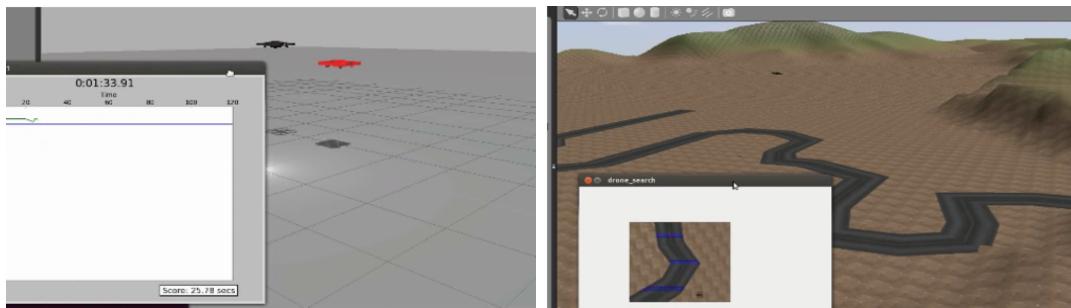


Figure 1.12: TFG Daniel Yagüe.

<sup>7</sup><http://jderobot.org/Daniyague-pfc>

## CAPÍTULO 1. INTRODUCCIÓN

---

Jorge Cano <sup>8</sup> [5] trabajó en el diseño, construcción y programación de un drone real. El driver que realizó para su control se basa en el protocolo MAVLink, en parte por su gran popularidad. Por una parte trabajó en el diseño y construcción de la plataforma hardware, como se muestra en las dos primeras imágenes de la figura 1.12, y por otra en el controlador software para la comunicación con el vehículo. Para hacer tests de vuelo realizó pruebas perceptivas y de control, con filtros de colores y un controlador PID, como se muestra en la tercera imagen de la figura 1.12.



Figure 1.13: TFG Jorge Cano.

Una vez terminada la breve introducción, se va a explicar lo realizado en este proyecto. En el capítulo 2 se redactan los objetivos propuestos y la metodología seguida para llegar hasta ellos. En el capítulo 3 se presenta la infraestructura utilizada y como se ha trabajado con ella. En el capítulo 4 se describe el algoritmo realizado, cómo se ha programado y porqué se han seguido unos u otros caminos. En el capítulo 5 se detalla los distintos experimentos realizados y los resultados que se han obtenido. Por último, en el capítulo 6 se incluyen las conclusiones a las que se han llegado tras finalizar este proyecto.

---

<sup>8</sup><http://jderobot.org/J.canoma-tfg>

# Chapter 2

## Objetivos

Una vez presentado el contexto de este TFG en el primer capítulo, tanto en general como el más cercano, en este capítulo se van a explicar los objetivos concretos a conseguir y la metodología utilizada para llegar a ello.

### 2.1 Objetivo principal

El objetivo propuesto de este trabajo es conseguir que un drone navegue desde una baliza de partida a otra baliza de llegada, que debe reconocerse por visión y cuya posición es desconocida. Para esto, se han propuesto tres fases principales:

- **Percepción visual:** La percepción se basa un filtro de color. Dada una imagen de entrada, se deberá detectar una baliza previamente asignada para, en función de lo obtenido, enviar una información u otra al drone. Habrá que tener en cuenta objetos que sean del mismo color de la baliza, los cuales querremos descartar para que no interfieran.
- **Control:** En función de lo que perciba y del momento en el que se encuentre, se tomará una u otra decisión para realizar los movimientos oportunos en cada mo-

## CAPÍTULO 2. OBJETIVOS

---

mento. Es importante que éstos no sean bruscos, provocando desestabilizaciones del drone.

- **Validación experimental:** Cada vez que se da un avance en los apartados anteriores, hay que validar su funcionamiento, tanto en simulación como en el drone

## 2.2 Requisitos

Para la realización de los objetivos anteriormente citados, se tiene que conseguir una solución que cumpla las siguientes características:

- Que funcione sobre JdeRobot-5.4.2, utilizando las herramientas necesarias para el control del drone, filtros de color y diagrama de estados.
- Que el algoritmo sea vivaz, la frecuencia de iteraciones es importante para un comportamiento ágil y fluido del drone, el procesamiento de imágenes o el control del drone no puede ser pesado computacionalmente, pues puede llevar a un algoritmo lento, y por lo tanto un mal control del drone.
- Que el algoritmo sea robusto, teniendo que funcionar tanto en el simulador, con imágenes puras y sin perturbaciones del vuelo, como en el drone real ArDrone2 de Parrot, donde las imágenes no son ideales, existe ruido o donde sí hay corrientes y turbulencias que afectan al vuelo.
- Programado en Python 2.7.

## 2.3 Metodología

Se propuso un desarrollo en espiral. Para ello se proponían semanalmente reuniones con el tutor, en las cuales se presentaban unos objetivos a seguir en función

## CAPÍTULO 2. OBJETIVOS

---

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. Después se probaban en simulación, en el robot real o en ambos, dependiendo del objetivo. Por último se proponía otra reunión para determinar si los avances eran los deseados o no, y en función de esto proponer los nuevos objetivos.



Figure 2.1: Método de desarrollo en espiral

Se ha mantenido una bitácora web en la que se ha ido documentando el progreso en el desarrollo, el cual ha quedado reflejado mediante vídeos e imágenes en <http://jderobot.org/Jvela-tfg>. Además el código está disponible públicamente accesible GitHub <https://github.com/RoboticsURJC-students/2016-tfg-jorge-vela>.

## 2.4 Plan de trabajo

La planificación seguida en el desarrollo ha incluido las siguientes fases:

## CAPÍTULO 2. OBJETIVOS

---

- Formación: 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 únicamente en un trabajo sobre simulador.
- Desarrollo de la percepción robusta de la baliza: Comenzar a trabajar con imágenes, datos que se obtenían y los cambios que podíamos hacer para obtener datos deseados a partir de los cuales mandar órdenes. Para ésto es muy importante el uso de la biblioteca OpenCV.
- Desarrollo de control visual de aterrizaje y despegue: Unir estas dos tecnologías y conseguir un buen algoritmo que nos permitiera enviar información a partir de la imagen de la baliza obtenida en tiempo real a un drone y que éste la ejecutara de forma correcta y fluida.
- Desarrollo del control de búsqueda: Al igual que en el apartado anterior, hay que enviar información al drone en tiempo real diciendo lo que tiene que hacer, pero en este caso diciendo cómo se tiene que comportar cuando no hay objetos de interés o cuando hay objetos que posiblemente lo sean.

# Chapter 3

## Infraestructura utilizada

Después de fijar los objetivos concretos de este TFG, en este capítulo se va a hablar sobre las distintas tecnologías utilizadas, tanto hardware como software, y el uso que han tenido en este proyecto.

### 3.1 Parrot AR.Drone 2.0

Éste ha sido el drone utilizado para hacer las pruebas en un entorno real y comprobar así que el desarrollo del algoritmo era el correcto.

La batería era de 1500mah, siendo la duración en tiempo de vuelo de unos 12 minutos.

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 drone 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 aplicación específica para ello), o como en este caso, mediante el ordenador, y de esta forma poder controlarlo. Esta placa base funciona con Linux.

## CAPÍTULO 3. INFRAESTRUCTURA

---

Además, dicho drone 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.

Ya para terminar, la *velocidad* de este drone puede llegar a alcanzar los 18km/h, algo que en este proyecto nunca se ha 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 drone tardaría 10 segundos en salirse de este rango.

### 3.2 Simulador Gazebo

Es un proyecto de Open Source Robotics Fundation distribuido bajo la licencia Apache 2.0. Durante este trabajo se ha trabajado con Gazebo 7.

Tiene la capacidad de simular de forma precisa ambientes con robots, objetos y sensores en distintos tipos de entornos. Ésto permite trabajar con el Parrot ArDrone y realizar pruebas. 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 escenario principal aquí utilizado consistía en el drone y un coche con una baliza, el cual tenía que encontrar, centrarse sobre ella y aterrizar.

Destacar que este simulador se utilizó en el DARPA Robotics Challenge, programa que trata de abordar problemas de operaciones humanitarias, como son casos de socorro en desastres naturales, que pueden ser demasiado grandes para que el ser humano se enfrente a ellos y responda de forma adecuada.

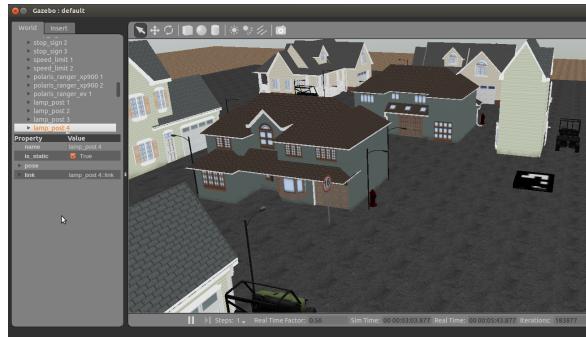


Figure 3.1: Mundo Gazebo.

### 3.3 Entorno JdeRobot

Éste es el software principal con el que se ha trabajado, cuya web oficial es [www.jderobot.org](http://www.jderobot.org). Se trata de un proyecto de software libre que se centra en la robótica y en la visión por computación. Este software cuenta con distintas herramientas como filtro de color o control de los diferentes robots, que ha permitido un continuo desarrollo desde el inicio.

#### 3.3.1 Driver ArDroneServer

Éste es el servidor que permite comunicar nuestra aplicación con el drone. Tiene dos partes principales. Por un lado se encuentra el ArDrone SDK, el cual se comunica con el robot aéreo mediante WiFi. Por otro lado se encuentra el envoltorio C++, el cual se comunica con la aplicación mediante las interfaces ICE, permitiendo de esta forma la comunicación de nuestra aplicación con el cuadricoptero.

#### 3.3.2 Plugin modelo de cuadricoptero Gazebo

Plugin que permite tener un ArDrone2 de Parrot en el simulador. Éste tiene distintos sensores con los que trabajar (sonar, IMU y cámara), además de contar con

## CAPÍTULO 3. INFRAESTRUCTURA

---

los distintos controles (cargar,iniciar, actualizar,despegue, velocidad y aterrizaje). Por otro lado, cuenta con la interfaz ICE para poder comunicar con los distintos elementos.

### 3.3.3 Herramienta UAVviewer

Esta herramienta permite el control del drone y obtener los datos de sus sensores. Su interfaz ICE tiene los drivers de Cámara, Pose3D, CMDVel, NAVData y ArDrone\_Extra.

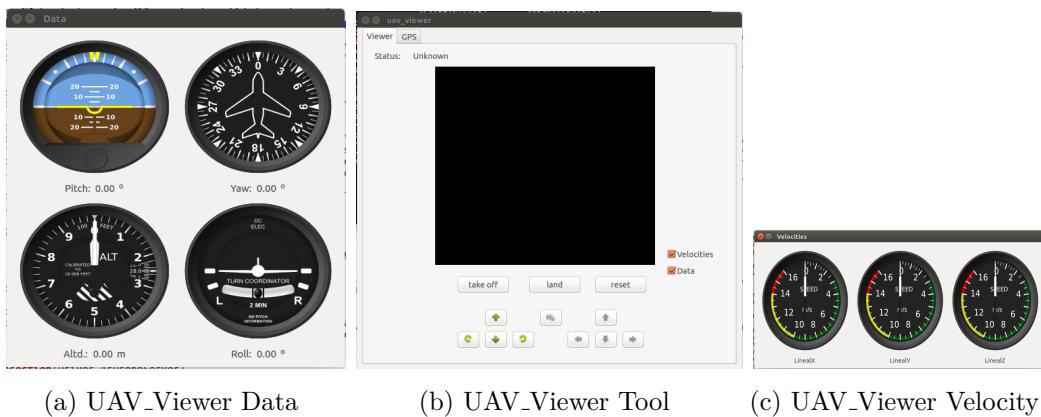


Figure 3.2: UAV viewer

### 3.3.4 Herramienta ColorTuner

Al inicio del proyecto era la herramienta *ColorFilter*. Esta herramienta permite, a partir de una imagen o vídeo, trabajar en los distintos espacios de color (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 cuáles no, viéndose de esta forma, en otra imagen, sólo los colores que pasan este filtro dejando el resto como un fondo negro.

### 3.3.5 Práctica choca-gira con autómatas.

De esta práctica se ha obtenido la máquina de estados, la cual permite crear diferentes estados, crear transiciones de unos a otros y marcar un estado u otro para saber en que punto se encuentra la aplicación. Con los siguientes comandos en el código python, la aplicación crea una máquina (indicando el número de estados que tendrá), añade un estado (indicando el nombre de este), añade una transición (indicando origen, destino y el nombre) y marca el estado en el que se encuentra(indicando el numero del estado y TRUE para activarlo).

```
Machine(n)
machine.addState(name)
machine.addTransition(orig, fin, name)
machine.setStateActive(n, flag)
```

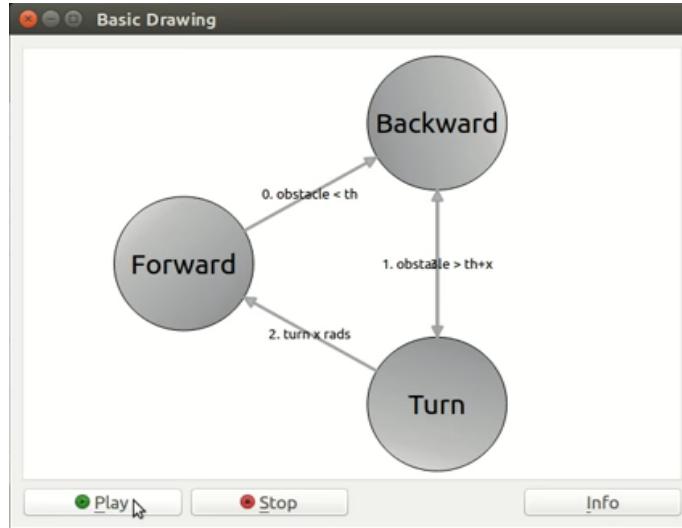


Figure 3.3: Diagrama de estados.

Esta práctica se encuentra en el GitHub de Jderobot Academy [https://github.com/JdeRobot/Academy/tree/master/src/bump\\_and\\_go\\_py](https://github.com/JdeRobot/Academy/tree/master/src/bump_and_go_py)

### 3.3.6 Práctica follow\_turtlebot.

Al igual que UAV\_Viewer 3.3.3, permite controlar el movimiento del drone, así como ver los datos de sus sensores. A esta interfaz se le ha añadido la implementación de la maquina de estados obtenida de la práctica de choca-gira, teniendo de esta forma el control del drone y poder visualizar el estado en el que se encuentra.

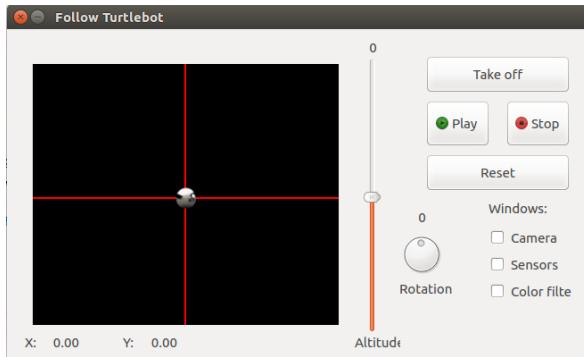


Figure 3.4: Follow Turtlebot.

Esta práctica se encuentra en el GitHub de Jderobot Academy [https://github.com/JdeRobot/Academy/tree/master/src/follow\\_turtlebot](https://github.com/JdeRobot/Academy/tree/master/src/follow_turtlebot)

Con la aplicación de la práctica de follow\_turtleboot como interfaz, mas el componente añadido, el componente sobre el que se ejecuta el algoritmo está compuesto por tres hilos. El primer hilo es el encargado de la ejecución del algoritmo. El segundo hilo es el encargado del interfaz gráfico para el control del drone, la cámara y los sensores. El tercer hilo es la ventana que permite ver el diagrama de estados.

## 3.4 Biblioteca OpenCV

Esta librería tiene un conjunto de funciones que sirven para el procesamiento de imágenes y visión computerizada, cuya web oficial es <https://opencv.org/>. Está programada en C++ y es un estandar de facto en la visión artifical. Para el

## CAPÍTULO 3. INFRAESTRUCTURA

procesamiento de imágenes nos hemos apoyado principalmente sobre ésta, trabajando en la versión 3.1. Gracias a ésta, al obtener la imagen que transmite el drone se puede tanto detectar objetos como aplicar cambios sobre ella, para marcar las zonas de interés o diferentes objetos. Permite la realización de filtros de color, para eliminar objetos no deseados dependiendo el momento. Además permite el uso de operadores morfológicos (erosión y dilatación) gracias a los cuales se evitan imperfecciones en las imágenes como puede ser el ruido, que clasifica objetos inexistentes o de no interés como objetos de interés, así como zonas importantes las detectaba como píxeles de fondo. También han sido de gran utilidad funciones como *drawContours* ó *findContours*, las cuales permiten encontrar los bordes de los objetos e indicar sobre una imagen final, con otro color, donde se encuentran dichos objetos.

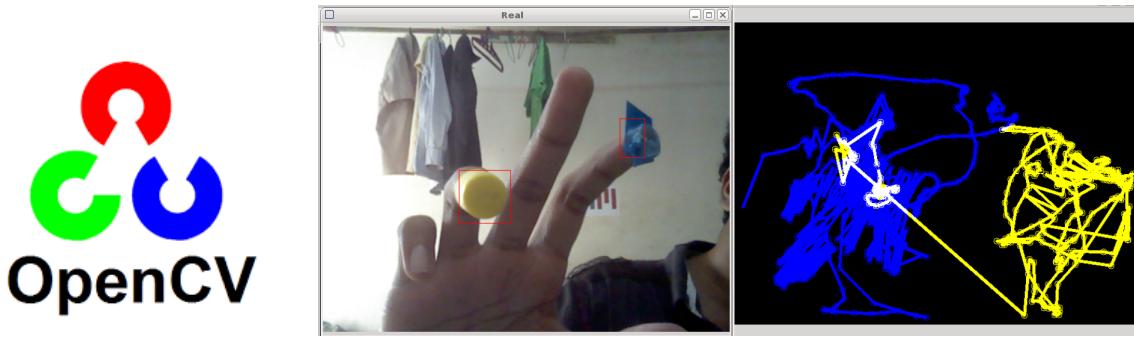


Figure 3.5: OpenCV

# Chapter 4

## Algoritmo de Navegación autónoma

En este capítulo se describe el modo por el cual, con la infraestructura que se tiene, se ha llegado a una solución para los objetivos planteados.

Este algoritmo tiene que permitir al drone despegar de forma controlada, realizar una navegación autónoma para encontrar una baliza sobre la que aterrizar. La organización de este capítulo tiene primero una sección de diseño en la que se explica el funcionamiento del programa. Tras esto una sección de percepción para explicar los datos que se obtienen a través de los sensores. Una sección de control, para explicar los distintos movimientos del drone y de que dependen estos. Y finalmente se explica la arquitectura software en la que se han programado las partes perceptivas y de control del algoritmo.

### 4.1 Diseño

El diseño de este algoritmo consta de un comportamiento reactivo de iteraciones continuas. Es un proceso basado en adquisición-procesado-envío de órdenes, tal y como muestra la figura 4.1. La adquisición de los datos se realizan mediante los sensores del drone. Estos datos sensoriales serán recogidos para su procesamiento y tras ésto se

## CAPÍTULO 4. ALGORITMO

---

enviarán las instrucciones al drone para que las ejecute. El sensor utilizado en este proyecto ha sido la cámara, y los movimientos del drone dependerán de lo que esta capte en cada momento.

Por otro lado, la parte de control es un autómata finito de estados, el cual comienza en un estado inicial (despegue), y en función de lo que recibe a la entrada (imagen del sensor), realizará el procesamiento necesario para producir la información que enviar al drone, y en ocasiones le llevará a pasar de un estado a otro. Este autómata está compuesto por 6 estados: despegando, buscando, posible baliza, centrándose en la baliza, aterrizando y aterrizado. A excepción del primero, los demás estados dependerán de lo que detecte el drone en cada momento. El primer estado está controlado por tiempo, pues son 10 segundos al iniciarse en los que el drone inicia el vuelo sobre una baliza y trata de estar centrado sobre ésta, para así tener un despegue controlado y evitar que el drone se mueva en caso de tener alguna deriva o haya factores externos que produzcan esto. Una vez transcurridos los 10 segundos se pasará al estado de búsqueda e irá pasando por los distintos estados hasta su aterrizaje, según se explicará con detalle en la sección 4.3.

## CAPÍTULO 4. ALGORITMO

---

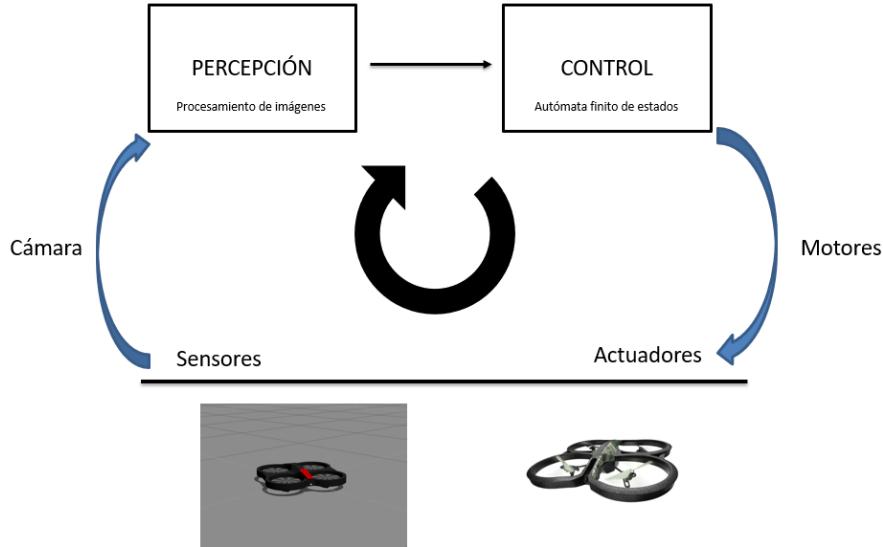


Figure 4.1: Esquema que muestra la adquisición, proceso y envío de ordenes

El lugar de aterrizaje del dron es una baliza previamente definida 4.2. Esta baliza es un cuadrado que en su interior tiene cuatro cuadrantes, dos naranjas, y dos azules o verdes, dependiendo si se trabaja con el dron real o con el simulador. Esta baliza se diseñó así para que sea difícil confundirla con otro objeto, pues de ser una baliza simple se podrían confundir los colores y para que se pudiera detectar del mismo modo a diferentes alturas. Lo que busca el software será la cruceta que forman estos cuatro cuadrados y el punto central de ésta. Lo que buscará es un objeto que tenga determinadas características y patrones, algo que se puede detectar a diferentes alturas, distintas condiciones y situaciones.



Figure 4.2: Baliza utilizada para el dron real.

Por último, para el control del drone con nuestro algoritmo se ha utilizado la herramienta *follow\_turtlebot* de JdeRobot Academy, explicada en 3.3.6. Esta aplicación tiene un interfaz gráfico de usuario (GUI) que permite controlar el drone y ver los datos de los distintos sensores, así como la imagen que obtiene la cámara. Por otro lado, cuenta con las interfaces ICE que permite la comunicación con el servidor, pudiendo así recibir datos de sensores y motores, y enviar las instrucciones de velocidad necesarias en cada momento. La imagen de la aplicación se encuentra en la figura 3.4

## 4.2 Percepción

En esta sección se va a tratar la obtención de la imagen de la cámara y el procesamiento que se realiza sobre ésta. Gracias a lo que el drone ve en todo momento, sabe el punto del comportamiento en el que se encuentra y la información que debe enviar. En primer lugar se obtiene una imagen de entrada. Esta imagen es procesada con filtros de color y operadores morfológicos, obteniendo una imagen de salida. A partir de los datos de esta imagen, se detecta si hay objetos de interés o no, y por tanto se envía unas instrucciones u otras al drone.

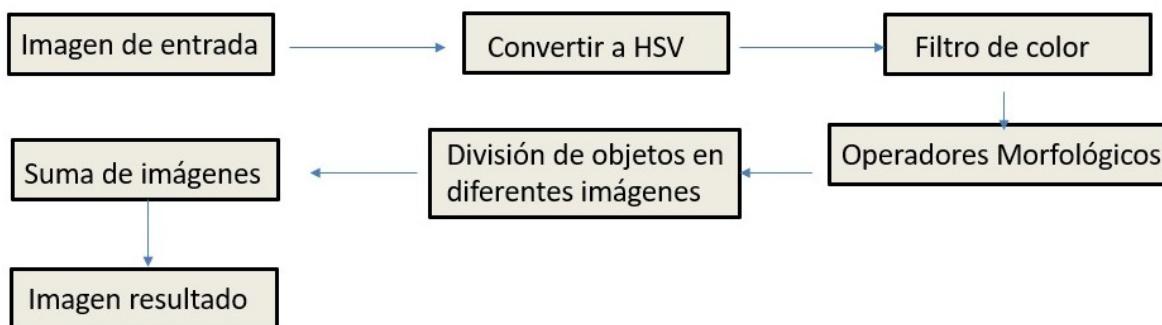


Figure 4.3: Esquema de percepcion del algoritmo

### 4.2.1 Pre-procesado

Una vez tenemos la imagen de entrada, hay que detectar la información de interés que nos aporta. En primer lugar, la imagen que está en RGB se transforma a HSV, para que sea más fácil su interpretación, ya que en lugar de trabajar en función de tres colores puros (rojo, verde y azul) se trabaja en función de tres parámetros (tono, saturación y valor). De esta forma, se depende menos de la luz que haya en cada momento y en cada lugar. En segundo lugar se pasan los filtros de color a la imagen. En el filtro a cada uno de los parámetros se le asigna un rango de valores entre 0 y 255. La herramienta *colorTuner* de JdeRobot permite, a partir de una imagen de entrada, dar valores a estos parámetros, viendo que objetos cumplen estos requisitos y dejándolos en primer plano, y cuáles no, dejando estas zonas en negro como píxeles de fondo. Una vez se obtienen estos valores se añaden al filtro, obteniendo a la salida una imagen que solo muestra los objetos de los colores de interés.

El siguiente código es un ejemplo de cómo a partir de una imagen de entrada en RGB, se transforma a HSV y se filtran los objetos de color naranja.

```
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)
```

En caso de trabajar sobre simulador, con esto se obtiene a la salida una imagen bastante parecida a la deseada, debido a la pureza de los colores. Sin embargo, al trabajar sobre imágenes reales, la imagen de salida de este filtro aún tiene ruido y objetos de interés imperfectos. Para arreglar esto se utilizan los operadores morfológicos, que eliminan estas imperfecciones. Una breve explicación de los utilizados es la siguiente:

- **Erosión:** Operación morfológica que comprime los píxeles en primer plano,

## CAPÍTULO 4. ALGORITMO

---

eliminando así píxeles aislados.

- **Dilatación:** Transformación dual a la erosión. Operación morfológica que dilata los píxeles en primer plano.
- **Cierre:** Realizar una dilatación en la imagen seguida de una erosión.
- **Apertura:** Realizar la erosión en una imagen seguida de una dilatación.

De esta forma, al obtener una imagen sin apenas ruido, se evita que objetos de no interés los detecte como tal y que los objetos de interés los trate como píxeles de fondo. La figura 4.4 muestra la realización de un filtro de color y el uso de operadores morfológicos:

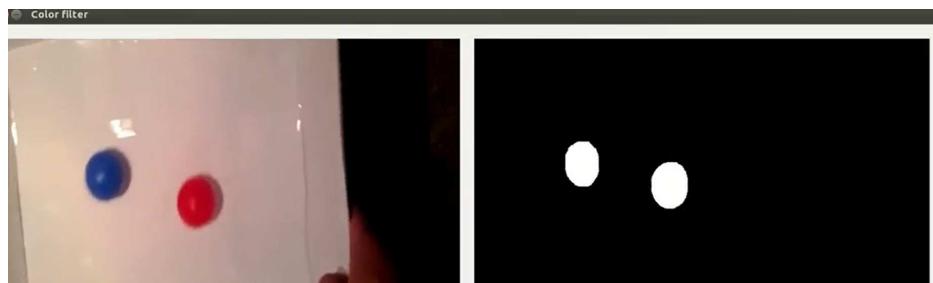


Figure 4.4: Ejemplo de un filtro de color y uso de operadores morfológicos para eliminar ruido.

El siguiente fragmento de código muestra cómo se define la matriz a partir de la cual se realizarán las operaciones oportunas y se realizan la erosión y la dilatación.

```
kernel = np.ones((3,3),np.uint8)
maskRGBOrange = cv2.erode(maskRGBOrange,kernel,iterations = 4)
maskRGBOrange = cv2.dilate(maskRGBOrange,kernel,iterations = 3)
```

### 4.2.2 Detección de la baliza

Una vez se obtienen los colores de los objetos de interés, hay que buscar el objeto deseado. Para esto hay que buscar el punto de intersección entre los cuatro cuadrados de la baliza. Para ello los pasos a realizar son los siguientes:

1. Para la imagen de entrada se hacen dos filtros de color, uno por cada color de la baliza tal y como se ha descrito en la subsección previa.
2. De cada imagen se obtienen el número de objetos, las áreas y los contornos. En cada objeto se obtiene el valor de su área, en caso de ser menor de un valor determinado, se descarta. En caso de tener ese área o mayor, se crea una imagen negra, se pintan los contornos del objeto sobre ésta con un valor RGB(0,1,0) y se dilatan. De esta forma, tendremos tantas imágenes como objetos.
3. Se suman las imágenes obtenidas. Sobre una imagen negra final se pintan los contornos dilatados de todos los objetos. Cada contorno se pintará con un valor RGB(0,1,0), por lo tanto en el caso de que varios objetos coincidan sumarán sus valores. Con esto conseguiremos que el punto donde interseccionen cuatro objetos tenga un valor (0,4,0), y por tanto será el centro de la cruceta.

El siguiente fragmento de código muestra la creación de una imagen por objeto, la dilatación de los contornos y la suma final de las imágenes:

```
f = []
i=0
imgray2 = cv2.cvtColor(maskRGBOrange, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray2,255,255,255)
_,contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
                                         cv2.CHAIN_APPROX_SIMPLE)
\’areas = [cv2.contour\’area(c) for c in contours]
for extension in \’areas:
```

## CAPÍTULO 4. ALGORITMO

---

```
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]
```

4. A partir de la imagen final, se pasará un filtro que se quedará solo con los valores mayores a RGB(0,3,0), por lo tanto el único valor que no se eliminará sera el del centro de la cruceta.
5. Sobre la imagen obtenida, utilizando la función *drawcontours*, se obtienen la situación de las fronteras entre cuadrantes y de la cruceta, y con estos valores se puede marcar el centro de la baliza sobre la imagen real. Para visualizar en el GUI, se puede hacer que el filtro deje pasar valores mayores o iguales a RGB(0,2,0) obteniendo así las fronteras entre los cuadrantes, o los valores mayores o igual a RGB(0,1,0), obteniendo así los cuadrantes de los objetos. Al dejar pasar más objetos en el filtro y por tanto no ser tan robusto, puede tener en ocasiones falsos positivos.

## CAPÍTULO 4. ALGORITMO

---

En el siguiente fragmento de código se muestra como a partir de la imagen de la suma de objetos, se calcula la posición de la cruceta y se marcan sobre la imagen. En este caso, para que se obtuviera una imagen más clara, a los bordes de los objetos se les da un valor de 30, por lo tanto la intersección de 3 objetos tendrá un valor RGB(0,90,0) y de 4 objetos un valor RGB(0,120,0).

```
lower_green = np.array([0,80,0], dtype=np.uint8)
upper_green = np.array([0, 140,0], dtype=np.uint8)
maskSHI = cv2.inRange(show_image2, lower_green, upper_green)
show_image2 = cv2.bitwise_and(show_image2, show_image2, mask= maskSHI)

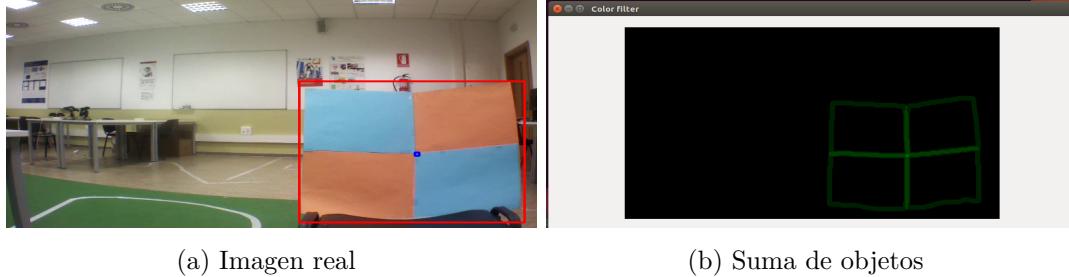
compare_image = np.zeros((y_img*2,x_img*2,3), np.uint8)
diff_total = cv2.absdiff(compare_image, show_image2)

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+alto),(0,0,255),2)
        positionX= (posicion_x+posicion_x+ancho)/2
        positionY= (posicion_y+posicion_y+ancho)/2
```

En la figura 4.5, se observa la imagen de entrada, en la cual esta marcado el centro de la baliza y el objeto de interés, y la suma de los bordes de los distintos

objetos, siendo los puntos donde más contornos se cruzan de un color más intenso.



(a) Imagen real

(b) Suma de objetos

Figure 4.5: Procesamiento de imagen de la baliza

### 4.3 Control

En esta sección se explica el control sobre el dron en función del momento del comportamiento en el que se encuentra y la imagen que se obtiene. Se puede dividir en tres partes: despegue, búsqueda y aterrizaje. La figura 4.6 indica el esquema de control que sigue el algoritmo, los estados que están en el mismo color, pertenecen a la parte de búsqueda, los otros dos, al despegue y aterrizaje:

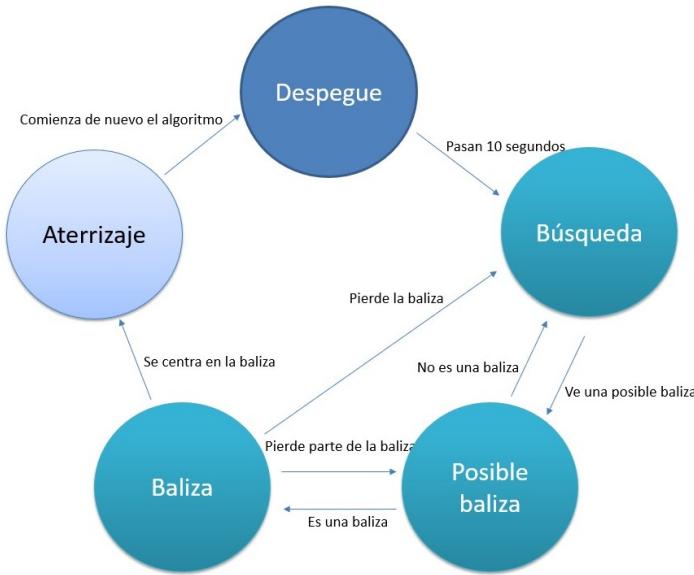


Figure 4.6: Esquema de control del algoritmo.

#### 4.3.1 Despegue

Esta fase está controlada por tiempo. Son los primeros diez segundos del algoritmo, y en ellos el dron despegue de forma controlada. Esta fase será ”Take off” en el autómata de estados de la figura 4.7. Para ello se sitúa el dron sobre una baliza sobre la cual tiene que estabilizarse. De esta forma, al despegar detecta ésta y trata de centrarse, evitando así que se desvíe por factores externos y quedándose en la situación correcta. Debido a que el dron va a despegar sobre la baliza, el control en esta parte es un control proporcional, para evitar que tenga que hacer múltiples operaciones, por tanto sea mas rápido el algoritmo, y suficiente para ser controlado.

#### 4.3.2 Búsqueda

Esta fase empieza cuando finalicen los diez segundos de despegue. El dron comenzará a navegar de forma autónoma en búsqueda de una baliza sobre la cual

## CAPÍTULO 4. ALGORITMO

---

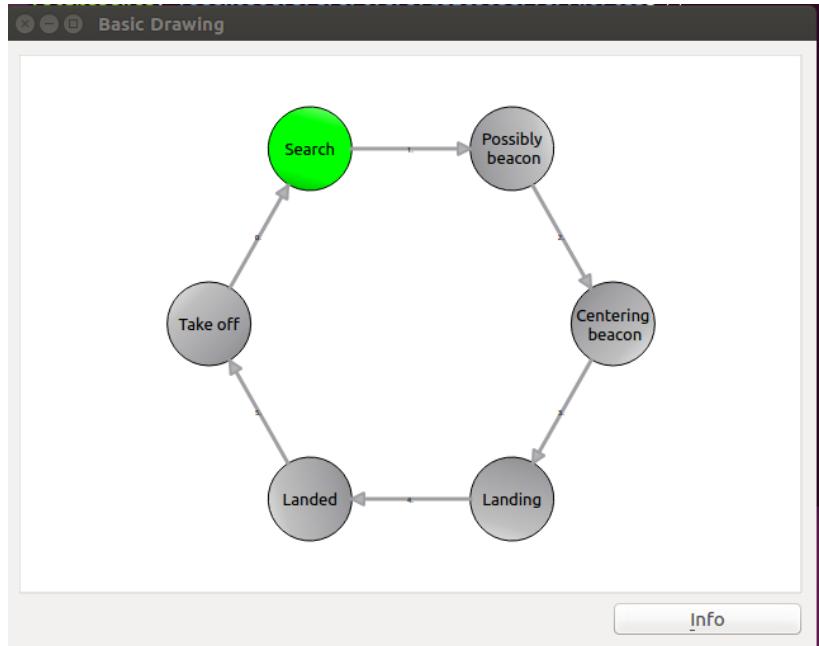


Figure 4.7: Diagrama de estados utilizado para el algoritmo.

aterrizar. Para ello sigue una navegación en espiral, estado “Search” de la figura 4.7, de forma que irá rastreando la zona ampliando su giro de forma continua, hasta que detecte una posible baliza. En el momento que la detecte, pasará al estado “Possibly beacon” de la figura 4.7 e intentará centrarse sobre ella. Si tras un número definido de iteraciones, no se detecta una bálica, continuará el algoritmo de búsqueda en el punto donde se había quedado, aumentando con la amplitud de las espirales a la que se había llegado. Pero en caso de detectarla pasará al estado “Centering beacon” de la figura 4.7, haciendo que coincida el centro de la baliza con el centro de la imagen de la cámara. Para realizar el movimiento de centrarse en la baliza se ha utilizado un control PD (proporcional y derivativo). Esto se debe a que sólo con el control proporcional, se producía una gran diferencia de velocidad y el drone cambiaba sus giros de forma muy brusca, lo que le llevaba a desestabilizarse y perder con facilidad la baliza. El algoritmo del control PD es de la siguiente forma:

## CAPÍTULO 4. ALGORITMO

---

- Por una parte, se tiene el control proporcional. Para éste se obtiene el centro de la cruceta o del objeto de interés. Se calcula la diferencia entre el centro de la imagen y el centro del objeto o la cruceta, obteniendo así el error, y como resultados, para el eje x  $\Delta_x$ , y para el eje y  $\Delta_y$ . Estos valores se multiplican por una constante que sirve para adaptar el resultado a la velocidad del drone. La fórmula de este control es la siguiente:

$$\Delta_x = \text{centroimagen}_x - \text{centroobjeto}_x$$

$$\Delta_y = \text{centroimagen}_y - \text{centroobjeto}_y$$

$$P_x = \Delta_x * 0.01$$

$$P_y = \Delta_y * 0.01$$

- Por otro lado, está el control derivativo. Para este caso, al tratarse de iteraciones, se trabaja con las diferentes muestras de cada iteración. Se obtiene el error de la iteración actual y de la iteración anterior para ambos ejes, y se resta el error anterior al actual:

$$\Delta_{diferenciax} = \Delta_x[n] - \Delta_x[n - 1]$$

$$\Delta_{diferenciay} = \Delta_y[n] - \Delta_y[n - 1]$$

$$D_x = \Delta_{diferenciax} * 0.003$$

$$D_y = \Delta_{diferenciay} * 0.003$$

Una vez se ha obtenido esto, se envían las instrucciones de velocidad al drone, utilizando el comando:

```
vely = (y_img-positionY)
velx = (x_img-positionX)

Kp=0.01
```

## CAPÍTULO 4. ALGORITMO

---

```
vy_P = vely*Kp
vx_P = velx*Kp

Kd = 0.003
vx_D = abs(xanterior-velx)*Kd
vy_D = abs(yanterior-vely)*Kd

Vy_tot = vy_P + vy_D
Vx_tot = vx_P + vy_D
self.cmdvel.sendCMDVel(Vy_tot,Vx_tot,0,0,0,0)

xanterior = velx
yanterior = vely

if(abs(Vx_tot-xanteriorTot)>0.3):
    if(Vx_tot<xanteriorTot):
        Vx_tot = xanteriorTot-0.3
    else:
        Vx_tot = xanteriorTot+0.3

if(abs(Vy_tot-yanteriorTot)>0.3):
    if(Vy_tot<yanteriorTot):
        Vy_tot = yanteriorTot-0.3
    else:
        Vy_tot = yanteriorTot+0.3
yanteriorTot = vytot
xanteriorTot = vxtot

self.cmdvel.sendCMDVel(Vy_tot,Vx_tot,0,0,0,0)
```

## CAPÍTULO 4. ALGORITMO

---

El siguiente fragmento de código consigue que el drone se mueva realizando espirales:

```
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
```

El esquema de este control cuando el drone está en el estado de búsqueda es el siguiente:



Figure 4.8: Esquema de control durante el algoritmo de búsqueda

### 4.3.3 Aterrizaje

Esta fase es un aterrizaje controlado. Una vez el drone se ha centrado sobre la baliza comienza a descender de forma constante , cambiando el estado a "Landing". Cuando área es menor a cierto umbral (en píxeles), en éste caso 19272135.0, se manda la instrucción de parar motores. En ese momento el drone desciende hasta posarse en el suelo, entonces para los motores y el estado cambia a "Landed". La razón de ir descendiendo poco a poco hasta detectar determinado área, es por si la baliza está lejos del drone, puede interferir algún objeto momentáneamente o darse algún factor externo que altere la posición del drone respecto de la baliza, y al enviar la instrucción

de parar motores se perdería este control. El umbral elegido es porque en ese momento la baliza ocupa prácticamente por completo la imagen, y si se le indicara al drone que siguiera descendiendo las corrientes que se producirían con el suelo podrían afectarle y desviar su trayectoria, pudiendo perder la referencia.

## 4.4 Arquitectura software de la implementación

El programa tiene que funcionar con fluidez, permitiendo enviar y recibir datos a la vez que se visualizan los datos que ya se tienen. Para ello, la aplicación PreciseLanding tiene una estructura que sigue el siguiente esquema:

Primero está el programa principal `precise_landing.py`. Éste es el encargado de crear la máquina de estados, decir el número de estados que va a tener y las posibles transiciones entre unos y otros. También se conecta con los sensores y actuadores a través de las interfaces ICE creando los manejadores respectivos (cámara, datos de navegación, datos de posición, comandos de velocidad y comandos extra para despegar y aterrizar) para pasárselos como parámetro a éste. Una vez se tenga todo definido, se pasa a la creación de los distintos hilos.

Por un lado se crea el interfaz gráfica y la ventana en la que se muestra el diagrama de estados. Se hace que éste se ejecute en un hilo y se le manda comenzar.

Para visualizar el diagrama de estados, se añadieron dos paquetes del GUI, 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)
```

La imagen del diagrama que veríamos durante la ejecución se encuentra en 4.7. La imagen cuando se abre la aplicación, es la que se encuentra en 4.9b

Por otro lado, se crea la ventana gráfica de control, que permite el control

## CAPÍTULO 4. ALGORITMO

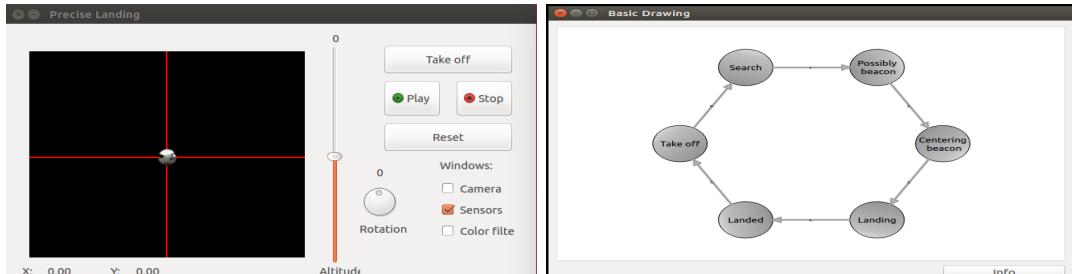
---

del drone y muestra la información de sus sensores. A éste se le pasan como datos los distintos manejadores anteriormente definidos para que pueda utilizarlas, como son la cámara para visualizar las imágenes o el *cmdvel* para enviar las ordenes de velocidad. Esta ventana también tiene los botones de "Play" y "Stop", los cuales permiten que ejecute el algoritmo creado o que pare su ejecución. Esta interfaz corresponde con la imagen 4.9a.

Para finalizar se llama a la función que ejecutará el algoritmo principal. A esta función se le pasan como parametros todo lo hasta ahora creado, para que así pueda acceder a los datos de la cámara, los distintos sensores y el diagrama de estados. Esta función permitirá obtener los datos deseados de los sensores, enviar instrucciones a los motores y al diagrama de estados. Este algoritmo, además de tener la función principal y las creadas para el funcionamiento de este, tiene cuatro funciones esenciales para su funcionamiento. Por un lado están las funciones de "play" y "stop" (asociadas a los botones respectivos de la ventana gráfica de control), las cuales mandan el evento de iniciar (en caso de la función Play), de parar (en caso de la función Stop), que se ejecutarán principalmente cuando se pulsen los botones del interfaz de control. Por otro lado está la función "kill", que se encarga de matar el proceso. Por último está la función "run", que es la encargada de llamar de forma continua al algoritmo a ejecutar mientras no se mate a la aplicación. Esta función, mientras se haya mandado la instrucción de iniciar y no la de parar, se encontrará en un bucle continuo obteniendo a final de cada iteración el tiempo que ha tardado en ejecutarse, en caso de no llegar a un mínimo de tiempo, la función parara la cantidad de tiempo necesaria para llegar a ese mínimo, y entonces dejará continuar con la ejecución. De esta forma, la velocidad de iteraciones no sólo dependerá del algoritmo y las funciones en si, sino que en caso de tardar muy poco tendrá determinados tiempos de espera.

## CAPÍTULO 4. ALGORITMO

---



(a) Precise Landing

(b) Diagrama de Estados

Figure 4.9: Interfaces que se abren al iniciar la aplicación.

# Chapter 5

## Experimentos

En este capítulo se van a comentar las distintas pruebas realizadas tanto con el drone real como en la simulación, las cuales validan cada fase del proyecto y la solución final. También estas pruebas han servido para depurar mientras se desarrollaba.

Para organizar las distintas pruebas, se va a hablar primero sobre la parte de percepción. Tras esto se comentarán por separado cada una de las partes del algoritmo de navegación (despegue, búsqueda y aterrizaje), para terminar con una ejecución típica del algoritmo completo.

En los experimentos realizados se han apreciado las diferencias importantes que hay entre trabajar en el entorno simulado y en el real. Primero, en el entorno simulado los colores son puros ideales y más fáciles de detectar, sin embargo, en el entorno real puede verse más o menos nítido debido a luces y sombras un color, además de ser más difícil aislarlo del resto. Segundo, las velocidades a los motores a aplicar sobre uno y otro son distintas, pues en el entorno simulado se le puede dar una mayor velocidad que trabajará sin problemas. Por el contrario, en el real una velocidad alta y un movimiento brusco pueden llevar a desestabilizarlo. Tercero, en el drone real interviene también la batería, en función de la carga que tuviera ésta el drone aterriza con más o menos fuerza y realiza movimientos más o menos fluidos. Por debajo

## CAPÍTULO 5. EXPERIMENTOS

---

de 30% de batería no permitía al drone realizar todos los movimientos, y en algunas ocasiones perdía altura, aunque después volvía a ganarla.

Los experimentos reales se han realizado con tres ArDrone2 de Parrot distintos, entre los que se han apreciado comportamientos distintos pese a ser el mismo modelo del mismo fabricante. Se debe a que alguno tenía más deriva que los otros o que los movimientos eran más bruscos según el drone.

### 5.1 Pruebas de percepción de la baliza

Para las pruebas de esta sección hay que tener en cuenta qué objeto detectar. Es un cuadrado con cuatro cuadrantes dentro, dos de color naranja, y otros dos de color azul (trabajando con el drone real) o verde (si se trabajaba en un entorno simulado).

Para verificar el funcionamiento de los filtros de color se realizaron tests en el simulador en los cuales se detectaban por separado los colores de la baliza y se contaba el número de objetos. Para una posible baliza se marcaba el objeto en el interfaz gráfico. En caso de que se detectara la cruceta de los 4 cuadrados, se señalaba ésta sobre la imagen y la baliza completa, como muestra la figura 5.1b.



Figure 5.1: Drone detectando objeto en escenario simulado.

Para realizar las pruebas en un entorno real se ponía el drone en una posición fija y se iba moviendo la baliza sobre distintas posiciones de la cámara y se comprobaba

## CAPÍTULO 5. EXPERIMENTOS

---

si las detectaba o no. Para observar lo que detectaba se marcaba en la interfaz gráfica en color azul el centro de la cruceta y con rojo los posibles objetos, coincidiendo con la baliza completa, como muestra la figura 5.2.

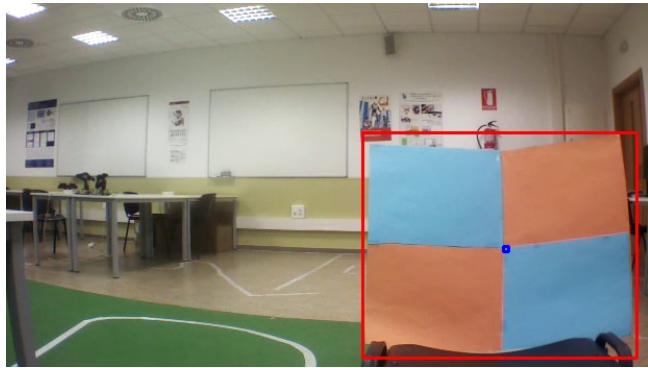


Figure 5.2: Detectando baliza real.

La percepción de la baliza es satisfactoria tanto en simulación como en escenarios reales, pues se ha conseguido una detección robusta. Apenas hay falsos negativos (no detectar una baliza existente), y tampoco falsos positivos (detectar como baliza algo que no lo es). Además, el algoritmo se ha probado en diferentes ambientes con distinta luminosidad.

## 5.2 Experimentos de despegue

Para verificar el despegue en un entorno simulado, las pruebas que se realizaron fueron fructíferas (Figuras 5.3 y 5.4). Esto se debe a que se trata de un entorno ideal en el cual el drone no tiene deriva ni se dan otros factores externos como las turbulencias. Las pruebas consistían en despegar el drone y que éste se centrara sobre la baliza, y despegar el drone sobre el coche con la baliza y mover el coche para ver que el drone le iba siguiendo. De este modo se probaba la capacidad de mantener al drone más o menos centrado sobre la baliza de despegue.

## CAPÍTULO 5. EXPERIMENTOS

---

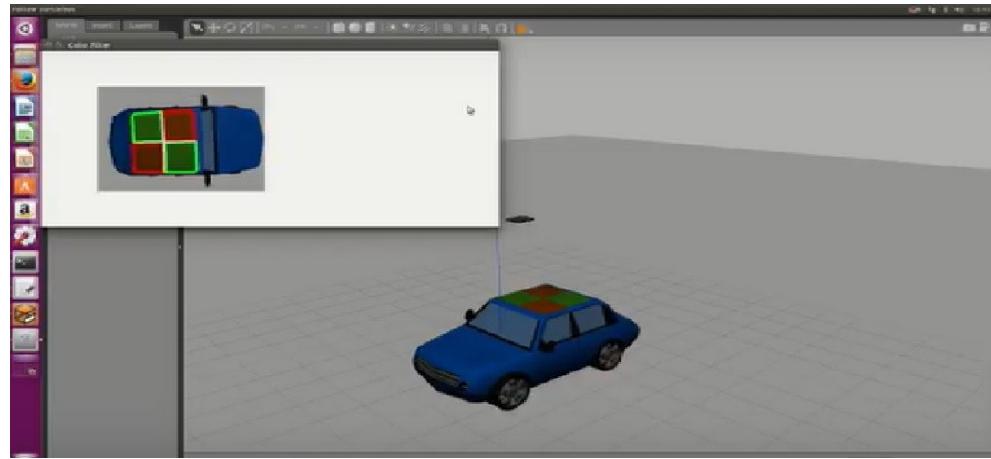
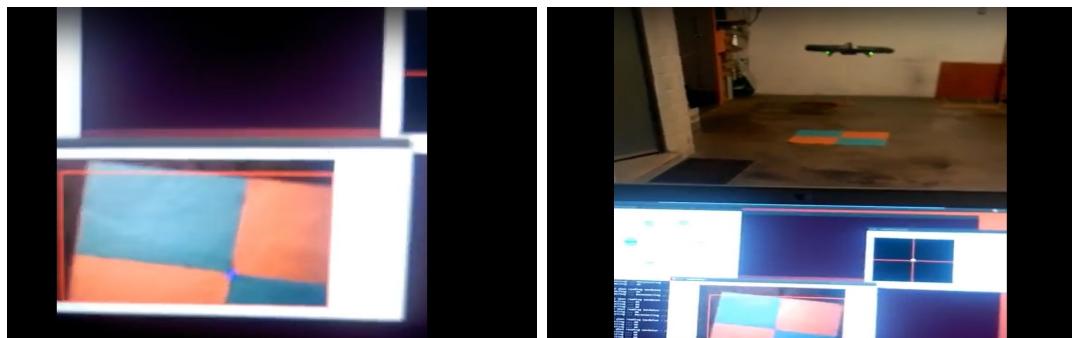


Figure 5.3: Despegue sobre la baliza del coche.

Para la realización de este experimento con un drone real se sitúa la baliza real sobre el suelo y el drone sobre ésta, y así al despegar tenía un punto sobre el que centrarse. La razón de programar un despegue controlado en vez de en lazo abierto se debe a que el drone real tenía una deriva inicial que le llevaba a desplazarse hacia atrás cuando tenía que estar en el sitio. El drone siempre despegaba en esta dirección y hay alrededor de dos segundos en los que no se pueden controlar sus movimientos, por tanto había que contar con este desplazamiento.



(a) Despegue vista baliza

(b) Despegue vista drone

Figure 5.4: Experimento de despegue

## CAPÍTULO 5. EXPERIMENTOS

---

Para la versión finalmente incorporada al algoritmo de navegación, la duración del periodo de despegue se ajustó en 10 segundos. El vídeo de un experimento ilustrativo del despegue del drone real está disponible en internet<sup>1</sup>. En él se puede apreciar un despegue con deriva y la corrección satisfactoria por el software de control desarrollado, de modo que la búsqueda de la baliza de destino se inicia más o menos centrado sobre la baliza de despegue, tal y como muestra también la figura 5.4.

### 5.3 Experimentos de búsqueda

Para la realización de esta parte del algoritmo de navegación, el drone va moviéndose en espiral hasta encontrar la baliza, para finalmente centrarse sobre ésta. Se han realizado diversos experimentos en simulación. Por una parte, escenarios donde el punto de despegue y el de aterrizaje se sitúan muy cerca, por tanto tiene que despegar, detectar la otra baliza y centrarse sobre esta. Por otra parte, escenarios situando lejos la baliza de despegue y de aterrizaje, observando así que el drone ampliaba su recorrido en cada vuelta de espiral que daba. Además, en ambas pruebas se han controlado las velocidades del drone para evitar movimientos bruscos.

En la figura 5.5 se observan fotogramas de la búsqueda de la baliza. En las dos primeras imágenes el drone realiza una búsqueda en espiral hasta que detecta una posible baliza, en la tercera imagen comienza a centrarse sobre ésta para finalmente darse cuenta de que es la baliza y se centra sobre ella, como se ve en las dos últimas imágenes.

---

<sup>1</sup><https://www.youtube.com/watch?v=HVGS1bA1tq4>

## CAPÍTULO 5. EXPERIMENTOS

---

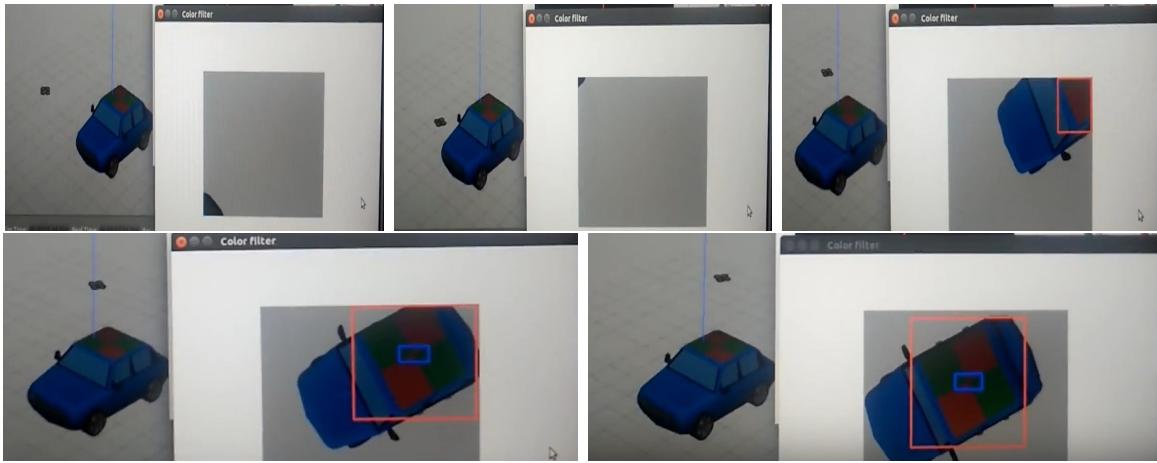


Figure 5.5: Búsqueda en espiral en escenario simulado

Para los experimentos del drone real se han realizado también dos pruebas principales. Por un lado, se han realizado varios tests detectando que el drone hacía espirales de forma correcta, ampliando la trayectoria de forma continua. Por otro lado, que al detectar la baliza no realizara movimientos bruscos para centrarse sobre esta. Para la realización de pruebas en lugares poco espaciosos se programó otra variante del algoritmo de búsqueda, pasando de hacer espirales a hacer cuadrados, para tener un movimiento más controlado del drone.

En la figura 5.5 se observa la búsqueda de la baliza con el drone en un escenario real. En este experimento el drone realizaba el algoritmo de búsqueda haciendo cuadrados, de forma que en las tres primeras imágenes se mueve realizando dicho algoritmo hasta que finalmente detecta la baliza y se centra sobre ésta, como se observa en las dos últimas imágenes.



Figure 5.6: Búsqueda en el escenario real

## 5.4 Experimentos de aterrizaje

En los experimentos de esta parte se trató de verificar que el drone se centrara sobre la baliza sin realizar movimientos bruscos. Una vez que el drone detectaba que estaba prácticamente centrado sobre la baliza, comenzaba a descender, y una vez detectaba que estaba lo suficientemente cerca enviaba la orden de parar.

En la figura 5.7 se puede observar el aterrizaje sobre el simulador en los distintos fotogramas, como una vez centrado en la primera imagen, el drone comienza a descender hasta finalmente posarse sobre la baliza.

## CAPÍTULO 5. EXPERIMENTOS

---

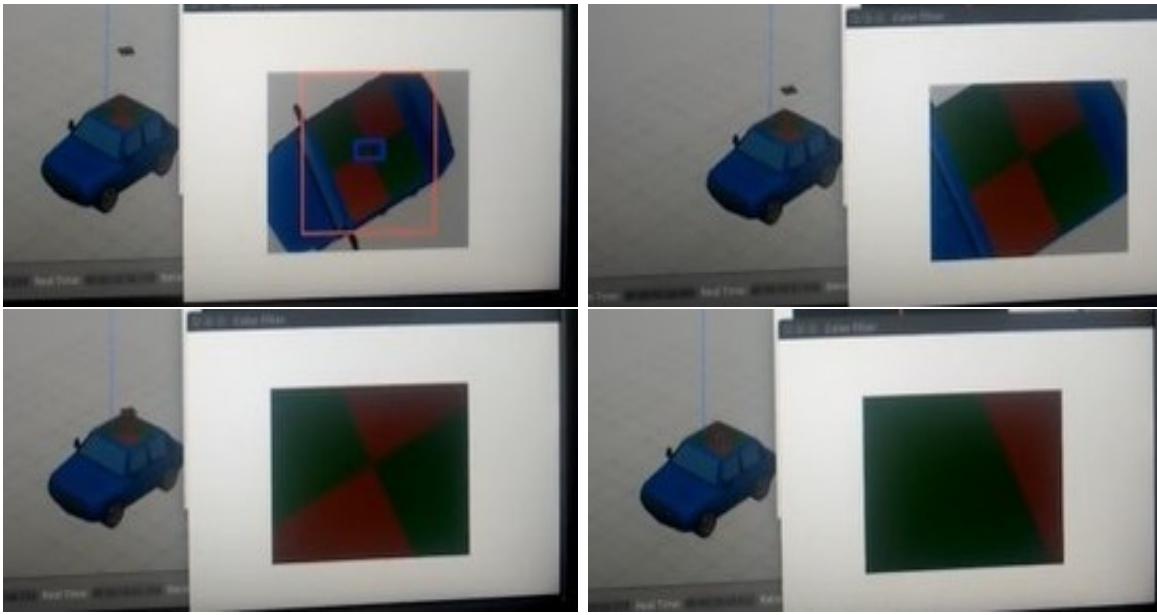


Figure 5.7: Experimento de aterrizaje en simulación

En la primera prueba, cuando el dron real estaba en el aire se ponía delante de la baliza de un color y al detectarla apagaba motores directamente. En la segunda prueba, cuando el dron estaba en el aire, en lugar de mostrar la baliza a la cámara frontal se le mostro a la cámara de abajo, probando así el funcionamiento de la cámara y que detectaba la baliza que se iba a utilizar. En la tercera prueba, mientras realizaba el algoritmo de búsqueda, una vez que detectara la baliza, aterrizaba sin más contemplaciones. En esta prueba los resultados eran buenos, ya que se conseguía que aterrizará, pero por la velocidad que tenía el dron y debido a la inercia, no aterrizaba en el sitio exacto sino que seguía en la misma dirección que tenía anteriormente hasta que se posaba en el suelo y ya se detenía, por lo tanto aterrizaba en las proximidades, pero relativamente lejos. Tras esto y al añadir el control para centrarse sobre ésta, se solucionó el problema y se consiguió que el dron aterrizará verticalmente sobre la baliza.

En la figura 5.8 se observa el aterrizaje del dron real. Tras haber realizado la

## CAPÍTULO 5. EXPERIMENTOS

---

búsqueda, una vez está centrado sobre la baliza, comienza el aterrizaje para finalmente posarse sobre ésta.

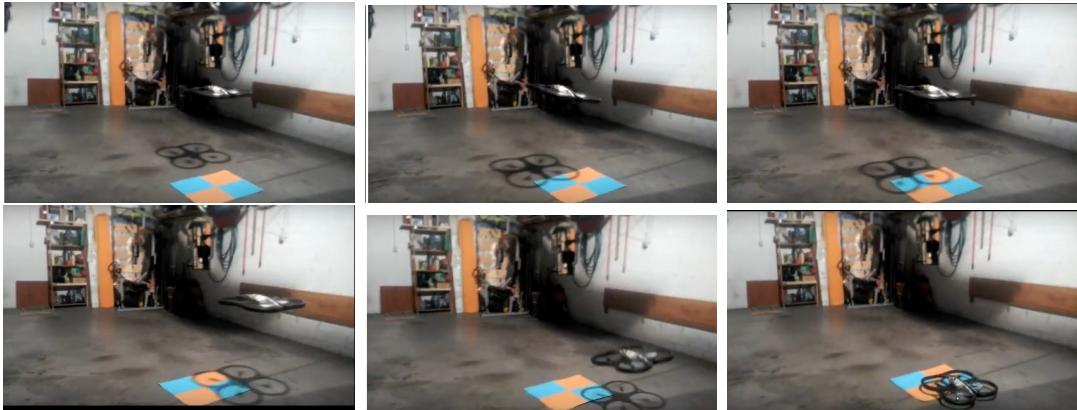


Figure 5.8: Experimento de aterrizaje en el drone real

## 5.5 Ejecución típica del algoritmo completo

Una vez validada cada una de las partes del algoritmo, tanto perceptivas como de control, se realizaron varios experimentos con todo el sistema integrado. Se llevaron a cabo para comprobar que todo lo que se había programado por partes funcionaba también si se probaba el algoritmo completo. Las pruebas salieron satisfactoriamente.

Primero, sobre el simulador se probó a poner el drone sobre la baliza, que despegaba y se situaba en el centro de ésta. Una vez pasaron 10 segundos y el drone continuaba en el centro, se hizo que éste se alejara y perdiera la referencia de la baliza, y comenzara su algoritmo de búsqueda. Una vez realizaba las espirales, en el momento que detectaba los colores de la baliza de destino se centraba sobre estos, y una vez estaba aproximadamente centrado, comenzaba a descender hasta que detectaba que estaba a una altura suficiente y se le podía mandar la orden de aterrizar, posándose así sobre la baliza.

En la figura 5.9 se muestra la secuencia de imágenes con el despegue y el drone

## CAPÍTULO 5. EXPERIMENTOS

---

alejándose de la baliza. Las imágenes de la búsqueda y el aterrizaje están en las figuras 5.5 y 5.7. El vídeo de la ejecución completa se encuentra disponible en la web <sup>2</sup>

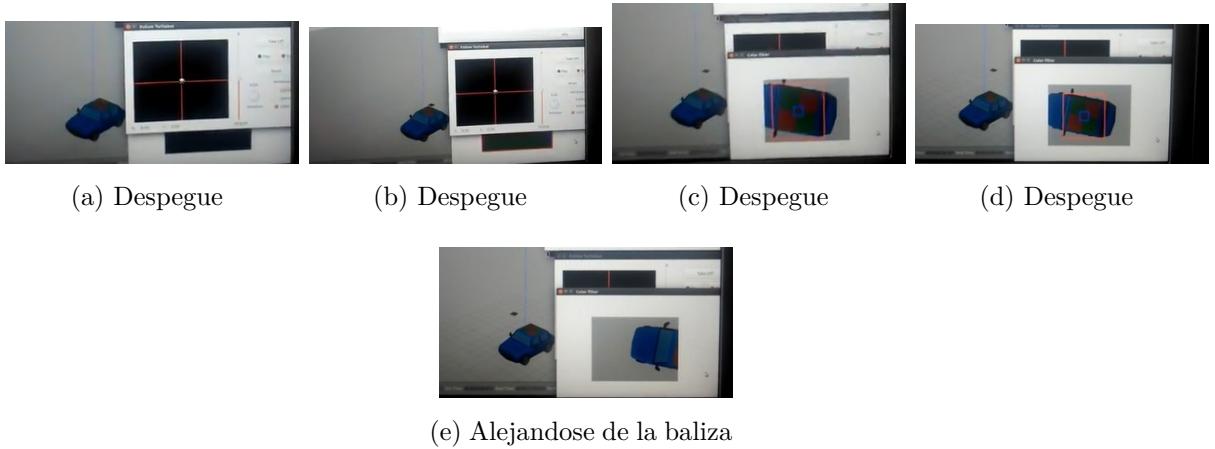


Figure 5.9: Algoritmo completo de navegación en escenario simulado.

Para las pruebas con el dron real se colocaron dos puntos de referencia en un pabellón deportivo. Una baliza sobre la que situarse al despegar y una baliza sobre la que aterrizar, en posición desconocida a priori. De esta forma al comenzar el algoritmo, cuando detectaba esta baliza se centraba sobre ésta, y una vez pasaron 10 segundos comenzaba el algoritmo de búsqueda. En este punto, el dron se movía en forma de espiral. En las siguientes imágenes de la figura 5.10 puede verse cómo en una primera vuelta de la espiral el dron no llega a ver la baliza de destino. Sin embargo, en la segunda vuelta de la espiral ya pasa sobre la baliza, detectándola y aterrizando sobre ella. El vídeo de la ejecución completa se encuentra disponible en la web <sup>3</sup>.

<sup>2</sup><https://www.youtube.com/watch?v=g9ZGJhRWTiY>

<sup>3</sup><https://www.youtube.com/watch?v=SkpuqEkoryY>

## CAPÍTULO 5. EXPERIMENTOS

---

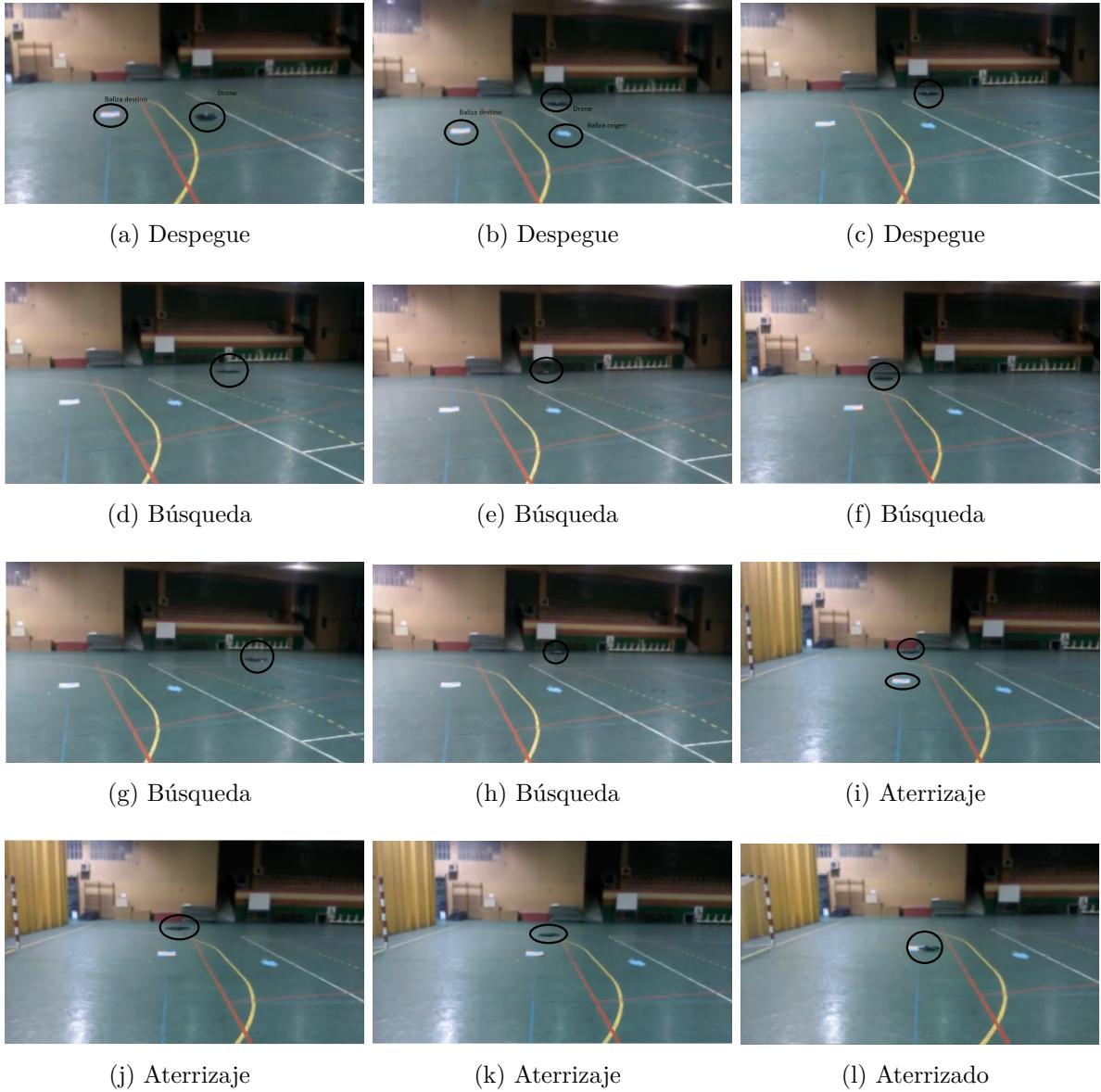


Figure 5.10: Algoritmo completo sobre el drone real.

# Chapter 6

## Conclusiones

Para finalizar, hay que realizar una evaluación del trabajo realizado, lo aprendido durante este periodo y los objetivos cumplidos. En un primer aspecto, podemos decir que el trabajo ha sido satisfactorio, se ha conseguido un algoritmo que desarrolle un despegue-búsqueda-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.

- Percepción: Se ha conseguido realizar un filtro de color para poder aislar los objetos interesantes del resto. Este objetivo se ha conseguido, trabajando principalmente con la librería OpenCV como hemos visto en la sección 3.4 . Destacar que en el simulador esto fue más sencillo debido a la nitidez de los colores y que en el drone real llevó más trabajo. Además se han tenido que hacer diversas pruebas debido a los distintos lugares donde se probaba el drone y la diferencia de colores que había debido a la luminosidad, pero al final se consiguió un buen filtro, que con ayuda de los operadores morfológicos (erosión, dilatación, cierre y apertura), nos permitían obtener una imagen de fondo negro y los objetos de los colores deseados en primer plano.

Una vez obteníamos los objetos de los colores deseados, ver si éstos eran

## CAPÍTULO 6. Conclusiones

---

los objetos que queríamos o no. Por una parte, se miraba el área que tenían las figuras, y en caso de no llegar a un tamaño predeterminado, éstas se descartaban como posibles objetos. Por otro lado, nos apoyamos en la forma de la baliza, pues los cuatro cuadrantes que la componían formaban una cruceta en su centro, la cual era la que se trataba de detectar, por lo que no se dependía sólo de unos colores determinados, sino también de una figura.

- Control: Se ha obtenido un comportamiento con tres fases principales:
  - Despegue: Al despegar, el drone detecta una baliza sobre la que situarse y se estabiliza sobre ésta, para obtener de esta forma un despegue controlado.
  - Búsqueda: En esta fase el algoritmo se desplaza en forma de elipse, recorriendo así la zona. Una vez que detecta un posible objeto trata de centrarse sobre éste.
  - Aterrizaje: Para finalizar, una vez se ha detectado la baliza y se ha centrado sobre ella, el drone comienza a descender, y una vez la baliza tiene un área determinado se envía la opción de aterrizar, terminando así el drone sobre la baliza.

En esta parte, a partir de la percepción, se han tenido que ajustar las velocidades, dependiendo si la prueba era para el drone real o para la simulación, como se ha contado en la sección 4.3 . Los ejemplos del control del algoritmo se pueden ver en las distintas secciones del capítulo 5 , destacando que el algoritmo completo se encuentra en la sección 5.5 , tanto para el drone real como para la simulación.

- Validación experimental: Cada parte del experimento se ha evaluado por separado a lo largo de éste, haciendo distintas pruebas para la detección de balizas, así como el control del drone una vez se detectaban éstas. Estas pruebas

pueden verse a lo largo del capítulo 5 , pero muchos de los avances y cambios que se han dado durante el desarrollo están disponibles en la wiki oficial del proyecto, indicada en el capítulo 2 . Aún con todo, destacar que trabajar en el drone real ha sido mucho más costoso de lo esperado,sobre todo para el control de velocidades, pues cualquier cambio pequeño en un valor suponía gran cambio en la realidad. Todo esto también fue importante para ver la diferencia que hay cuando tienes sólo un control proporcional y después le añades la componente derivativa.

Las distintas pruebas y los avances que se han ido dando durante el desarrollo, se han ido validando y están disponibles en la wiki oficial del proyecto:

<http://jderobot.org/Jvela-tfg>

Por otro lado, mirando los proyectos anteriores a éste en los que había trabajado RoboticsLabs URJC con drones, podemos ver que se ha conseguido algo diferente, ya que en este caso hemos conseguido que un drone navegue de forma autónoma guiándose por las balizas de color, consiguiendo que vaya de un punto a otro.

## 6.1 Líneas futuras

Es importante destacar que estos campo de la robótica y la visión se está produciendo un importante crecimiento, y añadiendo lo aprendido en el trabajo y como se puede trabajar en él creo que sería interesante continuar la línea de este proyecto para continuar con el desarrollo y aprendizaje de estas técnicas. Algunas aplicaciones podrían ser las siguientes.

- 1<sup>a</sup> Probar en exteriores con otro drone diferente, el 3DRSolo drone.
- 2<sup>a</sup> Probar trayectorias más lejanas donde la parte intermedia vaya recurriendo puntos GPS, y sólo en la parte final, ya cerca de las coordenadas del destino se active la búsqueda visual de la baliza de aterrizaje y el propio aterrizaje.

## CAPÍTULO 6. Conclusiones

- 3<sup>a</sup> Incorporar autolocalización visual al drone para elaborar una estrategia de búsqueda más elaborada y de mayor amplitud.

# Bibliography

- [1] Alberto Martín Florido. Navegación visual en un cuadricóptero para el seguimiento de objetos. *Proyecto Fin de Carrera, URJC*, 2014.
- [2] Arturo Vélez. Seguimiento de un objeto con textura desde un drone con cámara. *Trabajo de Fin de Grado, URJC*, 2017.
- [3] MediaWiki Manuel Zafra Villar. Seguimiento de rutas 3D por un drone con autolocalización visual con balizas. *Proyecto Fin de Carrera, URJC*, 2016.
- [4] Daniel Yagüe. Cuadricóptero ar.drone en gazebo y jderobot. *Proyecto Fin de Carrera, URJC*, 2015.
- [5] Jorge Cano. Building of an uav: from the hardware to the driver and autonomous applications. *Trabajo de Fin de Grado, ETSIT-URJC*, 2016.
- [6] Página oficial de JdeRobot. [http://jderobot.org/Main\\_Page](http://jderobot.org/Main_Page).
- [7] Página oficial de OpenCV. <https://opencv.org/>.
- [8] Página oficial de Python. <https://www.python.org/>.
- [9] Página oficial de Parrot. <https://www.parrot.com/es/>.
- [10] Página oficial de Gazebo. <http://gazebosim.org/>.
- [11] Página oficial de Pixhawk. <https://pixhawk.org/>.
- [12] Página oficial de ArduPilot. <http://ardupilot.org/ardupilot/index.html>.
- [13] Página de xdrones. Protocolo MavLink. <http://www.xdrones.es/mavlink/>.

## BIBLIOGRAFÍA

---

- [14] Darpa Robotics Challenge. <https://www.darpa.mil/program/darpa-robotics-challenge>.
- [15] Página de OpenCV. Transformaciones morfológicas.. [https://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_imgproc/py\\_morphological\\_ops/py\\_morphological\\_ops.html](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html).
- [16] Página de ROS de sus avances con los drones. <http://www.ros.org/news/robots/uavs/>.
- [17] Página de Dronica. Información reparto de paquetes con drones. <http://www.dronica.es/las-patentes-de-amazon-y-el-drone-del-futuro/>.
- [18] Página de El Mundo. Información reparto de paquetes con drones. <http://www.elmundo.es/tecnologia/2016/12/29/5864da5a22601dfb658b45bb.html>.
- [19] Drone para el inventario de los almacenes en MIT. <https://www.inverse.com/article/35924-new-mit-drone-system-rfly-us-retailers-rfid>.
- [20] Página de El drone. Historia de los drones. <http://eldrone.es/historia-de-los-drones/>.
- [21] Página de tododrone. Uso de drones en Walmart. <http://www.todrone.com/drones-dentro-de-tiendas/>.
- [22] Página de GitHub de la práctica choca-gira. [https://github.com/JdeRobot/Academy/tree/master/src/bump\\_and\\_go\\_py](https://github.com/JdeRobot/Academy/tree/master/src/bump_and_go_py).
- [23] Página de GitHub de la práctica follow\_turtlebot. [https://github.com/JdeRobot/Academy/tree/master/src/follow\\_turtlebot](https://github.com/JdeRobot/Academy/tree/master/src/follow_turtlebot).