

ñno4.130Diseñosection.4.1



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. Los UAV se comenzaron a utilizar hace muchos años con fines bélicos, y poco a poco su desarrollo ha permitido que se utilicen en ámbitos muy diferentes, como la robótica, gracias, por ejemplo, al comportamiento autónomo de éste, trabajando también en una navegación en interiores.

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

El algoritmo programado se ha validado experimentalmente, tanto en un drone simulado (en 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 está accesible públicamente.

# Contents

|                          |   |           |
|--------------------------|---|-----------|
| <b>1</b>                 | <b>Introducción</b>   | <b>1</b>  |
| 1.1                      | Historia de los drones . . . . .                                | 1         |
| 1.2                      | Aplicaciones actuales . . . . .                                 | 3         |
| 1.2.1                    | Medios Audiovisuales . . . . .                                  | 4         |
| 1.2.2                    | Seguridad . . . . .   | 4         |
| 1.2.3                    | Sector agrícola . . . . .                                       | 5         |
| 1.2.4                    | Inspección . . . . .  | 6         |
| 1.2.5                    | Emergencias . . . . .   | 7         |
| 1.2.6                    | Proyectos de empresas . . . . .                                 | 7         |
| 1.3                      | Hardware . . . . .  | 10        |
| 1.4                      | Software . . . . .  | 15        |
| 1.5                      | Robótica aérea en el laboratorio de la URJC . . . . .           | 18        |
| <b>Índice de figuras</b> |   | <b>1</b>  |
| <b>2</b>                 | <b>Objetivos</b>  | <b>20</b> |
| 2.1                      | Objetivo principal . . . . .                                    | 20        |
| 2.2                      | Requisitos . . . . .  | 21        |
| 2.3                      | Metodología . . . . .   | 21        |
| 2.4                      | Plan de trabajo . . . . .                                       | 22        |
| 2.4.1                    | Formación . . . . .   | 23        |
| 2.4.2                    | Desarrollo de la percepción robusta de la baliza . . . . .      | 23        |
| 2.4.3                    | Desarrollo de control visual de aterrizaje y despegue . . . . . | 23        |
| 2.4.4                    | Desarrollo del control de búsqueda . . . . .                    | 23        |

|   |           |
|---|-----------|
| <b>3 Infraestructura utilizada</b>                    | <b>24</b> |
| 3.1 Parrot AR.Drone 2.0 . . . . .                     | 24        |
| 3.2 Simulador Gazebo . . . . .                        | 25        |
| 3.3 Entorno JdeRobot . . . . .                        | 26        |
| 3.3.1 Driver ArDroneServer . . . . .                  | 26        |
| 3.3.2 Plugin modelo de cuadricoptero Gazebo . . . . . | 27        |
| 3.3.3 Herramienta UAVviewer . . . . .                 | 27        |
| 3.3.4 Herramienta ColorTuner . . . . .                | 27        |
| 3.3.5 JdeRobot Academy . . . . .                      | 28        |
| 3.4 Biblioteca OpenCV . . . . .                       | 29        |
| <b>4 Algoritmo</b>                                    | <b>30</b> |
| 4.1 Diseño . . . . .                                  | 30        |
| 4.2 Percepción . . . . .                              | 32        |
| 4.2.1 Pre-procesado . . . . .                         | 33        |
| 4.2.2 Detección de la baliza . . . . .                | 35        |
| 4.3 Control . . . . .                                 | 38        |
| 4.3.1 Despegue . . . . .                              | 38        |
| 4.3.2 Búsqueda . . . . .                              | 39        |
| 4.3.3 Aterrizaje . . . . .                            | 40        |
| <b>5 Experimentos</b>                                 | <b>42</b> |
| 5.1 Percepción . . . . .                              | 43        |
| 5.2 Despegue . . . . .                                | 44        |
| 5.3 Búsqueda . . . . .                                | 45        |
| 5.4 Aterrizaje . . . . .                              | 47        |
| 5.5 Algoritmo completo . . . . .                      | 49        |

|                              |           |
|------------------------------|-----------|
| <b>6 Conclusiones</b>        | <b>51</b> |
| 6.1 Líneas futuras . . . . . | 53        |

# 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 . . . . . | 5  |
| 1.3 | Empresa Novadrone utiliza drones para la gestión de las explotaciones agrícolas . . . . .             | 6  |
| 1.4 | Desarrollan en la universidad de Holanda un dron que lleva incorporado un desfibrilador . . . . .     | 7  |
| 1.5 | Drone detectando código RFID. . . . .   | 9  |
| 1.6 | Esta imagen muestra los movimientos que tiene un dron. . . . .  | 10 |
| 1.7 | Distintas partes del drone. . . . .   | 15 |
| 3.1 | Mundo Gazebo. . . . .   | 26 |
| 3.2 | UAV viewer . . . . .  | 27 |
| 3.3 | Diagrama de estados. . . . .  | 28 |
| 3.4 | Follow Turtlebot. . . . .   | 28 |
| 3.5 | UAV viewer . . . . .  | 29 |
| 4.1 | Baliza utilizada para el drone real. . . . .  | 32 |
| 4.2 | Procesamiento de imagen de la baliza . . . . .  | 38 |
| 5.1 | Drone detectando objeto. . . . .  | 43 |
| 5.2 | Detectando baliza real. . . . .   | 44 |
| 5.3 | Despegue . . . . .  | 45 |
| 5.4 | Busqueda . . . . .  | 46 |
| 5.5 | Busqueda . . . . .  | 47 |

|     |  |    |
|-----|--|----|
| 5.6 | Aterrizaje . . . . .                           | 48 |
| 5.7 | Aterrizaje . . . . .                           | 49 |
| 5.8 | Algoritmo completo sobre el simulador. . . . . | 50 |

# Chapter 1

## Introducción

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

### 1.1 Historia de los drones

Los drones son conocidos como UAV (vehículos aéreos no tripulados). El primer registro de UAV se trata de un globo aeroestático en un entorno militar en el año 1849, que se podía utilizar para sobrevolar una zona y lanzar bombas desde cierta altura sin

## CAPÍTULO 1. INTRODUCCIÓN

---

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, pero en ello ya profundizaremos más adelante). Estos vehículos eran aviones tripulados por radiofrecuencia, por lo que se dió 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, 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 los inicios de estos tenían sólo fines militares y que su desarrollo era exclusivamente para ello. En comparación con estos datos, el avance sobre estos frenó en gran medida y ya lo que se hacía era modificaciones para poder dar uso a lo que ya había, pues se utilizaban para vigilancia aérea en zonas de conflictos, lo que llevó a mejorar el sistema de control haciendo así que se pudieran manejar a una mayor distancia.

Fue alrededor de 1980 cuando se vió que la tecnología y el software de los UAV eran de gran fiabilidad y se podían asignar a estas tareas de mayor responsabilidad para no jugarse la vida de los pilotos. 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 aguantarlos el

## CAPÍTULO 1. INTRODUCCIÓN

---

cuerpo humano, ahora podían hacerlos con la brusquedad que permitiera el sistema.

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

Con todo esto, cabe destacar el gran avance que ha sufrido la robótica aérea. Esto se debe a que, 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 cualidades como 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

Teniendo ya el contexto de las distintas características que tiene un drone, se va a proceder a describir los distintos usos que se le pueden dar a estos. Se ven en la sociedad en general para terminar viendo usos mas específicos de la robótica aérea, 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 a que el precio de estos es asequible, por lo que se puede acceder a ellos con facilidad, y en un sector tan amplio y vistoso como es este, lleva a un uso cada vez más común.



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

### 1.2.2 Seguridad

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

## CAPÍTULO 1. INTRODUCCIÓN

---



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

### 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 un buen control 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, este permite ver con facilidad el estado de la cosecha, detectar enfermedades o plagas, permiten fumigar desde el aire con mayor precisión, ya que les puedes programar una ruta y que sigan ésta. También podrían obtener otros datos como las zonas con mas 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 podemos englobar diversas actividades, como puede ser mantenimiento de edificios y construcciones, redes eléctricas o diversas instalaciones industriales como aerogeneradores eólicos y estados de paneles solares. Al igual que en el apartado anterior, aquí podemos 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 podríamos observar si todas las placas están en las condiciones óptimas y en caso de existir algún fallo poder identificarlo con facilidad. Para edificios y aerogeneradores lo que hay que tener en cuenta es la altura que pueden tener, y a la cual con un drone llegaríamos con facilidad y observaríamos si hay algún problema. En estos casos lo que evitamos, como anteriormente he comentado, es que alguien tenga que ir sitio a sitio perdiendo mucho tiempo. Destacando también que este tipo de actividades se pueden implementar programas que directamente detecten las anomalías, sin necesidad de que haya una persona revisando en todo momento las imágenes, donde también con una inversión inicial, al final ahorraríamos mucho tiempo y dinero. Un ejemplo de ésto lo podemos ver en la empresa Iberdrola, la cual ha incorporado drones para el mantenimiento e inspección de infraestructuras, utilizando drones por ejemplo para el mantenimiento de las palas de los aerogeneradores. Unión Fenosa también ha

## CAPÍTULO 1. INTRODUCCIÓN

---

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 Sídney, que puede llevar materiales quirúrgicos o plasma sanguíneo, por ejemplo.



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

### 1.2.6 Proyectos de empresas

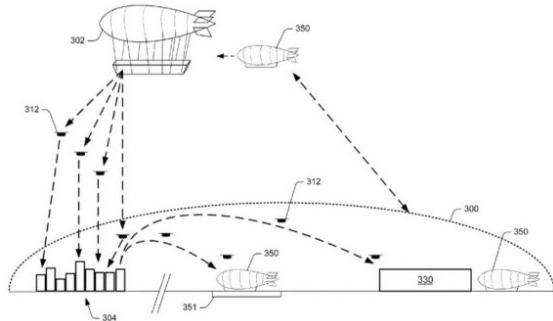
Es muy importante este ámbito, pues determinados grupos y grandes empresas están trabajando en su desarrollo debido al futuro que se viene por delante. Esto se debe a la facilidad que puede llevar esto para realizar grandes tareas, como el control de material en almacenes o transporte de material de un lugar a otro. Una característica importante es la rapidez con la que pueden llegar los drones de un lugar a otro, y acceder a lugares que es difícil por otros vehículos. De momento las distintas empresas que están integrando esta tecnología en el uso diario, no tienen todavía proyectos

## CAPÍTULO 1. INTRODUCCIÓN

---

terminados que realicen todas las tareas, pero si tienen distintos prototipos o proyectos piloto, de los que se pueden destacar algunos casos.

Por un lado, como pionero en este área se encuentra **Amazon**, el cual lleva desarrollando desde 2013 una tecnología que permite 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.



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 en una universidad de reparto de comida, 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, como puede ser el de tener controlados los paquetes dentro de un almacén.

## CAPÍTULO 1. INTRODUCCIÓN

---

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 este pueda tener controlados los paquetes que hay y donde están situados.



Figure 1.5: 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 con esto, es que en caso de que los empleados estén ocupados, un cliente no tenga que estar a la espera, sino que con un dispositivo podamos pedir el producto y un dron se encargará de ir a por él y traerlo al punto donde nos encontraremos. Destacar que se incorporaron en las tiendas controladores aéreos para que los vehículos sigan una trayectoria segura.

## 1.3 Hardware

Hay que destacar las partes que tiene un drone y su forma, pues es gran parte lo que lo hace tan especial, permite que tenga una gran libertad de movimientos, ya que puede moverse sin problema desde cualquier punto hacia los ejes X, Y y Z. Lo que ganamos con esto son cosas como poder permitirse un aterrizaje y un despegue totalmente vertical, sin depender de un espacio en el que coger velocidad para poder levantar el vuelo, e igual con el aterrizaje, pudiendo el drone estando quieto en el aire bajar totalmente en vertical hasta tocar 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 movimientos Roll, Yaw y Pitch y sin necesidad de hacer movimientos bruscos. Sin embargo, en los anteriores UAV solo tenemos el movimiento hacia adelante, teniendo que jugar con Roll, Yaw y Pitch para poder movernos en los distintos ejes.

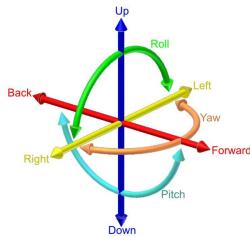


Figure 1.6: Esta imagen muestra los movimientos que tiene un dron.

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

**Frame:** También conocido como marco, estructura o chasis. Es la estructura principal sobre la que se sitúan el resto de los elementos. Cambiará su forma dependiendo del drone, variando la longitud de las patas o el número de soportes para hélices, por ejemplo. Esta puede estar hecha por diversos materiales, generalmente se trata de algún tipo de plástico, ya que es un material que tiene poco coste y pesa poco. Un ejemplo es el polipropileno, que es ligero y con mucha resistencia, lo que permite

## CAPÍTULO 1. INTRODUCCIÓN

---

colocar sobre el la batería. Otro material que suele utilizarse es la fibra de carbono, ya que se trata de un material que pesa poco y es muy resistente, aunque puede tener factores negativos como su conductividad. Por ultimo, también nombrar la fibra de vidrio. Este material también es muy utilizado por ser ligero, y tiene características como que no es conductor de la electricidad. Es común ver estructuras híbridas entre distintos materiales, sobre todo juntando los dos tipos de fibra.

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

**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. Este, a su vez, tiene distintos parámetros que serán principalmente los que permitan al drone llevar mayor velocidad.

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

El tamaño que éste tenga. Al mirar las especificaciones de un drone está en un numero 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 hacer 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, 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 es importante la tasa de descarga, que se trata de la máxima energía que puede entregar

## CAPÍTULO 1. INTRODUCCIÓN

---

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 se trata de radiofrecuencia o de Wifi. Existen casos, como el modelo 3DR, que combina ambas tecnologías, utilizando la radiofrecuencia para la información del movimiento, batería y posicionamiento, y el WiFi para la transmisión de imágenes en directo. Podemos 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. La principal característica de éste es que no pierde información cuando no hay contacto directo entre las dos estaciones. Pero vía WiFi es algo muy utilizado en los últimos momentos, pues permite controlar el dron desde una aplicación móvil, por lo que conectando estos dos tendríamos un mando que nos permite cambiar gran parte de la configuración del drone. También existen dispositivos que permiten el control mediante Bluetooth, pero éste 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. Es básicamente el hardware que utiliza el drone. Hay una gama muy amplia dentro de éste, donde cabe destacar **Pixhawk**, pero podemos encontrar varias con gran trascendencia:

- Pixhawk: Es el más utilizado debido a que trabaja con 3DRobotics y Ardupilot. Sirve para diversos dispositivos como son drones, helicópteros y barcos. Está

## CAPÍTULO 1. INTRODUCCIÓN

---

pensado para cualquier vehículo que tenga movimiento. Se trata de un proyecto hardware abierto, cuyo objetivo principal es proporcionar el hardware de autopiloto a comunidades académicas o gente que tiene esto como un hobby, teniendo así un bajo costo y una alta disponibilidad. Se trata de un piloto automático en tiempo real y muy eficiente, proporcionando un entorno de estilo POSIX. Éste es el autopiloto estándar de la industria, y por lo tanto, como veremos a continuación, a partir del cual se han desarrollado diversos autopilotos con distintas mejoras.

- Pixhawk2: Es una versión avanzada de la placa anterior. Tiene mejoras como aislamiento de vibraciones, 3 IMUs para redundancia (3 acelerómetros, 3 giroscopios, 3 magnetómetros y 2 barómetros) y sensor para controlar la temperatura.
- PixRacer: Éste se ha desarrollado para los drones de carreras, aunque también se utiliza en minidrones. Suele tener una mayor memoria flash.
- Navio2: Piloto automático diseñado de Raspberry Pi. Te permite convertir esta en un controlador de drone.
- PXFmini: Se trata de otro piloto automático de Raspberry Pi. Éste tiene la electrónica para la mayoría de los componentes que puede utilizar un dron.
- FlytPOD: Se trata de una placa Odroid XU4 SBC junto con una PixHawk. Puede volar diversos vehículos aéreos y su principal característica es el WiFi que tiene integrado. Existe una placa FlytPOD pro que se trata de una versión extendida de la anterior, teniendo todas sus características, pero con 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

## CAPÍTULO 1. INTRODUCCIÓN

---

automática. Su radioenlace con frecuencia en torno a 900Mhz permite controlar el dispositivo a una distancia de 100km.

Hay que destacar un elemento importante como es la **cámara**, que aunque no todos los drones la llevan sí que es algo 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 poder incorporar ciertas cámaras, normalmente consideradas camaras de acción, para así obtener una mejor calidad, e incluso incorporar adaptadores como puede ser una Gimbal para controlar la parte hacia la que queremos que apunte la 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. Cabe destacar que en muchas ocasiones, utilizado normalmente para carreras de drones, la cámara sirve para integrar la tecnología FPV (First Person View), que es junto a la cámara, el transmisor de vídeo y el receptor de vídeo, poder ver en tiempo real las imágenes sobre una pantalla LCD o utilizando unas gafas de realidad virtual. En este aspecto, en los últimos años se ha visto por otro lado un gran avance de la realidad virtual, y es también hay una gran variedad en este mundo, pues existe una gama que va desde las *cardboard*, que permiten con un trozo de cartón y un par de lentes, poniéndolos de cierta forma y con el uso de un smartphone, tener de forma sencilla unas gafas 3D, hasta gafas para la videoconsola que te permiten entrar en el videojuego, añadiendo una gran calidad de imagen y con un sonido envolvente para entrar de lleno en el ambiente. Pero una de las cosas mas 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 dron, y con cualquier movimiento que sientan las gafas la cámara del drone lo imitará. En caso de que esto parezca incómodo el drone tiene un joystick con el que enviará la información de los elementos a realizar a la cámara.



Figure 1.7: Distintas partes del drone.

## 1.4 Software

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

- **Ardupilot:** Es un sistema OpenSource encargado de recibir la información que se le da y de esta forma enviar las señales correspondientes a los actuadores. Se trata del software más importante debido a lo completo que es y confiabilidad que proporciona, debido a la gran cantidad de gente que lo utiliza (pilotos de drones profesionales y aficionados) y por el equipo de ingenieros que lo han desarrollado. Este software se caracteriza por la variedad de dispositivos que puede llegar a controlar, ya que trabaja con diversos dispositivos aéreos

## CAPÍTULO 1. INTRODUCCIÓN

---

(aviones, helicópteros, drones, etc) y con dispositivos marinos (como pueden ser los barcos y submarinos). Éste ha tenido un gran desarollo debido a su principal característica: OpenSource. Hay mucha gente creando interfaces para este y dichos usuarios comparten sus avances con el resto. A partir de éste han nacido controladores como Ardupilot Mega. El problema que tiene dicho software es que sólo permite trabajar con plataformas de los mismos creadores, lo que lleva al siguiente software. Otra característica es la facilidad con la que se le pueden añadir diferentes sensores, como pueden ser modulos GPS o cámaras, algo que facilitará la navegación autónoma.

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

Estos primeros son los softwares principales que estarán 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 este nos envía. Normalmente el fabricante del drone tiene ya un programa que realiza esta función.

En este punto es importante el protocolo de comunicación que habrá para comunicarse el vehículo con la estación terrena. Aquí hay un protocolo que destaca sobre los demás, el **MAVLink** (Micro Air Vehicle Communication Protocol). Éste protocolo tiene la información contenida en ficheros .xml, lo que permite utilizarlo en

## CAPÍTULO 1. INTRODUCCIÓN

---

diversos lenguajes de comunicación, lo que conlleva una mejora notable en su desarrollo. Al tener el fichero .xml los tipos de mensaje, permite con facilidad añadir nuevos tipos para asignar una tarea nueva a cada uno. Otra ventaja de este es que hay muchos software de drones que lo soportan, como pueden ser Ardupilot, Autopilot, algunos derivados de estos y otros como Gentlenav o Flexipilot, y desde la estación tierra algunos como MAVProxy, Mission Planer o APM planner. Un problema en este protocolo es que los datos no están encriptados en la comunicación, por lo que es más fácil un ataque y que se manipulen los datos, siendo detectable si se hace perder algún dato ya que se 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 poder acceder a los datos del vehículo, como la velocidad y las imágenes, lo que nos permitirá también saber qué datos mandarle para que funcione de forma correcta.

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

## 1.5 Robótica aérea en el laboratorio de la URJC

Es importante destacar que el inicio de este proyecto se ha podido dar gracias a otros proyectos realizados en el laboratorio de la universidad, en los que se hicieron grandes avances en el área de la robótica aérea y que nos han llevado a centrarnos en el caso de la situación por visión mediante balizas. Algunos casos de otros trabajos son los siguientes:

Alberto Martín trabajó en la navegación visual para el seguimiento de objetos. En este proyecto el drone trabajaba en los ejes X,Y y Z. Teniendo un objeto de determinadas características el drone era capaz de seguirlo, funcionando tanto en el eje vertical como en el horizontal. También era capaz de cambiar la orientación del drone en función de la situación del objeto.

Daniel Yagüe trabajó en con el ArDrone sobre gazebo, probando distintas funcionalidades y distintos escenarios. En este proyecto, también mediante control visual, realizó diversas pruebas como seguir líneas, seguir otro drone o seguir un recorrido mediante April Tags.

Manuel Zafra trabajo en la localización del Drone mediante April Tags. En este proyecto el drone iba detectando las diferentes April Tags que había situadas en los distintos lugares y gracias a ello sabía la posición en la que se encontraba y los movimientos a realizar.

Arturo Velez realizó un proyecto en el cual el drone seguía un objeto dada una textura y la autolocalización a partir de ésta. Si no se encontraba la textura el drone realizaba una búsqueda. También trabajó con April Tags y la autolocalización en función de lo que detectaba.

Por último nombrar el proyecto de Jorge Cano, quizás es el mas distinto a los anteriores y a lo que se ha trabajado aquí, ya que se centraba más en el hardware del drone, especificando los distintos materiales y su configuración con el software. Destacar que también realizó pruebas de vuelo para las cuales tuvo que trabajar en las

## CAPÍTULO 1. INTRODUCCIÓN

partes perceptiva y de control.

Una vez terminada la breve introducción, se va a explicar lo realizado en este proyecto. El orden seguido ha sido el siguiente:

En el capítulo 2 se redactan los objetivos propuestos y la metodología seguida para llegar hasta ellos.

En el capítulo 3 se redacta la infraestructura utilizada y como se ha trabajado con ella.

En el capítulo 4 se redacta el algoritmo realizado, como se ha programado y porque se han seguido unos u otros caminos.

En el capítulo 5 se redacta los distintos experimentos realizados y los resultados que se han obtenido.

Por ultimo, en el capítulo 6 se redactan las conclusiones a las que se han llegado tras finalizar este proyecto.

# 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 navegará 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 estos no sean bruscos, provocando desestabilizaciones del drone.

- **Validación experimental:** Cada vez que se dá 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:

- 1<sup>a</sup> Que funcione sobre JdeRobot-5.6, utilizando las herramientas necesarias para el control del drone, filtros de color y diagrama de estados.
- 2<sup>a</sup> 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.
- 3<sup>a</sup> 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, tienen ruido o donde si hay corrientes y turbulencias que afectan al vuelo.
- 4<sup>a</sup> Programado en Python 2.7 .

## 2.3 Metodología

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

## CAPÍTULO 2. OBJETIVOS

---

función de lo que se había conseguido hasta el momento. Una vez propuestas estas tareas se evaluaban los distintos riesgos que se podían tomar en función de trabajar de una forma u otra, y los avances a los que se podía llegar por cada camino. Una vez hecho esto, se comenzaban a desarrollar los algoritmos para conseguir estos objetivos en función de la forma elegida. Después se probaban (en simulación o con el robot real, dependiendo del objetivo), se proponía otra reunión para determinar si los avances eran los deseados o no, y en función de esto planear unos objetivos u otros.



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:

#### **2.4.1 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 solo en un trabajo sobre simulador.

#### **2.4.2 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 información. En este punto hubo que comprender la importancia que tendría la biblioteca OpenCV para este proyecto.

#### **2.4.3 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.

#### **2.4.4 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 como se tiene que comportar cuando no hay objetos de interés o cuando hay objetos que posiblemente lo sean.

# Chapter 3

## Infraestructura utilizada

Despues 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 dron tiene una *placa base* ARM Cortex A8 de 1Ghz y un DSP de vídeo de 8Ghz. También posee una memoria RAM DDR2 de 1GB y 200Mhz. Este chip es el encargado de levantar una red WiFi, a la cual nos podemos conectar mediante el smartphone (hay una 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 dron cuenta con dos cámaras, una horizontal y otra vertical, lo que nos permite ver diversos planos en todo momento. Estas cámaras están conectadas a la placa base, lo que permite en todo momento la transmisión de imágenes en directo, gracias a las cuales podremos trabajar, será la información gracias a la cual podremos darle instrucciones al dron.

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

### 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. Por esto, nos permite trabajar con el Parrot ArDrone, y de esta forma poder probar los distintos avances que se iban realizando. Éste nos permitía realizar pruebas sin tener que preocuparnos por el material ni los daños que e podían causar. 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 dron 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,

## CAPÍTULO 3. INFRAESTRUCTURA

---

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.

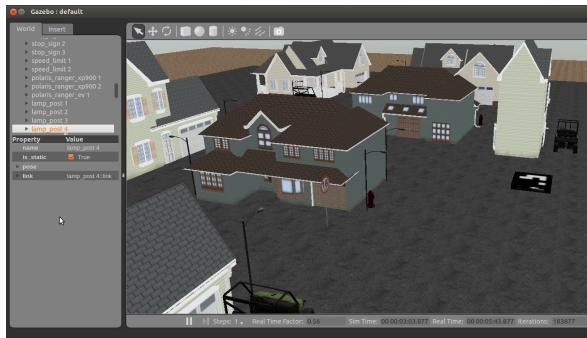


Figure 3.1: Mundo Gazebo.

## 3.3 Entorno JdeRobot

Este 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

Este es el servidor que nos permitía 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 cuadricóptero.

### 3.3.2 Plugin modelo de cuadricoptero Gazebo

Plugin que permite tener un ArDrone2 de Parrot en el simulador. Este tiene distintos sensores con los que trabajar (Sonar, IMU y Cámara), ademas de contar con 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 Camara, Pose3D, CMDVel, NAVData y ArDrone\_Extra.

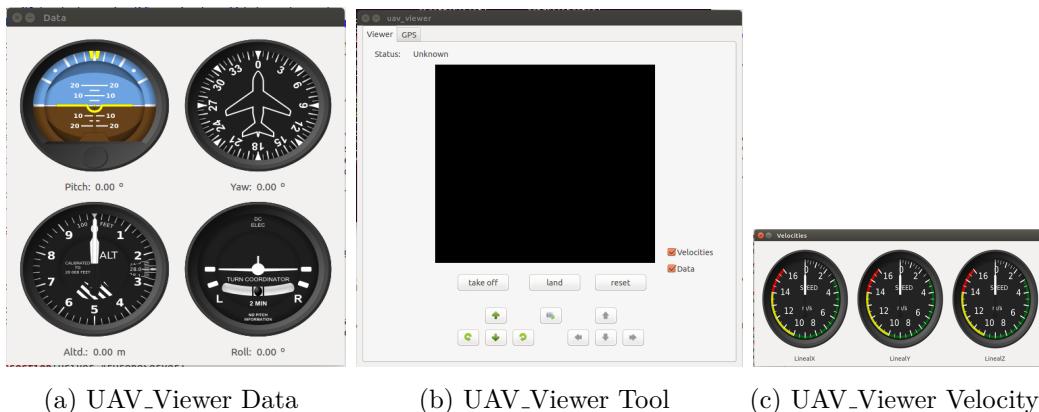


Figure 3.2: UAV viewer

### 3.3.4 Herramienta ColorTuner

Al inicio del proyecto era la herramienta ColorFilter. Esta herramienta nos permite, a partir de una imagen o vídeo, trabajar en los distintos espacios de color (como puede ser RGB, HSV, HSI...) y poder ir variando los parametros máximo y mínimo (entre 0 y 255) de cada uno para ver que colores cumplen las características y

## CAPÍTULO 3. INFRAESTRUCTURA

---

cuales no, viendose de esta forma, en otra imagen, sólo los colores que pasan este filtro dejando el resto como un fondo negro.

### 3.3.5 JdeRobot Academy

Con esta herramienta se ha trabajado en dos secciones.

- 1<sup>a</sup> Por una parte se ha utilizado para la práctica de autómatas. Esta nos permitia crear un diagrama de estados para saber en cada momento en que punto del algoritmo nos encontramos.

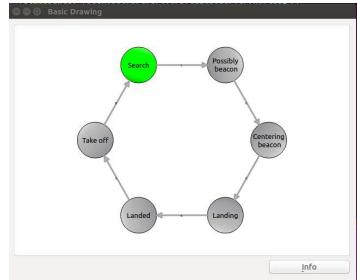


Figure 3.3: Diagrama de estados.

- 2<sup>a</sup> Por otro lado se ha utilizado follow\_turtlebot, al igual que UAVviewer, nos permite controlar el drone y obtener los datos de sus sensores, además de poder ejecutar nuestro algoritmo.

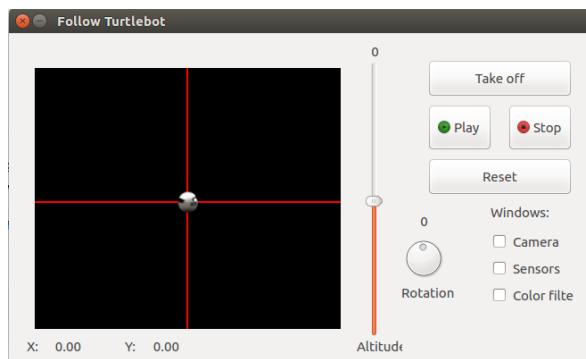
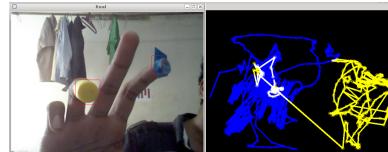


Figure 3.4: Follow Turtlebot.

### 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/>. Esta programada en C++ y es un estandar de facto en la visión artifical. Para el procesamiento de imágenes nos hemos apoyado principalmente sobre esta, trabajando en la versión 3.1. Gracias a ésta, al obtener la imagen que nos transmitía el drone podíamos tanto detectar objetos como aplicar cambios sobre ella, para marcar las zonas de interes o diferentes objetos. Esta librería sirve para el procesamiento de imágenes y visión computerizada. Nos ha permitido 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 evitaban imperfecciones en las imágenes como podía ser el ruido, que clasificaba 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 permitían encontrar los bordes de los objetos e indicar sobre una imagen final, con otro color, donde se encontraban dichos objetos.



(a) Logo OpenCV

(b) Procesamiento con OpenCV

Figure 3.5: UAV viewer

# Chapter 4

## Algoritmo

En este capítulo se describe el modo por el cual, con la infraestructura que se tiene, se llega a una solución para lo planteado en los objetivos.

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 el 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, y para finalizar una sección de control, para explicar los distintos movimientos del drone y de qué dependen estos.

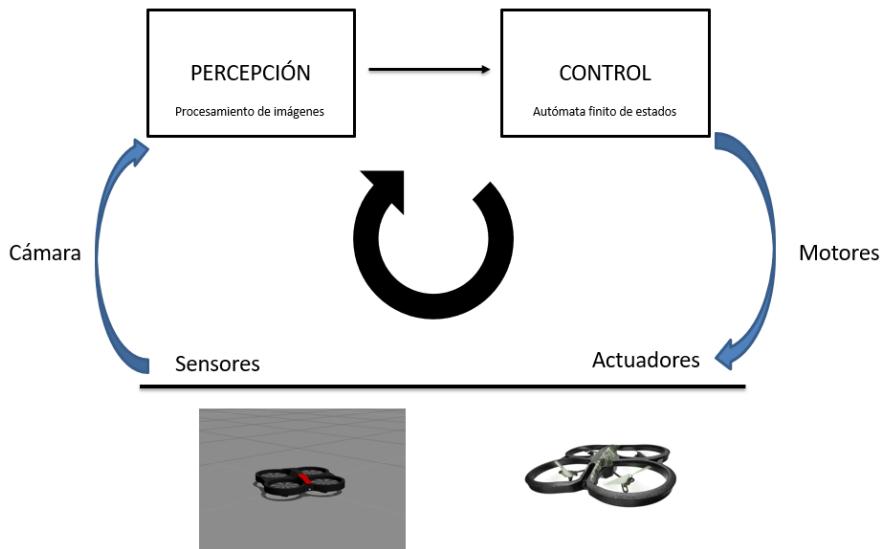
### 4.1 Diseño

El diseño de este algoritmo consta de un comportamiento reactivo de iteraciones continuas. Es un proceso basado en adquisición-procesado-envío de datos. La adquisición de los datos se realizan mediante los sensores del drone. Estos datos serán transmitidos al dispositivo para que lo procese y tras estos se enviarán las instrucciones al drone para que este 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.

## CAPÍTULO 4. ALGORITMO

---

Por otro lado, la parte 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. Destacar que este autómata está compuesto por 6 estados: despegue, búsqueda, 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 esta 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 esta, 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.



El lugar de aterrizaje del drone es una baliza previamente definida. Esta baliza es un cuadrado que en su interior tiene cuatro cuadrados, dos naranjas, y dos azules o verdes, dependiendo si se trabaja con el drone real o con el simulador. Esta baliza se ha elegido para que sea difícil confundirla con otro objeto, pues de ser una

## CAPÍTULO 4. ALGORITMO

---

baliza simple se podrían confundir los colores. De esta forma lo que buscaremos será la cruceta que forman estos cuatro cuadrados y el punto central de ésta. Por lo tanto, en lugar de un objeto en si lo que se busca es un objeto que tenga determinadas características y patrones, algo que se puede detectar a diferentes alturas, distintas condiciones y situaciones.

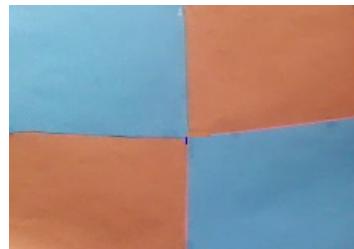


Figure 4.1: Baliza utilizada para el drone real.

Por último, para el control del drone con nuestro algoritmo se ha utilizado la herramienta follow\_turtlebot de JdeRobot Academy, explicada en 3.3.5 . Ésta 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 ésta 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 proceso 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 ésta imagen, se detecta si hay objetos de interés o no, y por tanto se envía unas instrucciones u otras al drone.

#### 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 (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 el filtro, a cada uno de los parámetros se le asigna un rango de valores entre 0 y 255. La herramienta colorTuner 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, dejando el fondo negro.

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 e 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

## CAPÍTULO 4. ALGORITMO

---

eliminan estas imperfecciones. Una breve explicación de los utilizados es la siguiente:

- **Erosión:** Dada una imagen y un elemento estructural, la erosión es el conjunto de los elementos  $x$  para los cuales el elemento estructural trasladado por  $x$  está contenido en la imagen.

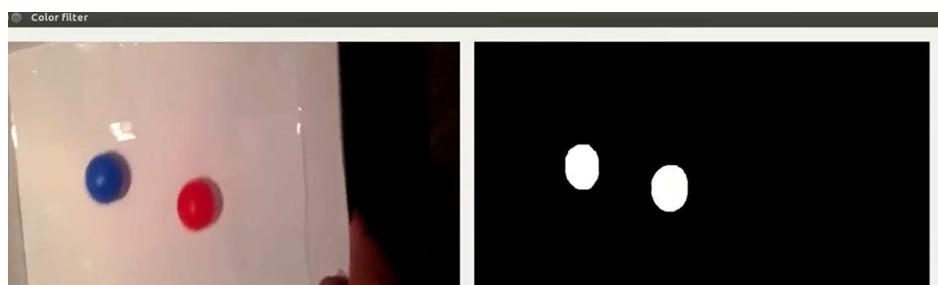
Aplicación: Cuando un píxel que parece pasar el filtro, pero los elementos de su alrededor(en concordancia con el elemento estructurante) no lo pasan, este pasa a ser parte del fondo.

- **Dilatación:** Transformación dual a la erosión. El resultado de esta es el conjunto de elementos tal que al menos algún elemento del conjunto estructurante esta contenido en  $x$ , cuando el elemento estructurante se desplaza sobre  $x$

Aplicación: píxeles que parecen de fondo, pasan a ser de la figura si están cerca de píxeles que pasan el filtro.

- **Cierre:** Se trata de realizar una dilatación en la imagen seguida de una erosión.
- **Apertura:** Se trata de realizar la erosión en una imagen seguido de la dilatación.

De esta forma, al obtener una imagen sin apenas ruido, evitando así que objetos de no interés los detecte como tal y que los objetos de interés los trate como píxeles de fondo. Una imágen en la que se observa la realización de un filtro de color y el uso de operadores morfológicos es la siguiente:



## CAPÍTULO 4. ALGORITMO

---

El siguiente fragmento de código, muestra como se define la matriz a partir de la cual se realizarán las operaciones oportunas, y despues 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 ésto 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.
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 este se descarta. En caso de tener ese area o mayor, se crea una imagen negra, se pintan los contornos del objeto sobre esta con un valor RGB(0,1,0) y se dilatan. De esta forma, tendremos tantas imágenes como objetos.
3. Se suman las imágenes obtenidas, por lo tanto, sobre una imagen negra final se pintaran los contornos dilatados de todos los objetos. Cada contorno se pintara 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 éste 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:

## CAPÍTULO 4. ALGORITMO

---

```
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:
    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 RGB que filtre 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.

## CAPÍTULO 4. ALGORITMO

---

5. Sobre la imagen obtenida, utilizando la función drawcontours, obtenemos la situación de estos y píxeles, teniendo así la situación de la cruceta, y con estos valores se puede marcar el centro de la baliza sobre la imagen real. En caso de haber filtrado la intersección de dos objetos en lugar de cuatro en el apartado anterior, se podrían marcar también posibles objetos de interés, como puede ser la baliza completa.

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 éste caso, para que se obtuviera una imagen mas 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 el conjunto de imágenes siguiente, 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.

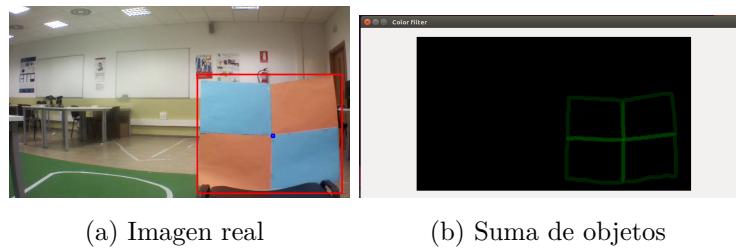


Figure 4.2: Procesamiento de imagen de la baliza

## 4.3 Control

En ésta sección se explica el control sobre el drone en función del momento en el que se encuentra y la imagen que se obtiene. Se puede dividir ésta sección en tres partes: despegue, búsqueda y aterrizaje.

### 4.3.1 Despegue

Esta fase está controlada por tiempo. Son los primeros diez segundos del algoritmo, y en ellos el drone despegue de forma controlada. Esta fase será ”Take off” en el autómata de estados. Para ello se sitúa el drone 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 drone 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 comenzará cuando finalicen los diez segundos de despegue. El drone comenzará a navegar de forma autónoma en búsqueda de una baliza sobre la cual aterrizar. Para ello comenzará un algoritmo en espiral, estado ”Search”, de forma que ira 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” e intentara centrarse sobre ella. En caso de no tratarse de la baliza continuara con el algoritmo de búsqueda, pero en caso de serlo pasará al estado ”Centering beacon”, 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 PID(progresivo, integral y derivativo). Esto se debe a que solo con el control progresivo, 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. Este control se basa en derivar el error con respecto al tiempo y multiplicarlo por una constante. Dado que nuestro algoritmo ejecuta una vez cada cierto tiempo y no esta continuamente pasando por este punto, podemos considerar que se trata de un sistema en tiempo discreto, y por tanto en lugar de trabajar con derivadas trabajaremos con sumatorios. De esta forma, para añadir un control derivativo se realiza una operación que depende de la velocidad anterior y la que tenemos ahora:

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

Para evitar que el valor sea 0 y al multiplicarlo por el valor progresivo se

## CAPÍTULO 4. ALGORITMO

---

quede en el sitio, en caso de ser un valor menor a 0.1 se iguala a este. Este resultado lo multiplicamos a la velocidad final, y lo que conseguimos es que si el valor entre dos velocidades continuas es muy alto este se atenúa de forma que el drone no cambie mucho y vaya oscilando, sino que lleve una velocidad mas continua a la hora de acelerar o frenar.

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
```

Para visualizar el diagrama de estados, se tuvieron que añadir dos paquetes del GUI de JdeRobot, uno que permitía abrir otra ventana y otro que permitía añadir estados y transiciones entre ellos. Para marcarlo, siendo el numero que aparece el numero del estado que queremos marcar, valía con la siguiente linea de código:

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

La imagen del diagrama que veríamos durante la ejecución se encuentra en 3.3.

### 4.3.3 Aterrizaje

Esta fase es un aterrizaje controlado, en el cual, una vez que el drone se ha centrado sobre la baliza comienza a descender, cambiando su estado a ”Landing”, y una vez el area de la baliza es suficiente, como para detectar que el drone está muy próximo a ella, se envía la instrucción land, aterrizando de ésta forma el drone, siendo

## CAPÍTULO 4. ALGORITMO

aquí su estado ”Landed”. El hecho de enviarse la orden de aterrizar cuando se está a una distancia cercana a la baliza es porque esta instrucción es a ciegas, una vez que se envía el drone solo se encarga de descender hasta que nota que está sobre un lugar sobre el cual posarse. Por lo tanto si se envía esta orden a una distancia lejana, el drone puede perder la referencia y aterrizar en otro sitio.

# 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.

Antes de comenzar, comentar diferencias importantes que hay entre trabajar en el entorno simulado y en el real. Primero, en el entorno simulado los colores puros ideales y mas fáciles de detectar, sin embargo en el entorno real debido a luces y sombras un color puede verse mas o menos nítido, además de ser mas difícil aislarlo del resto. También las velocidades a aplicar sobre uno y otro son distintas, pues en el entorno simulado se le puede dar una mayor velocidad que trabajara sin problemas, sin embargo en el real una velocidad alta y un movimiento brusco puede llevar a desestabilizarlo.

Por otra parte, en el drone real se daba el factor de la batería, en función de la carga que tuviera esta aterriza con mas o menos fuerza y realizaba movimientos mas o menos fluidos. A partir del 30% de batería no permitía al drone realizar todos los movimientos, y en algunas ocasiones perdía altura, aunque despues volvía a ganarla.

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

completo.

## 5.1 Percepción

Para las pruebas de esta sección, hay que tener en cuenta el objeto a detectar. Como ya se ha dicho anteriormente, este trata de un cuadrado con cuatro cuadrados 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).

Los primeros test se realizaban en el entorno simulado, al principio detectando ambos colores y viendo que se trataba de 4 objetos distintos. Después se comenzaron los test con el algoritmo final, viendo que los cuatro cuadrados formaban una cruceta y detectaba esta en la imagen, para así poder centrarse. Dependiendo de los distintos estados del algoritmo, cuando veía una parte de la baliza la detectaba y marcaba como posible objeto, en caso de tratarse de la baliza completa la marcaba como tal.



(a) Percepción posible objeto

(b) Detecta objeto

Figure 5.1: Drone detectando objeto.

Una vez se realizaron estos con éxito, pasaron a realizarse las pruebas con el drone real. Las primeras pruebas eran con el drone quieto y situando las balizas en distintas posiciones y viendo que las detectaba. Tras esto con el drone volando se hicieron pruebas, viendo que cuando se situaba la baliza debajo de este era capaz de detectarla.

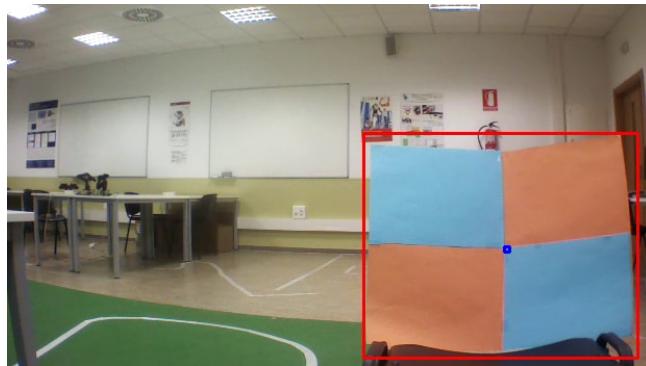
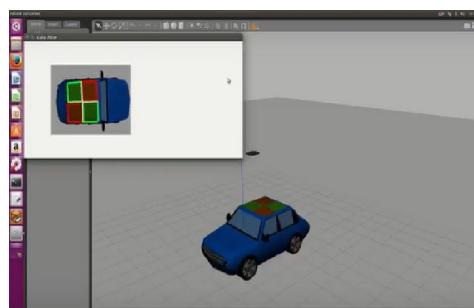


Figure 5.2: Detectando baliza real.

## 5.2 Despegue

Las primeras pruebas del despegue sobre el simulador fueron las mas sencillas. 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. Las pruebas consistían en despegar el drone y que este se centrara sobre la baliza, aunque también se probó a despegar el drone sobre el coche con la baliza y mover el coche para ver que el drone le iba siguiendo. Esta parte se decidió implementar porque el drone tenía una deriva que le llevaba a desplazarse hacia atrás cuando tenía que estar en el sitio. Cuando se hicieron las primeras pruebas de esto, se vió que el drone siempre despegaba en esta dirección y había un dos segundos en los que no se pueden controlar sus movimientos, por tanto había que contar con este desplazamiento.



## CAPÍTULO 5. EXPERIMENTOS

Para la realización de este experimento situábamos la baliza real sobre el suelo y el drone sobre esta, y así al despegar tenía un punto sobre el que centrarse.

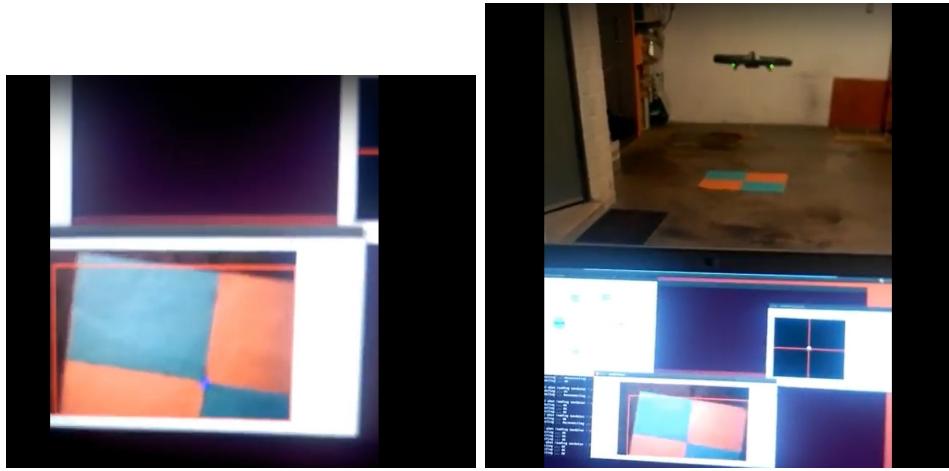


Figure 5.3: Despegue

Para la realización del algoritmo, el periodo de despegue se controla por tiempo, dejando 10 segundos para este. En el siguiente enlace esta el video con el despegue real del drone:

[https://www.youtube.com/watch?time\\_continue=26&v=HVGS1bA1tq4](https://www.youtube.com/watch?time_continue=26&v=HVGS1bA1tq4)

## 5.3 Búsqueda

Para la realización de ésta parte del algoritmo, como ya he comentado anteriormente, el drone va moviéndose en espiral hasta encontrar la baliza, para ya centrarse sobre esta. Al principio de los experimentos, el drone se situaba cerca de la baliza para empezar su desplazamiento y que al encontrarla se centrara. Una vez esto funcionaba, se probaba a poner el drone más lejos de ésta para que en la primera vuelta de la espiral no encontrara ningún objeto de interés, pero ya en la segunda vuelta pudiera

## CAPÍTULO 5. EXPERIMENTOS

---

verlo y se centrara, comprobando de esta forma que las espirales se realizaban de forma correcta. En los primeros experimentos el control, pues solo tenía componente proporcional y el momento que pasaba de no detectar nada a encontrar una posible baliza los movimientos eran muy bruscos, arreglando esto con el control PID.

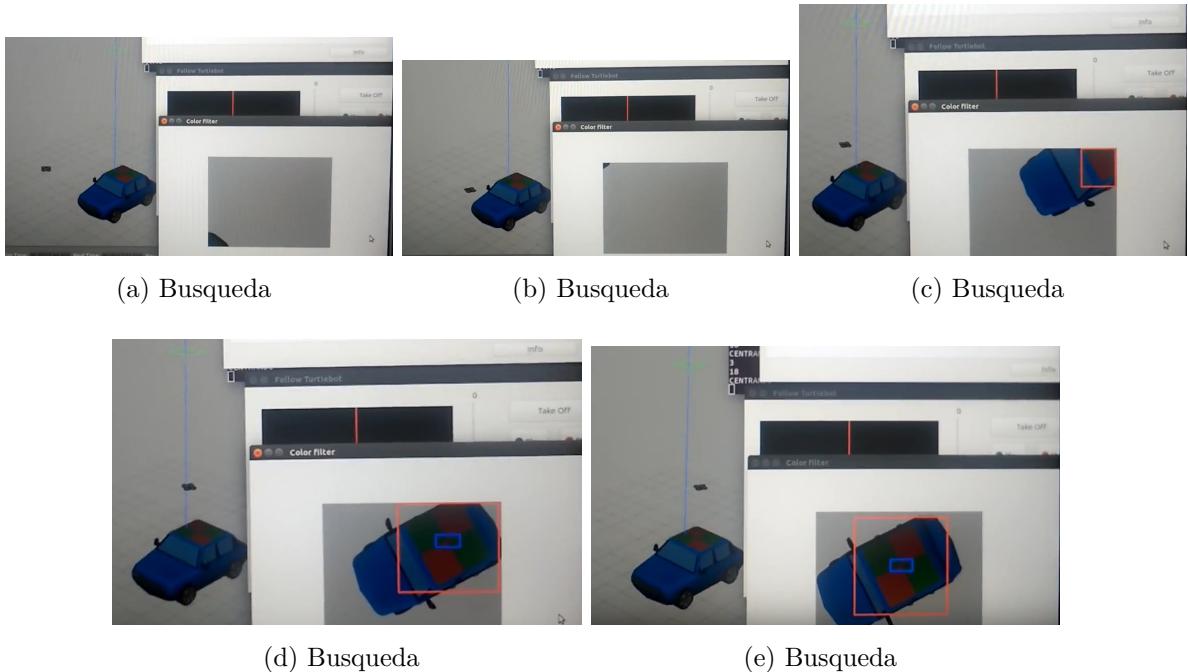


Figure 5.4: Busqueda

Para las pruebas con del drone real, primero se probó solo que el algoritmo de búsqueda era correcto, y que el drone realizaba espirales de forma correcta. Pero después, al realizar las pruebas finales en espacios más pequeños se modificó el algoritmo de búsqueda, haciendo que el drone se moviera haciendo cuadrados para tener un mayor control sobre éste.

## CAPÍTULO 5. EXPERIMENTOS

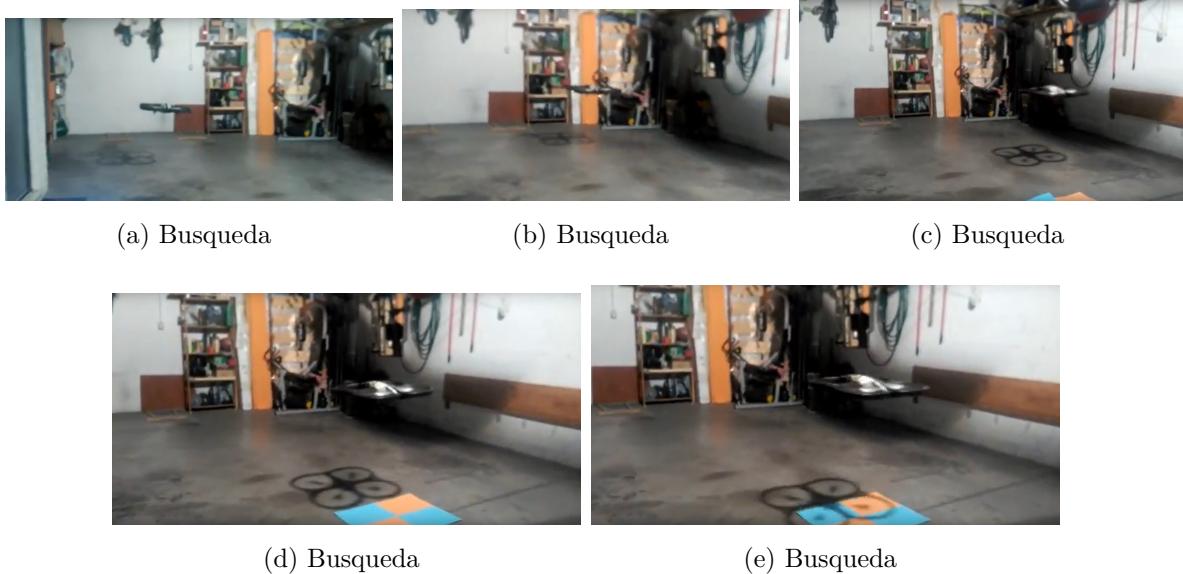


Figure 5.5: Busqueda

## 5.4 Aterrizaje

En los experimentos de esta parte, lo que se trató fue 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 aterrizar. Como podemos observar, en estos experimentos la parte de percepción se basan en controlar el centro de la baliza y el área, para tener una aproximación de la distancia que había a ésta.

## CAPÍTULO 5. EXPERIMENTOS

---

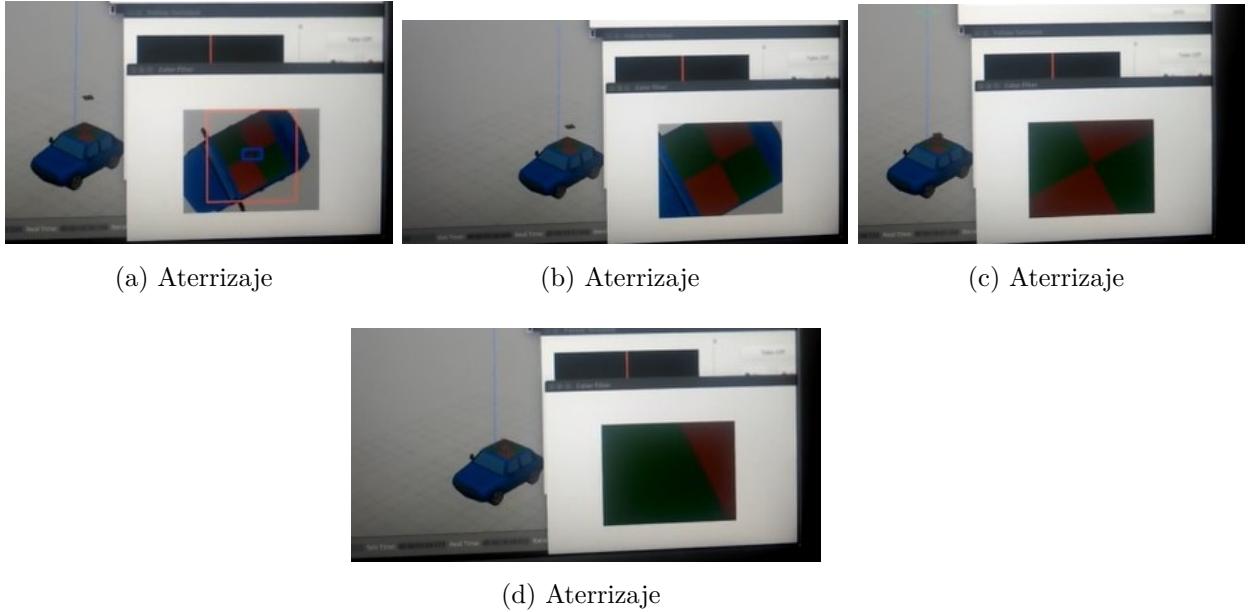


Figure 5.6: Aterrizaje

Para la realización de estas pruebas ha habido varias fases. En la primera, cuando el drone estaba en el aire se ponía delante de el la baliza de un color y al detectarla aterrizaba. Una vez funcione esto, se probó poniendo la baliza bajo el mientras no se le mandaban órdenes de velocidad, y ya para finalizar que mientras realizaba el algoritmo de búsqueda, una vez detectara la baliza aterrizará. Esta prueba los resultados eran buenos, ya que se conseguía que aterrizará, pero por la velocidad que tenía el drone y debido a la inercia, no aterrizaba en el sitio 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 lejos de la baliza. Pero tras esto y ya añadir los movimientos para centrarse sobre ésta, se solucionó el problema y se consiguió que el drone aterrizará verticalmente sobre la baliza.

## CAPÍTULO 5. EXPERIMENTOS

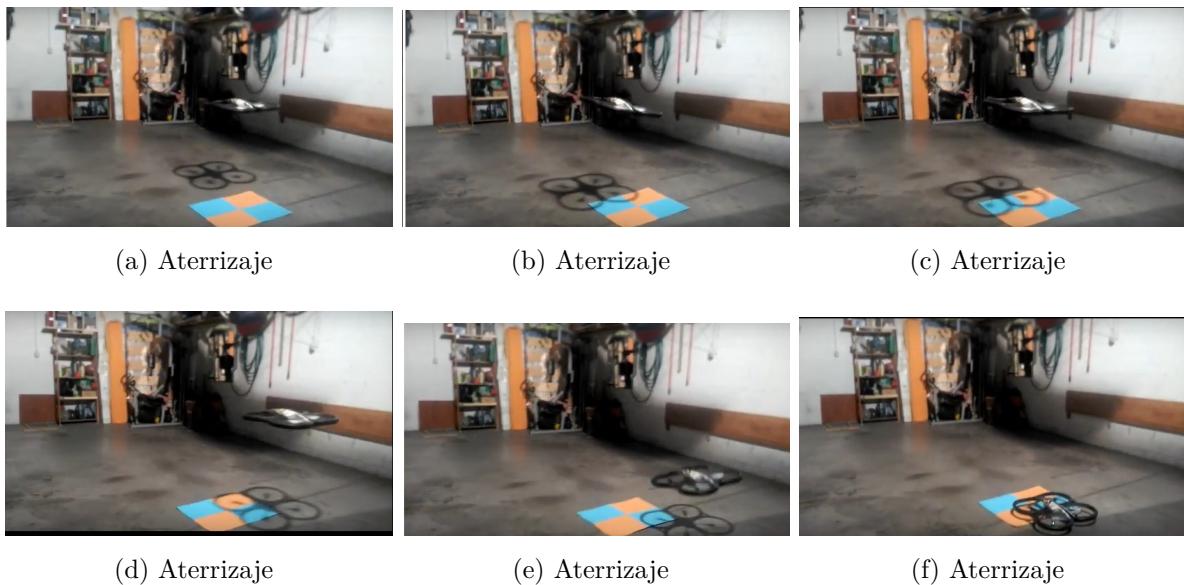


Figure 5.7: Aterrizaje

## 5.5 Algoritmo completo

Este experimento se realizó para comprobar que todo lo que se había programado por partes funcionaba también si se probaba el algoritmo completo. Cabe destacar que las pruebas salieron correctamente. Sobre el simulador se probó a poner el drone sobre la baliza, que despegaba y se situaba en el centro de esta. Una vez pasaron 10 segundos y el drone continuaba en el centro, se hizo que este 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 se centraba sobre estos, y una vez estaba prácticamente 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.

A continuación se muestra la secuencia de imágenes con el despegue y el drone alejándose de la baliza. Las imágenes de la búsqueda y el aterrizaje están en las figuras

## CAPÍTULO 5. EXPERIMENTOS

---

5.4 y 5.6. La referencia para con el vídeo es la siguiente:

[https://www.youtube.com/watch?time\\_continue=129&v=g9ZGJhRWTiY](https://www.youtube.com/watch?time_continue=129&v=g9ZGJhRWTiY)

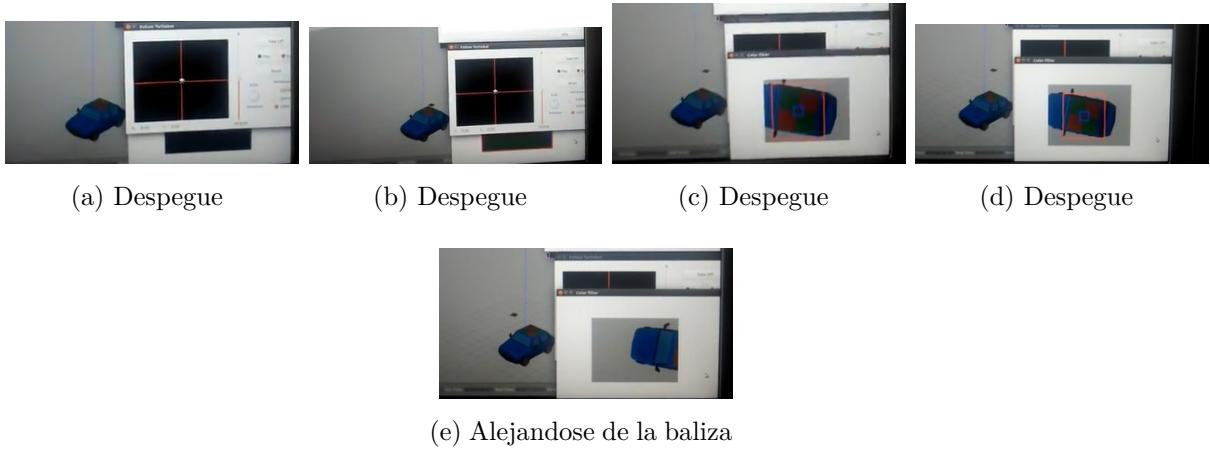


Figure 5.8: Algoritmo completo sobre el simulador.

Para la prueba con el drone real se colocaron dos puntos de referencia. Una baliza sobre la que situarse al despegar. 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 esta parte el drone se iba desplazando en función de las ordenes que se le mandaban en cada momento, hasta el momento en el que encontraba la baliza, momento en el que se centraba sobre esta y se le mandaba la orden de aterrizaje, quedando de esta forma el drone sobre la baliza.

# 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 mas 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. Estas diferencias se pueden observar en las sección 3.4 , donde se muestran imágenes de la percepción con el drone real

## CAPÍTULO 6. Conclusiones

---

y con el simulador.

Una vez obteníamos los objetos de los colores deseados, ver si estos eran 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, estas se descartaban como posibles objetos. Por otro lado, nos apoyamos en la forma de la baliza, pues los 4 cuadrados 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 esta, 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 este, haciendo distintas pruebas para la detección de

## CAPÍTULO 6. Conclusiones

---

balizas, así como el control del drone una vez se detectaban estas. Estas pruebas pueden verse a lo largo del capítulo 5 , pero muchos de los avances y cambios que se han dado durante el desarrollo estan disponibles en la wiki oficial del proyecto, indicada en el capitulo 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 las componentes integral y 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 este 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 navege 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.

## CAPÍTULO 6. Conclusiones

---

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