



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE SISTEMAS
AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

Seguimiento de un objeto con textura desde un drone con
cámara

Tutor: José María Cañas Plaza
Autor: Arturo Vélez Duque

Curso académico 2016-2017

Índice general

1. Introducción	1
1.1. Visión artificial	2
1.1.1. Aplicaciones	3
1.2. Drones (UAV)	6
1.2.1. Marco legal	8
1.3. Antecedentes	9
2. Objetivos	11
2.1. Descripción del problema	11
2.2. Requisitos	12
2.3. Metodología	12
2.4. Plan de trabajo	12
3. Infraestructura	14
3.1. Sistema operativo virtualizado	14
3.2. Gazebo	15
3.2.1. ARDrone	16
3.3. JdeRobot	17
3.4. Bibliotecas	18
3.4.1. Biblioteca de comunicaciones: ICE	18
3.4.2. OpenCV	19
3.4.3. NumPy	20
4. Desarrollo	22
4.1. Diseño	22
4.2. Percepción visual	23
4.3. Algoritmos de control	26

ÍNDICE GENERAL

5. Experimentos	28
5.1. Ejecución típica	28
5.2. Seguimiento visual de objetos dentro de una imagen	31
5.3. Detección manual del objeto	32
5.4. Seguimiento desde el drone de objetos de colores	33
6. Conclusiones y trabajos futuros	35
6.1. Conclusiones	35
6.2. Trabajos futuros	37

Índice de figuras

1.1.	Preprocesamiento para reducción de ruido.	2
1.2.	Coche autónomo y los datos que procesa.	4
1.3.	Acceso a parking mediante OCR.	5
1.4.	Termoimagen para detección de puntos calientes	5
1.5.	Proyecto MagicEye del Instituto Politécnico de Guarda	5
1.6.	Cuadricóptero, drone con cuatro motores.	6
1.7.	Dron MQ-1 Predator equipado para ataque.	7
1.8.	Amazon Prime Air: proceso de aterrizaje	7
1.9.	Intel Asctec Firefly: cámaras y detrás, mapa creado	7
1.10.	Campeonato de Drones de JdeRobot	10
3.1.	Logo de Gazebo	15
3.2.	Mundo simulado en Gazebo	15
3.3.	ArDrone de Parrot	16
3.4.	ArDrone simulado en Gazebo	16
3.5.	Logo del grupo de robótica de la Universidad Rey Juan Carlos	17
3.6.	Logo de JdeRobot	17
3.7.	Arquitectura de cliente y servidor de ICE	19
3.8.	Logo de OpenCV	19
3.9.	Logo de Numpy	20
4.1.	Esquema básico del diseño del componente	22
4.2.	Esquema ampliado del componente	23
5.1.	Objeto de pruebas	29
5.2.	Modo de inicio del comportamiento autónomo	29
5.3.	Drone en modo búsqueda	29
5.4.	Objeto localizado. Inicio de algoritmos de control.	30

ÍNDICE DE FIGURAS

5.5. Objeto centrado.	30
5.6. Drone persiguiendo al objeto. Inicio.	30
5.7. Drone persiguiendo al objeto.	30
5.8. Objeto de prueba: sombrero	30
5.9. Objeto de prueba: persona	30
5.10. Objeto de prueba: cuadrícula	31
5.11. Objeto de prueba: imagen de una flor	31
5.12. Objeto de prueba: copos de nieve	32
5.13. Objeto de prueba: copos de nieve en movimiento	32
5.14. Objeto de prueba: copo de nieve con región de interés	32
5.15. Objeto de prueba: copo de nieve seleccionando región de interés	33
5.16. Objeto de prueba: robot terrestre con colores	34

Introducción

La vista es el sentido que más utilizamos para captar información de nuestro entorno y consiste en la habilidad de detectar la luz y de interpretarla. Tanto los humanos como los animales poseemos un sistema visual que nos permite crear un esquema de nuestro entorno y conocer detalles de los objetos que nos rodean. Gracias a estos detalles somos capaces de reconocer dichos objetos por su color, por su forma, detectar movimiento en ellos o incluso estimar aproximadamente la distancia que nos separa.

Cada vez está más a la orden del día el comportamiento autónomo de las máquinas, ya que cada día estas se parecen más al humano, gracias a que les hemos dotado de sentidos, instalando en ellas sensores con los cuales pueden percibir la realidad , como por ejemplo usando cámaras para que vean. Mediante el desarrollo de algoritmos y gracias a la capacidad de procesamiento que tienen, se pueden tratar las informaciones que reciben de sus sensores para otorgarles cierta autonomía en función de la aplicación que le vayamos a dar a cada máquina en concreto.

Este trabajo de fin de grado es una aplicación directa de control visual. Consiste en un sistema automático que permite seguir a un objeto mediante la detección de puntos de interés que el mismo tiene para realizar un seguimiento en un entorno en este caso virtual, pero aplicable al mundo real.

Este primer capítulo recae en la intersección entre dos campos: la robótica aérea (en concreto los drones) y la visión artificial.

1.1. Visión artificial

La visión artificial o visión por computador es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un computador.

El objetivo principal que tiene esta disciplina, es que las máquinas sean capaces de ver e interpretar el mundo como un humano lo hace observando con sus ojos y procesando la información en el cerebro. Para ello, las máquinas encargadas de esto son dotadas de cámaras para percibir la realidad, tomando una o varias imágenes para posteriormente analizarla en su procesador. La comprensión de esta “realidad” captada por los sensores de los que dispone la maquina está basada en el procesamiento de la imagen o imágenes gracias al análisis mediante algoritmos de geometría, estadística, física y otras disciplinas.

La adquisición de estas imágenes no siempre es perfecta, existen ruidos e interferencias de diferente naturaleza, y estas pueden provocar un mal procesamiento de la mismas dando lugar a errores. Para ello, existen diferentes técnicas de preprocesamiento de imágenes para que estas sean lo más fidedignas a la realidad.

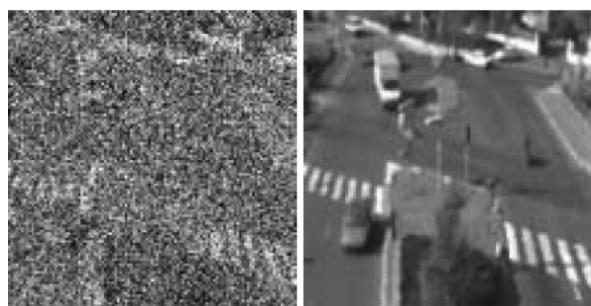


Figura 1.1: Preprocesamiento para reducción de ruido.

Una vez que se ha realizado el preprocesamiento de la imagen, se tiene algo muy parecido a la realidad, esta se puede utilizar con diversos fines. Dos de los más empleados son el aprendizaje automatizado (*machine learning*) y la detección de objetos.

El primero utiliza técnicas que tienen como objetivo conseguir diferenciar automáticamente patrones usando algoritmos matemáticos. Se pueden distinguir dos tipos de técnicas:

- Aprendizaje supervisado: Con este aprendizaje se entrena al ordenador con patrones previamente etiquetados, de forma que el algoritmo que emplee la máquina debe encontrar las fronteras que separan los posibles tipos de patrones.
- Aprendizaje no supervisado: Con este aprendizaje se entrena al ordenador con patrones que no han sido previamente clasificados y es el propio ordenador el que debe agrupar los distintos patrones en diferentes clases.

El segundo, la detección de objetos, es la parte de la visión artificial que estudia cómo detectar la presencia de objetos en una imagen sobre la base de su apariencia visual, bien sea atendiendo al tipo de objeto o a la instancia del objeto. La parte más importante es la extracción de características. Esta consiste en la obtención de modelos matemáticos compactos que hagan que la información de la imagen sea más simple con el fin de que el proceso de reconocimiento de objetos sea más sencillo. Las características que estos modelos obtienen se llaman descriptores, entre los que podemos encontrar histogramas, LBP, HOG, SIFT, SURF, ORB... Posteriormente para la búsqueda de objetos se pueden utilizar técnicas de aprendizaje automatizado para encontrar los clasificadores apropiados y que la detección se realice correctamente.

1.1.1. Aplicaciones

Algunas aplicaciones que están teniendo mucho auge en la actualidad son el coche autónomo, el reconocimiento de caracteres y en medicina entre otras. Entre estas aplicaciones sin duda la que más destaca entre las demás es el coche autónomo debido a que ya es una realidad y podemos encontrarnos estos vehículos en nuestras carreteras circulando entre coches no autónomos.

Los vehículos autónomos perciben el entorno mediante técnicas complejas como láser, radar, lidar, sistema de posicionamiento global y visión computarizada. Los sistemas avanzados de control interpretan la información para identificar la ruta apropiada, así como los obstáculos y la señalización relevante utilizando el hardware anteriormente mencionado. Mediante las cámaras se detectan los objetos para mantener el control del vehículo por la carretera, identificando las líneas o los márgenes de estas. También son capaces de identificar otros vehículos y peatones prediciendo su trayectoria mediante un software desarrollado para ello. El sistema de cámara lo complementa con un sistema de radares situados en el vehículo con el cual son capaces de controlar con más precisión la distancia a la cual se encuentra el objeto detectado. De esta manera, el vehículo es capaz de interpretar la realidad de una

manera muy parecida a la que podría hacer un humano.



Figura 1.2: Coche autónomo y los datos que procesa.

Otra aplicación es OCR, que son las siglas de *Optical Character Recognition*. Como su nombre indica es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente, a partir de una imagen dada, símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos. Así podremos interactuar con estos mediante un programa de edición de texto o similar. En los últimos años la digitalización de la información (textos, imágenes, sonido, etcétera) ha devenido un punto de interés para la sociedad. En el caso concreto de los textos, existen y se generan continuamente grandes cantidades de información escrita, tipográfica o manuscrita en todo tipo de soportes. En este contexto, poder automatizar la introducción de caracteres evitando la entrada por teclado implica un importante ahorro de recursos humanos y un aumento de la productividad, al mismo tiempo que se mantiene, o hasta se mejora, la calidad de muchos servicios. También se lleva utilizando durante muchos años en el reconocimiento de matrículas de vehículos con los radares, proporcionando una cadena de caracteres que se tienen que ajustar a un modelo conocido: el formato de una matrícula.

1.1. VISIÓN ARTIFICIAL



Figura 1.3: Acceso a parking mediante OCR.

El mundo de la medicina también ha aplicado la tecnología de la visión artificial. En la medicina deportiva se está empezando a utilizar una técnica de detección de lesiones en atletas mediante el uso de cámaras térmicas. La imagen tomada por estas cámaras es procesada para localizar los puntos calientes en el cuerpo del deportista, sinónimo de que en la zona detectada existe la posibilidad de que haya una lesión¹. También se está desarrollando un proyecto que tiene como objetivo principal permitir a las personas sin ningún tipo de movimiento en las extremidades superiores y que no pueden controlar los movimientos de la cabeza, controlar un ratón del ordenador sólo con el movimiento de los ojos mediante una cámara y un seguimiento de los mismos².

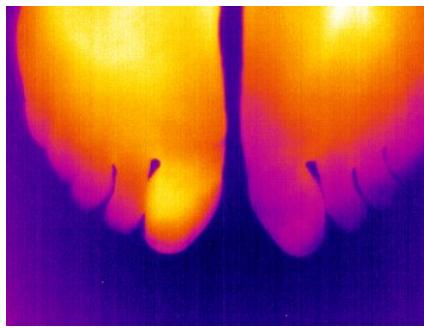


Figura 1.4: Termoimagen para detección de puntos calientes

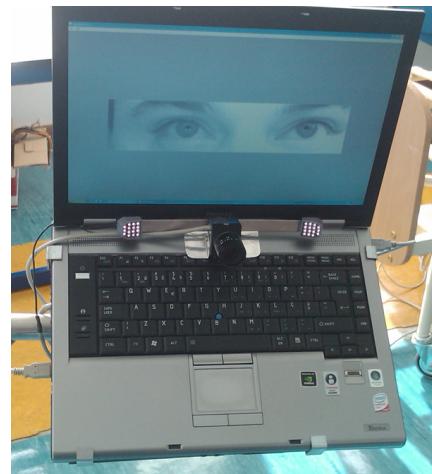


Figura 1.5: Proyecto MagicEye del Instituto Politécnico de Guarda

¹<http://blog.infaimon.com/2014/05/la-termografia-como-herramienta-de-diagnstico-en-medicina-deportiva/>

²<http://www.magickey.ipg.pt/>

1.2. Drones (UAV)

Un vehículo aéreo no tripulado (VANT), UAV (Unmanned Aerial Vehicle) o drone es una aeronave que vuela sin tripulación, concretamente se define como un vehículo sin tripulación reutilizable, capaz de mantener de manera autónoma un nivel de vuelo controlado y sostenido siendo propulsado por uno o varios motores.



Figura 1.6: Cuadricóptero, drone con cuatro motores.

Históricamente, los drones se emplearon para entrenar a los soldados en el uso de cañones antiaéreos, pero posteriormente, cuando se desarrollaron más era posible controlarlos remotamente gracias, en parte, a que además habían mejorado su autonomía. Unos de los drones más conocidos es el MQ-1 Predator de General Atomics, utilizado por el ejército americano, primero en misiones de reconocimiento y posteriormente en misiones de ataque.

Actualmente el drone se ha hecho muy comercial, gracias en parte al abaratamiento de costes. Hay empresas muy importantes que comercializan estas aeronaves tales como Syma, DJI o Phantom, las cuales son tres de las más punteras en el ámbito de los drones. Este tipo de aparatos pueden ser equipados con cámaras entre otros tipos de sensores como GPS, IMU, barómetro y brújulas.



Figura 1.7: Dron MQ-1 Predator equipado para ataque.

Gracias a todo esto, los drones hoy en día son empleados en aplicaciones agrícolas, vigilancia de fronteras y terrenos, búsquedas en zonas de difícil acceso, topografía, ocio, producciones audiovisuales e inspección de infraestructuras. Entre las líneas de desarrollo en las que se está trabajando con drones, se encuentra Amazon Prime Air³ e Intel Asctec Firefly⁴. En primer lugar Amazon, una de las empresas líderes de compras por internet, está desarrollando un drone con el cual podrá realizar sus entregas de una manera más rápida. Este drone va equipado con cámaras con las cuales detecta las etiquetas de la persona a la cual tiene que entregar el paquete. Desde el momento en que se despacha el paquete hasta que este es recogido por el comprador, es totalmente autónomo. Por otro lado, el proyecto de Intel consiste en la generación de un mapa 3D con unas cámaras que lleva el drone y así controlar las distancias con los objetos, evitando colisiones.



Figura 1.8: Amazon Prime Air: proceso de aterrizaje



Figura 1.9: Intel Asctec Firefly: cámaras y detrás, mapa creado

Un drone principalmente es un robot aéreo, compuesto de hardware y software. El hardware que compone a estos robots consta de actuadores, entre los que podemos encontrar sistemas de vuelo y hélices para otorgar la movilidad al robot, además de los ya mencionados sensores que éste puede llevar embarcados. Todo este hardware es controlado mediante un

³<https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011>

⁴<http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-firefly/>

software, que puede ser de mayor o menor complejidad en función del uso que se le vaya a dar al drone, pudiendo ser este controlado desde una tablet o smartphone o mediante una programación de rutas. Gracias al software también se puede asumir una locomoción resuelta y estable gracias a los mecanismos de control implementados.

1.2.1. Marco legal

En conceptos legales, la nueva ley temporal que regula el uso de drones en España fue aprobada el 4 de julio de 2014. Esta nueva ley va dirigida a los drones con un peso menor de 150 kg, quedando definidas las condiciones en las que se puede emplear un drone, entre las que se encuentran: grabación, vigilancia y monitorización, revisión de infraestructuras y obtención de mapas.

Inicialmente los drones se categorizan según su masa. Los menores de 2 kg, los que tienen entre 2 kg y 25 kg y los superiores a 25 kg. A medida que aumenta la masa, su uso está más controlado. Los drones con un peso menor de 25 kg tienen una restricción por la que se prohíbe su vuelo a altitudes superiores a 120 metros. Sea cual sea su masa, es necesario tener el carné de piloto de drones para poder manejar estos vehículos. Además, la aeronave deberá llevar una placa identificativa con el nombre del fabricante y los datos fiscales de la empresa.

El carné oficial para el manejo de drones no será necesario para aquellos que dispongan de un título de piloto de avión, ultraligero u otro específico. Sin embargo, los demás necesitarán pasar unos exámenes y pruebas oficiales para obtenerlo. Es importante que la escuela en la que se realice el curso sea ATO, es decir, escuelas certificadas por AESA. Además hay que tener en cuenta que hay dos cursos, uno normal y otro avanzado. El curso normal solo te habilita para volar el vehículo mientras lo tengas a la vista y el avanzado permite todo el alcance de la aeronave.

En cuanto a su uso en el espacio aéreo, si se quiere utilizar un dron es necesario pedir un permiso con una antelación de cinco días a la AESA (Agencia Estatal de Seguridad Aérea). Es muy importante recordar que está prohibido sobrevolar núcleos urbanos o espacios con una gran masificación de gente sin el consentimiento de AESA. Por seguridad será necesario tener un manual de operaciones cumplimentado, además de un estudio de seguridad de cada una de las operaciones que se llevarán a cabo. Cualquier infracción de las normas anteriores comportaría sanciones económicas de entre 3 000 y 60 000€.

1.3. Antecedentes

Es necesario situar en contexto este trabajo de fin de grado. Actualmente, hay presentados tres trabajos que guardan relación con el presente. Estos son los trabajos de Aitor Martínez, Iván Rodríguez y Alberto Martín Florido.

En los dos primeros la temática principal se basa en el control de la aeronave mediante diferentes aplicaciones. En el caso de Aitor Martínez, su trabajo consiste no solo en el manejo de aeronaves, sino de diferentes clases de robots mediante un interfaz web. En el caso de Iván Rodríguez, su trabajo consiste en el control de un drone concretamente, empleando para ello la tecnología webRTC. En el último mencionado, siendo este el que más se asemeja al desarrollo del presente trabajo, es el proyecto llevado a cabo por Alberto Martín Florido. Su objetivo principal se basa en el seguimiento de objetos con unas características marcadas, tales como la forma y el color. Gracias a la percepción visual del drone y a los mecanismos de control, el drone es capaz de ejecutar su tarea y ser capaz de seguir un objeto que tenga las características que cumplan con los criterios.

Hay que destacar los trabajos desarrollados por Jorge Cano, el cual ha desarrollado un drone el cual embarca distintos dispositivos y ha desarrollado los drivers mediante los cuales sería posible aplicar lo desarrollado en este trabajo a un drone real, y el trabajo de Daniel Yagüe, el cual ha desarrollado lo mismo que lo anteriormente mencionado pero en terreno virtual, pudiendo así desarrollar el presente trabajo de fin de grado.

El laboratorio de robótica ha organizado también el campeonato de programación de drones en el cual planteaban el reto de programación de un drone (gato) para que busque, persiga a otro robot aéreo (ratón) y se mantenga cerca de él, basándose en la información recibida de las cámaras que este lleva instaladas. El campeonato se desarrolla empleando un mundo virtual de Gazebo, una simulador para la creación de objetos 3D que se explicará más adelante.

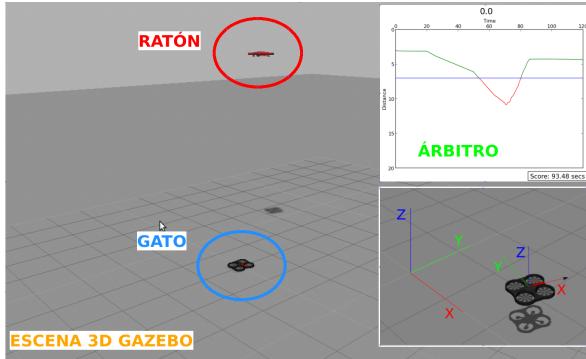


Figura 1.10: Campeonato de Drones de JdeRobot

Actualmente, en relación a este tema, muchos de los problemas iniciales para la ejecución de este trabajo están resueltos, ya que los drivers para la plataforma esta ya creado, además hay aplicaciones para el control de drones mediante aplicaciones web y desde teléfonos móviles y ahora se esta trabajando en nuevas aplicaciones aprovechando lo que ya está creado.

En el momento que se desarrollaba este trabajo de fin de grado, se estaban desarrollando paralelamente otros como por ejemplo el de Manuel Zafra, en el cual se desarrolla la navegación 3D en interiores, o el de Jorge Vela, el cual desarrolla un algoritmo mediante el cual el drone es capaz de buscar y aterrizar encima de un coche, o el trabajo de Diego Jiménez, el cual se centra menos en el procesamiento de la información tomada por los sensores y se centra en el soporte para el drone Solo Drone 3DR.

El presente aporta a este estado del arte una nueva forma de proceso de la información visual, siendo capaz de realizar un seguimiento de objetos más complejos que se asemejan más a la realidad, es decir, objetos con textura. En esta memoria se explica el desarrollo de esta nueva aplicación empezando por los objetivos iniciales marcados, siguiendo con la infraestructura empleada y concluyendo con las pruebas y conclusiones que se han obtenido durante el desarrollo del mismo.

Capítulo 2

Objetivos

Una vez expuesto el contexto en el que se realiza este trabajo fin de grado, en el presente capítulo se detallan los objetivos concretos y requisitos de la aplicación, así como la metodología y el plan de trabajo realizados.

2.1. Descripción del problema

El objetivo global de este trabajo de fin de grado es diseñar y programar una aplicación para que un drone pueda realizar un seguimiento de objetos con textura que se mueven por una superficie. Este seguirá un objeto detectado inicialmente acorde a lo programado. El algoritmo será capaz de manejar un robot cuadricóptero desarrollado en un entorno simulado en 3D.

Este objetivo general se ha articulado en tres subobjetivos:

1. Percepción visual: el algoritmo será capaz de detectar objetos de interés captados mediante una cámara y aplicando las diferentes técnicas de detección y emparejamiento de puntos para conseguir un mecanismo de control correcto.
2. Control del drone: se buscará un objeto y el drone será capaz de seguir dicho objeto de interés durante toda la prueba. Así mismo, deberá localizar el objeto de interés en caso de pérdida del objeto.
3. Experimentación: tras el desarrollo de ambos algoritmos se pondrán a prueba trabajando ambos a la vez y se validará experimentalmente la solución desarrollada en el drone simulado.

2.2. Requisitos

Los objetivos anteriormente descritos guiarán la realización del proyecto. La solución desarrollada además debería cumplir con los siguientes requisitos:

- El algoritmo tendrá que ser válido para su funcionamiento en JdeRobot 5.5 y Gazebo 7.4.
- Será necesario un procesamiento rápido y vivaz del flujo de imágenes.
- El algoritmo tendrá que ser robusto para minimizar los errores.
- Los resultados deberán ser válidos para diferentes tipos de objetos de interés.

2.3. Metodología

La metodología seguida se ha compuesto de varias herramientas empleadas para un avance seguro y constante en el desarrollo de los algoritmos.

El modo de trabajo ha seguido el desarrollo en espiral donde se iban alcanzando unos hitos para el correcto desarrollo de trabajo. Se buscaba el cierre de versiones por hito, de manera que hasta que una versión no estaba desarrollada y probada, no se pasaba al siguiente hito.

Cuando la versión era estable, se filmaba el resultado y se creaba una entrada en el cuaderno de bitácora, el cual es público y se puede encontrar la página dedicada al trabajo de la plataforma JdeRobot¹. Además, desde el principio del desarrollo se ha utilizado un repositorio en la conocida herramienta Github.

Se concertaban una reuniones periódicas con el tutor con el cual se discutía los problemas que se había tenido durante la ejecución del hito y se establecía un nuevo hito o se proseguía con los mismos hasta conseguir la resolución del problema y completar lo marcado.

2.4. Plan de trabajo

Los principales pasos seguidos durante el desarrollo del presente trabajo han sido:

1. Aprendizaje de JdeRobot y Python: En esta fase se abordan la comprensión de la plataforma de JdeRobot, la instalación del entorno, la interacción con los componentes

¹<http://jderobot.org/Avelez-tfg>

y la compresión de los mismos para su posterior modificación en función del objetivo deseado. También incluye el aprendizaje de la sintaxis de Python.

2. Aprendizaje de ICE y Gazebo: Durante este período se aprendió la manera de funcionamiento del interfaz de comunicaciones de ICE así mismo se empezaron a hacer las primeras pruebas con los modelos de Gazebo para comprender mejor ICE y las posibilidades que este tiene.
3. Aprendizaje de OpenCV: Una vez comprendida la infraestructura a usar se pasó a aprender la biblioteca OpenCV, de una manera más específica para el desarrollo del proyecto, aprendiendo a usar filtros de color, detección de puntos y flujo óptico, entre otras sencillas operaciones realizable por esta biblioteca.
4. Desarrollo de algoritmos de detección: Gracias al aprendizaje obtenido durante las fases anteriores, se ha procedido a implementar algoritmos de visión, como detección de puntos de interés.
5. Desarrollo de algoritmos de control: Tomando como punto de partida los datos obtenidos anteriormente, se ha desarrollado unos algoritmos capaces de enviar información precisa al drone para llevar a cabo el seguimiento del objeto, dotándole de autonomía para la toma de decisiones si se encuentra con diferentes situaciones durante las pruebas.
6. Validación experimental: empleando el entorno simulado, se ha procedido a hacer una serie de pruebas con el fin de detectar posibles comportamientos erróneos en el drone, corregirlos y validar el desarrollo.

Capítulo 3

Infraestructura

Para el desarrollo de este trabajo de fin de grado se ha empleado la plataforma del proyecto JdeRobot de software libre y software de terceros para de este modo poder adaptar la infraestructura disponible a las condiciones ideales y creación de contenido para la correcta ejecución de los algoritmos creados.

3.1. Sistema operativo virtualizado

Con la mejora de las redes actuales, y la velocidad que estas ofrecen cada vez más se están utilizando las tecnologías conocidas como “cloud” o “en la nube”. Este tipo de tecnologías consisten en una virtualización de las máquinas en centros de datos con servidores muy potentes y con múltiples replicaciones de las mismas para mayor seguridad de los datos en caso de catástrofe. Entre las empresas que ofrecen este servicio podemos encontrar Microsoft o Amazon, las cuales te permiten crear desde máquinas muy simples a maquinas tremadamente potentes capaces de procesar grandes cantidades de información.

En el caso de este proyecto, la virtualización se ha hecho localmente, en un disco duro externo y una copia de seguridad en un disco duro exactamente igual, con el objetivo de, en caso de destrucción parcial o total de la unidad, que haya una copia exacta en otro lugar para salvar la perdida de información. Además, se ha diseñado la infraestructura de manera que cualquiera de las replicas de la maquina virtual, pueda ser arrancada en cualquier máquina física, con el único requisito de que esta tenga instalado el programa apropiado para poder arrancar la maquina virtual.

Concretamente, la máquina empleada en este caso, ha sido una computadora Apple MacBook Pro con procesador Intel Core i5 de 2,4 GHz, 8GB de memoria RAM y tarjeta gráfica Intel Iris de 1,5 GB de VRAM y todo ello con un sistema operativo Mac OS 10.12.

Para la virtualización de la maquina se ha empleado la aplicación Parallels Desktop 11.0.2, dotando a la maquina virtual de 4GB de memoria RAM y 256MB de VRAM.

Durante la realización del proyecto hubo una descontinuación del soporte de Parallels Desktop 11.0.2 con Ubuntu 16.04, siendo necesario la migración de la máquina virtual creada en ese software a otro software llamado VirtualBox en su versión 5.1.14 y posteriormente una nueva migración a una máquina física por motivos de rendimiento. Esta máquina contaba con un procesador Intel Pentium Dual Core a 2.0 Ghz, 3GB de memoria RAM y 128MB de VRAM.

3.2. Gazebo

La simulación de robots es una herramienta esencial en la caja de herramientas de un desarrollador de robots. Un simulador bien diseñado permite probar rápidamente algoritmos, diseñar robots y realizar pruebas de regresión utilizando escenarios realistas.



Figura 3.1: Logo de Gazebo

Gazebo ofrece la capacidad de simular de forma precisa, realista y eficiente poblaciones de robots en entornos complejos de interior y exterior. Utiliza un robusto motor de física, gráficos de alta calidad y cómodas interfaces gráficas.

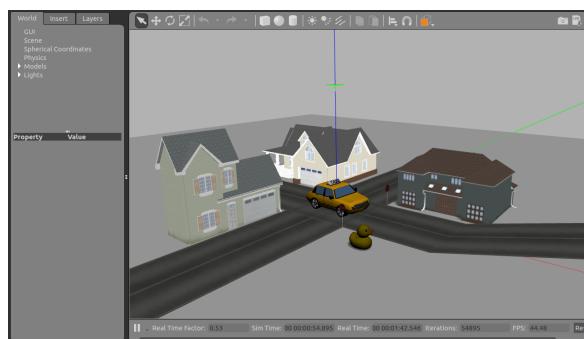


Figura 3.2: Mundo simulado en Gazebo

Gracias a que hay una cantidad importante de robots y objetos diseñados, se puede realizar todo tipo de pruebas de algoritmos de una manera muy próxima a la realidad pero sin la complejidad de necesitar un hardware para realizar las pruebas.

La última versión en la que se ha trabajado en la realización del trabajo es la 7.4, y en esta versión es donde se ha realizado la experimentación para validar así los algoritmos desarrollados.

3.2.1. ARDrone

El modelo empleado para el desarrollo del proyecto ha sido el AR.Drone de la marca Parrot, o más bien su modelo en el entorno de simulación Gazebo.

Este modelo es un drone recreativo de uso civil que fue diseñado originalmente para ser controlado desde un dispositivo iOS o Android, pero posteriormente Parrot liberó los applets de control bajo código abierto.



Figura 3.3: ArDrone de Parrot

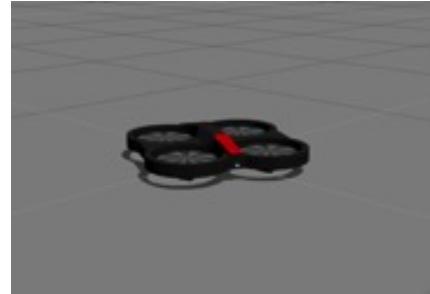


Figura 3.4: ArDrone simulado en Gazebo

Los algoritmos desarrollados son funcionales tanto en el drone modelado como en el drone real. Entre las especificaciones físicas podemos encontrar una velocidad de marcha (crucero) 5 m/s, 18 km/h y dos cámaras:

- Cámara frontal con sensor CMOS de tipo gran angular de lente diagonal de 93 grados de amplitud. Resolución 640x480 píxeles (VGA). Respuesta en Frecuencia: 15 cuadros/s.
- Cámara con sensor CMOS de alta velocidad de lente diagonal. 64° de amplitud. Resolución 176x144 píxeles. Respuesta en Frecuencia: 60 cuadros/s.

Estas características se han implementado en el simulador Gazebo con la generación de los drivers en JdeRobot, siendo estos los empleados para la ejecución del trabajo de fin de grado. Este modelo desarrollado para el simulador de entornos virtuales sigue de manera fidedigna el comportamiento, peso y manejo del modelo original pero de una manera simulada.

3.3. JdeRobot

JdeRobot es la plataforma del laboratorio de Robótica de la Universidad Rey Juan Carlos. Es un entorno para el desarrollo de aplicaciones domóticas, robóticas y de visión artificial que se rige por los términos de la licencia GNU GPL versión 3. La arquitectura de JdeRobot está orientada a componentes que funcionan de manera independiente para llevar a cabo tareas complejas. Hay componentes que sensan la realidad y envían los datos a otros componentes que analizan los datos recibidos. La comunicación entre los citados componentes se realiza mediante interfaces ICE, que nos abstraen de la implementación de las comunicaciones.



Figura 3.5: Logo del grupo de robótica de la Universidad Rey Juan Carlos



Figura 3.6: Logo de JdeRobot

La arquitectura de JdeRobot está dividida en tres partes bien diferenciadas:

1. Los componentes, son las mini-aplicaciones que proveen de datos o de herramientas de análisis.
2. Las interfaces, que permiten la intercomunicación entre componentes
3. Las bibliotecas, que facilitan el desarrollo con funcionalidades complejas.

Internamente los componentes se puede dividir en dos grupos: los drivers, proporcionan los flujos de datos en bruto y las herramientas que analizan estos datos e implementan la

funcionalidad final. Un ejemplo es cameraserver que recibe las imágenes de la cámara y las sirve en la red a la herramienta cameraview que las muestra e indica los fotogramas por segundo del vídeo recibido. Yendo un paso más existe el componente followturtlebot, el que ha servido como base para la realización del trabajo. Este componente contiene parte de visión y parte de control, sirviendo perfectamente de punto de partida para el desarrollo de los objetivos. Este componente incluye un interfaz gráfico donde se puede dar órdenes al drone, permite visualizar la cámara y los datos de los diferentes sensores y además incluye la opción de ver el procesamiento que se ha realizado sobre la imagen. El uso de este componente ha sido posible gracias a la utilización de cameraserver como servidor de imágenes. Es una herramienta muy útil que provee imágenes de una cámara, de un fichero o de un flujo de datos en línea, lo que facilita las cosas ya que con unos sencillos cambios en un fichero de configuración obtendremos las imágenes deseadas. Otro componente que cabe destacar es OpenCVDemo, gracias al cual ha sido posible la compresión de los métodos empleados para el desarrollo de los objetivos, ya que incluye funciones básicas de la biblioteca OpenCV, la cual se explicará más adelante.

Durante el desarrollo del trabajo de fin de grado se han llevado a cabo diferentes actualizaciones de la infraestructura de JdeRobot. La última versión empleada ha sido la 5.5.

3.4. Bibliotecas

El lenguaje empleado en este trabajo de fin de grado es Python. Para el desarrollo de los algoritmos se han empleado fundamentalmente dos bibliotecas o paquetes que permiten a este lenguaje que realice las operaciones necesarias para el correcto funcionamiento del proyecto, además de emplear una biblioteca de comunicaciones para conectar los componentes. Estas son OpenCV, Numpy y ICE.

3.4.1. Biblioteca de comunicaciones: ICE

ICE o Internet Communication Engine es un middleware orientado a objetos con soporte para diversos lenguajes de programación: C++, .NET, Java, Python, ObjectiveC, Ruby y PHP. ICE provee de objetos orientados a RPC, computación distribuida y servicios de publicación/suscripción. Es desarrollado por Zeroc y tiene una licencia dual: bajo una licencia privativa y bajo GNU GPL. Se encuentra disponible en la mayoría de los sistemas operativos actuales y tiene también una versión reducida para entornos móviles.

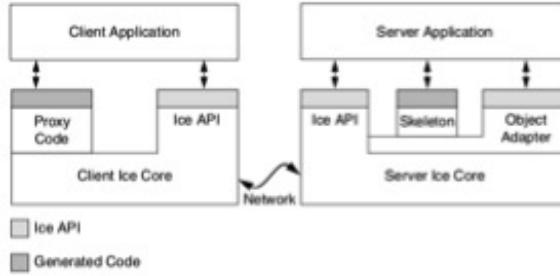


Figura 3.7: Arquitectura de cliente y servidor de ICE

ICE proporciona la abstracción de la implementación de la capa de comunicaciones a través de la red. Gracias a esto es posible la comunicación remota con cámaras que proveen imágenes o con otros componentes de JdeRobot. Estas interfaces se encuentran definidas en un lenguaje SLICE o Specification language for ICE que puede ser compilado en varios lenguajes de programación diferentes, permitiendo el desarrollo multilenguaje de aplicaciones.

3.4.2. OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.



Figura 3.8: Logo de OpenCV

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y

Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión.

Esta escrita en C/C ++, la biblioteca puede aprovechar el procesamiento multi-core. Habilitando OpenCL, puede aprovechar la aceleración de hardware de la plataforma empleada. OpenCV tiene más de 47 mil personas en la comunidad de usuarios y el número estimado de descargas más de 9 millones. Los usos van desde el arte interactivo hasta la inspección de minas, mapas en la web o la robótica avanzada.

Gracias a esta biblioteca se han podido conseguir los objetivos de percepción visual, ya que las funciones de las cuales dispone esta biblioteca permiten hacer operaciones sobre las imágenes obtenidas por nuestro drone y obtener así una serie de datos muy útiles para el desarrollo de los mecanismos de control. La versión utilizada para el desarrollo del trabajo de fin de grado es la 3.2.0.

3.4.3. NumPy

NumPy es el paquete fundamental para la computación científica con Python. Contiene entre otras cosas:

- Manipulación de arrays de N dimensiones.
- Herramientas para integrar código C / C ++ y Fortran.
- Capacidad de manipulación de álgebra lineal, transformada de Fourier y números aleatorios.

Además de sus usos científicos, NumPy también puede ser utilizado como un eficiente contenedor multidimensional de datos genéricos y se pueden definir tipos de datos arbitrarios, esto permite a NumPy integrarse de forma transparente y rápida con una amplia variedad de bases de datos.



Figura 3.9: Logo de Numpy

NumPy se licencia bajo la licencia de BSD, permitiendo la reutilización con pocas restricciones.

Esta biblioteca, en su versión 1.12.1, ha sido utilizada para complementar a OpenCV y poder así simplificar algunos de los algoritmos empleados en el trabajo de fin de grado.

Capítulo **4**

Desarrollo

En este capítulo se describe la solución diseñada y programada para conseguir los objetivos anteriormente definidos, es decir, el desarrollo de cómo gracias a la cámara embarcada en el drone, en este caso, simulado, es capaz de hacer un procesamiento de la imagen y un control sobre los plugins de Gazebo para realizar la función que se ha planteado.

El desarrollo ha consistido en 3 partes marcadas:

1. Diseño
2. Percepción visual
3. Algoritmos de control

4.1. Diseño

El diseño del componente desarrollado es muy sencillo. Por un lado se recibe las imágenes desde la cámara del drone y se obtiene una salida en modo de movimiento del drone.



Figura 4.1: Esquema básico del diseño del componente

Por lo tanto basándonos en este esquema, el drone va a tener que ser capaz de tomar imágenes, tratarlas aplicando ciertos algoritmos y con esa información utilizar algoritmos y

funciones para poder tener la respuesta apropiada. Aquí es donde se subdivide el problema en dos, parte perceptiva y parte de control:

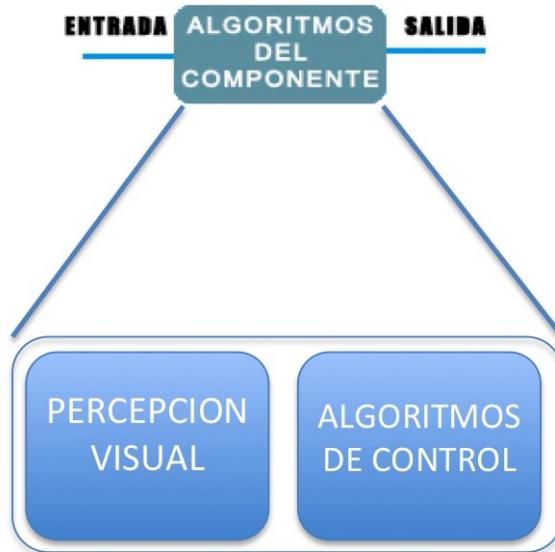


Figura 4.2: Esquema ampliado del componente

4.2. Percepción visual

Esta parte es la parte principal con la que tiene que lidiar el componente. Para obtener la entrada, el componente se conecta al interfaz de comunicaciones ICE mediante la función `self.camera.getImage()`, esta función se encarga de obtener las imágenes que el interfaz ofrece y este a su vez las puede obtener de diferentes fuentes, en este caso, la obtiene de la cámara del drone simulado de Gazebo.

Una vez se obtiene la imagen correctamente, el componente busca en la imagen el objeto de interés. Para ello, tiene un área de búsqueda en la misma y se mantiene en ese estado hasta que el objeto aparece en la imagen.

```
1 def setROI():
2     global refPt, size
3
4     refImg = self.camera.getImage()
5     size = refImg.shape
6     schSize = ((size[1]-10),(size[0]-10))
7     refPt = [(10, 10), schSize]
```

Cuando el objeto está en la imagen se inician los algoritmos de búsqueda y seguimiento de puntos. El primero con el que se encuentra es `cv2.goodFeaturesToTrack()`. Esta

función de OpenCV encuentra las N esquinas más fuertes en la imagen por el método Shi-Tomasi. Este método parte de la ecuación $R = \min(\lambda_1, \lambda_2)$, donde si R esta por encima de un umbral, se considera como una esquina. Como de costumbre, la imagen debe ser una imagen en escala de grises. En la función se especifica el número de esquinas a buscar, el nivel de calidad, que es un valor entre 0-1 e indica la calidad mínima de la esquina debajo de la cual todos son rechazados y por último proveemos la distancia euclíadiana mínima entre las esquinas detectadas. También, a modo de hacer más eficiente el código, elimina los puntos que están fuera del objeto.

```

1  src = image.copy()
2
3  src = cv2.medianBlur(src, 3)
4  src_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
5
6  p0 = cv2.goodFeaturesToTrack(src_gray, 100, 0.01, 10, None, None, 7)
7
8  index = 0
9  if p0 != None:
10     for i in (p0):
11         if (i[0][0] < refPt[0][0]) or (i[0][0] > refPt[1][0]) or (i[0][1] < refPt
12             [0][1]) or (i[0][1] > refPt[1][1]):
13             p0 = np.delete(p0, index, axis=0)
14         else:
15             index = index + 1

```

Entonces, una vez calculados los puntos en el fotograma N, pasa a calcular la posición de los mismos en el fotograma N+1. Partiendo de la suposición de que todos los píxeles vecinos tendrán movimiento similar, se utiliza el método de Lucas-Kanade que toma un parche 3x3 alrededor del punto, así que todos los 9 puntos tienen el mismo movimiento. Podemos encontrar (f_x, f_y, f_t) para estos 9 puntos. Así que ahora nuestro problema se convierte en la solución de 9 ecuaciones.

OpenCV soluciona todo esto en una sola función, `cv2.calcOpticalFlowPyrLK()` que realiza el seguimiento de algunos puntos en un vídeo. La función recibe el fotograma anterior, los puntos anteriores y el fotograma siguiente. Devuelve los puntos siguientes junto con algunos valores de estado que tiene un valor de 1 si se encuentra el siguiente punto, de lo contrario cero. Del mismo modo, la imagen tiene que ser en escala de grises.

```

1  src2 = self.camera.getImage()
2
3  src2 = cv2.medianBlur(src2, 3)
4  src2_gray = cv2.cvtColor(src2, cv2.COLOR_BGR2GRAY)
5
6  p1, st, err = cv2.calcOpticalFlowPyrLK(src_gray, src2_gray, p0, None, None, None, 30, 30),
7  2, (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

```

Pero esto no es suficiente para asegurar que los puntos obtenidos son los válidos, porque si estos llevan el valor 0 asignado, no tienen punto emparejado. Por ello, es necesario hacer

una comprobación de ese estado para ver cuales puntos están emparejados y cuales no. Esto lo podemos conseguir generando un array donde se van a ir almacenando todos aquellos puntos en los que su estado es 1, es decir, aquellos que están emparejados.

Con los puntos válidos, es el momento de definir cuales son los más externos del objeto, lo que ayuda a definir dinámicamente la región de interés dentro de la cual va a estar el objeto.

```

1 good_p1 = p1[st==1]
2
3 maxAll = np.amax(good_p1, axis = 0)
4 minAll = np.amin(good_p1, axis = 0)
5
6 maxX = maxAll[0]
7 maxY = maxAll[1]
8 minX = minAll[0]
9 minY = minAll[1]
```

Teniendo la región de interés donde esta contenido el objeto, la cual es rectangular, se obtiene el centro de la misma calculando el punto medio de la diagonal que une dos vertices opuestos.

```

1 def squareCenter(xmax, xmin, ymax, ymin):
2     global center
3     center[0] = ((xmax + xmin)/2)
4     center[1] = ((ymax + ymin)/2)
```

Con todo esto el componente es capaz de obtener la posición del objeto dentro del plano de la imagen y poder hacer un seguimiento del mismo durante todo el flujo de fotogramas de manera iterativa.

En ocasiones el objeto desaparecerá de la imagen, bien porque sale del plano o bien porque se hace más pequeño. En tal caso, los puntos soporte no podrán ser calculados y por tanto no habrá puntos “buenos” para poder realizar el seguimiento dentro de la imagen, por lo que la percepción del componente se reiniciará al modo de búsqueda hasta que el objeto vuelva a aparecer en la imagen o sea posible calcular los puntos.

En las versiones iniciales del trabajo de fin de grado se realizaron utilizando imágenes con unas características muy marcadas, como por ejemplo colores fuertes para poder emplear filtros de color. Aquí la percepción tiene unas ligeras modificaciones.

En este caso la percepción se basa en colores y, para saber si un objeto es de interés o no, emplea un filtro de color del cual resulta una imagen en blanco y negro donde, si el objeto tiene ese color, aparece en blanco.

```

1  hsv = cv2.cvtColor(input_image, cv2.COLOR_BGR2HSV)
2
3  lower = np.array([50,140,50])
4  upper = np.array([120,255,255])
5
6  mask = cv2.inRange(hsv, lower, upper)

```

Con esta imagen binaria se utilizan una serie de funciones que el propio OpenCV ofrece para obtener la región de interés y el centro del mismo utilizando los momentos:

```

1  contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
2  cnt = contours[0]
3  M = cv2.moments(cnt)
4  centroid_x = int(M['m10']/M['m00'])
5  centroid_y = int(M['m01']/M['m00'])

```

4.3. Algoritmos de control

La otra parte del componente es la encargada de controlar el drone. Para que esto sea posible, el componente hace uso de la función `self.cmdvel.sendCMDVel()`, la cual conecta con el interfaz de comunicaciones ICE, que es el encargado de mandar la información que la función toma de las velocidades en los ejes X, Y y Z, grado de guiñada e inclinación y la velocidad de giro entorno al eje vertical, al drone.

Los algoritmos de control se pueden dividir en dos:

1. Algoritmos de búsqueda
2. Algoritmos de seguimiento

Dentro de los algoritmos de búsqueda, se desarrollaron dos, el primero se basaba en elevarse hasta encontrar el objeto, pero este no era muy útil en algunas condiciones puesto que su referencia de altura se obtenía a partir del área de objetos simples. El otro método empleaba un método de búsqueda en espiral, abriendo y cerrando la misma. Los cambios de dirección del drone van dirigidos por tiempo, es decir, durante 1 segundo se mueve en X, durante 2 segundos se mueve en Y, durante 3 segundos en -X, durante 4 segundos en -Y, durante 5 segundos de nuevo en X y así sucesivamente hasta un tiempo definido experimentalmente en función del tamaño del escenario.

Para el tiempo se tiene en cuenta el intervalo de tiempo empleado para una orden y el anterior para poder hacer esa función de abrir y cerrar la espiral:

```

1 def changeTime():
2     global sec, secless
3     if sec > secless and sec < 20:
4         secless = sec
5         sec = sec + 1
6     elif sec < secless + 2 and sec > 2:
7         secless = sec
8         sec = sec
9     else:
10        sec = 1
11        secless = 0

```

El cambio de dirección se hace tomando la velocidad con la que estaba moviéndose el drone en el intervalo anterior:

```

1 def changeDirection():
2     global velX, velY
3
4     if velX != 0.0 and velY == 0.0:
5         velY = -velX
6         velX = 0.0
7     elif velX == 0.0 and velY != 0.0:
8         velX = velY
9         velY = 0.0

```

Todo ello gobernado por la función que gestiona el paso del tiempo y lanza la orden de cambio de dirección y aumento del tiempo:

```

1 def seek():
2     global velX, velY, sec, secless
3     self.cmdvel.sendCMDVel(velY, velX, 0,0,0,0)
4     time.sleep(sec)
5     changeTime()
6     changeDirection()

```

Una vez que el objeto esta localizado, se inician los algoritmos de seguimiento. Se obtiene el punto central del objeto desde la parte perceptiva y se aplica la formula, para calcular la velocidad, $V = (R_c - C)/R_c$, donde V es la velocidad, R_c es el punto de referencia, C es el punto central del objeto, el cual el drone tiene que hacer coincidir con R_c .

```

1 velX = ((refCenter[0] - center[0])/refCenter[0])
2 velY = ((refCenter[1] - center[1])/refCenter[1])
3
4 self.cmdvel.sendCMDVel(velY, velX, 0,0,0,0)

```

Este método pretende siempre tener el objeto en el punto de referencia, es decir, si el objeto se esta desplazando, el algoritmo va a tratar volverlo a poner en el centro, para así tratar de hacer coincidir el centro del objeto con el punto de referencia. Si el drone pierde el objeto, ejecuta de nuevo los algoritmos de búsqueda automática por espiral.

Capítulo **5**

Experimentos

Se han hecho diferentes experimentos con el componente para validar su correcto funcionamiento y que cumple con los objetivos marcados de hacer el seguimiento de un objeto con textura.

Estos experimentos con las diferentes características de las que se compone la versión final son la prueba de que el desarrollo es válido, y que individualmente cada una de sus características funcionan tal y como se espera, es decir, el componente tiene esa información de entrada y se le aplican los métodos necesarios para que a la salida se tenga el resultado esperado.

5.1. Ejecución típica

Así pues, en la versión final del componente el comportamiento habitual consiste en que, tras ejecutar el mundo simulado de pruebas y el componente, se le ordena al drone despegar a través del interfaz gráfico. La aplicación ofrece ese interfaz gráfico para teleoperar el drone o para activar su comportamiento autónomo

El objeto de interés para los experimentos, es un robot terrestre que lleva en su parte superior una etiqueta con una imagen con textura. Este robot se teleopera desde otro componente para poderlo mover por el escenario y comprobar que el drone realiza el seguimiento del mismo. En la figura 5.1 se puede ver el robot terrestre con la imagen de una persona vista desde arriba para realizar las pruebas.

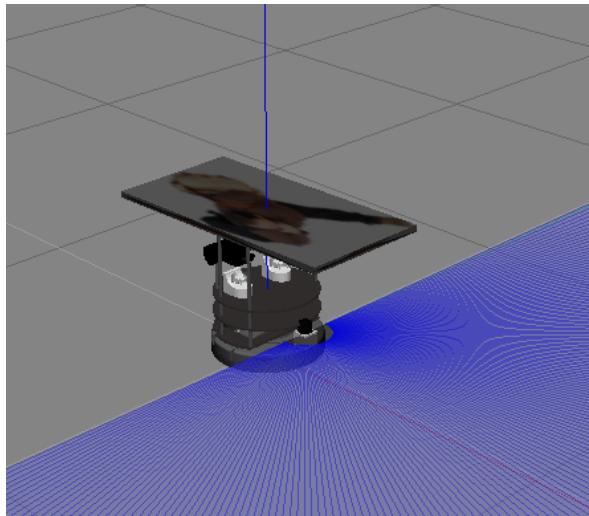


Figura 5.1: Objeto de pruebas

Con el drone en el aire, el método de ordenarle al mismo que empiece la búsqueda y seguimiento es a través del botón play del teleoperador. En este momento el drone empieza a ejecutar simultáneamente la percepción visual y los algoritmos de control, inicialmente la búsqueda del objeto.

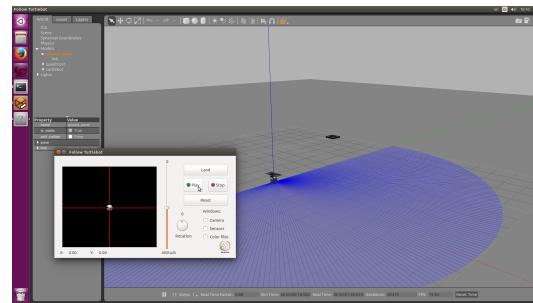


Figura 5.2: Modo de inicio del comportamiento autónomo

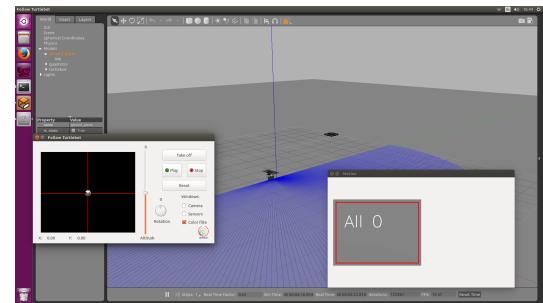


Figura 5.3: Drone en modo búsqueda

Mientras que el drone está en modo búsqueda se está desplazando por el escenario trazando una espiral de apertura y de cierre como se ha explicado en el capítulo anterior. Una vez que el objeto aparece en la imagen y el componente lo detecta, comienzan los mecanismos de seguimiento, tanto visual como persecución. En la parte visual se obtienen los puntos soporte, la región de interés y el centro de la misma para mostrarlos en la interfaz. En la parte de control, el centro calculado en la parte visual se emplea para mover el drone hasta que el centro del objeto se encuentre con el punto central de referencia.

5.1. EJECUCIÓN TÍPICA

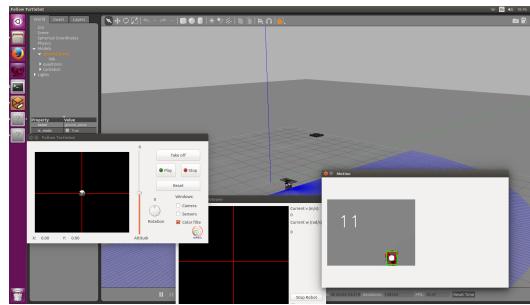


Figura 5.4: Objeto localizado. Inicio de algoritmos de control.

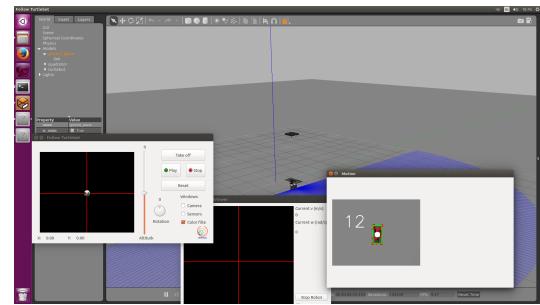


Figura 5.5: Objeto centrado.

En caso de que el objeto de interés se esté moviendo el drone intentará centralizarlo tal y como lo tiene programado, por lo que empezará a perseguir al objeto.

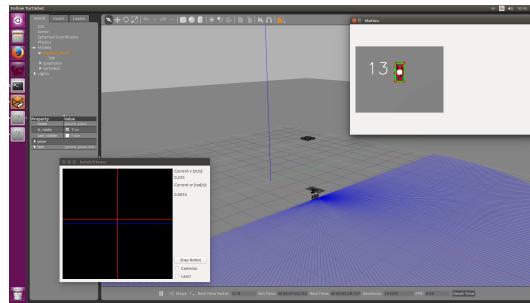


Figura 5.6: Drone persiguiendo al objeto. Inicio.

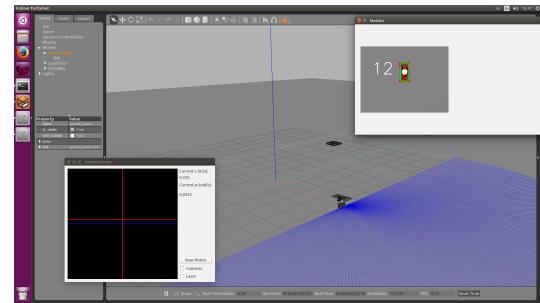


Figura 5.7: Drone persiguiendo al objeto.

Como se ha explicado anteriormente, en caso de perdida del objeto, el drone volverá al estado de búsqueda.

Se realizaron experimentos con diferentes objetos.

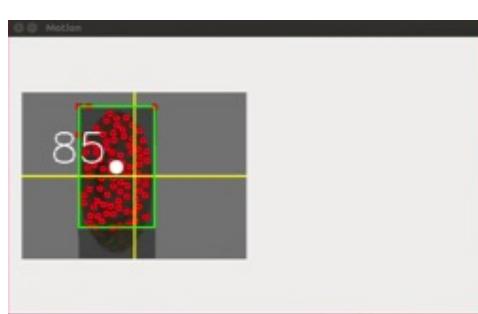


Figura 5.8: Objeto de prueba: sombrero

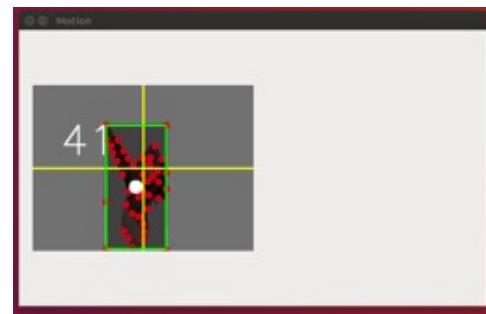


Figura 5.9: Objeto de prueba: persona

Debido a la diferencia entre que un objeto tiene más textura que el otro, como se puede observar en las figuras 5.8 y 5.9, los puntos soporte obtenidos en el que tiene más textura es mayor que en el otro. El número mostrado es el número de puntos soporte instantáneo.

5.2. Seguimiento visual de objetos dentro de una imagen

Además de los experimentos con cámaras simuladas, se estuvieron realizando experimentos con una cámara real para desarrollar y comprobar el funcionamiento del algoritmo y así validar la parte perceptiva del comportamiento global. El principal uso que se le dio al uso de la cámara real fue para probar que la función `GoodFeaturesToTrack` obtenía los puntos correctamente. Gracias a estos objetos de prueba se puede comprobar como la función obtiene los puntos principalmente en las esquinas.

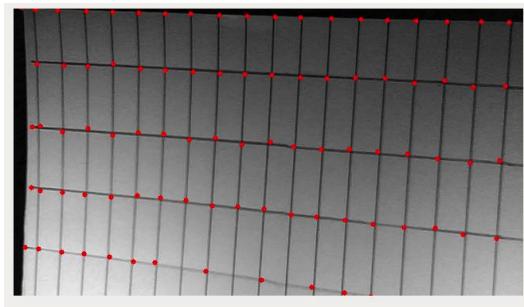


Figura 5.10: Objeto de prueba: cuadrícula



Figura 5.11: Objeto de prueba: imagen de una flor

Tal y como se muestra en las figuras 5.10 y 5.11, esta función extrae muy bien las esquinas como píxeles interesantes sobre los que hacer el seguimiento visual y consigue buen seguimiento en la imagen sobre esos píxeles.

Comprobado que `GoodFeaturesToTrack` funcionaba correctamente, algo que es indispensable para poder calcular el flujo óptico, se paso a experimentar con la función `calcOpticalFlowPyrLK` encargada de calcular el movimiento de los puntos de la imagen.

La función hace el seguimiento de los puntos calculados con un pequeño error pero aceptable para realizar el seguimiento del objeto por la imagen, tal y como se puede apreciar en las figuras 5.12 y 5.13.

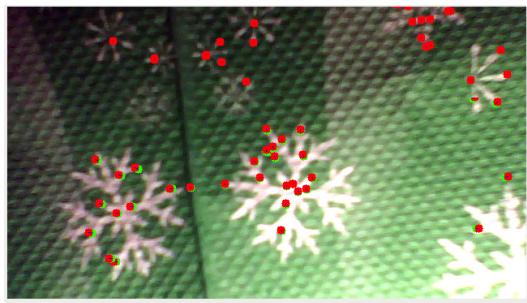


Figura 5.12: Objeto de prueba: copos de nieve

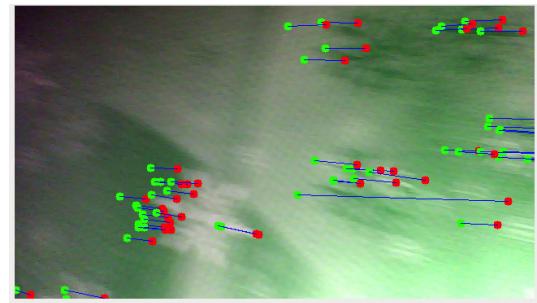


Figura 5.13: Objeto de prueba: copos de nieve en movimiento

Se probó también que se calcule correctamente la región de interés dentro de la cual estaría el objeto. Como se ha explicado, la región la definen los puntos más extremos del objeto.

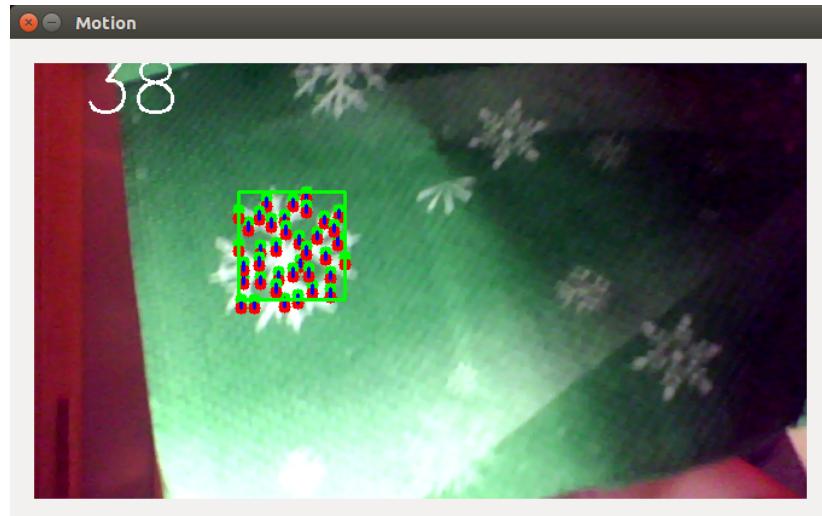


Figura 5.14: Objeto de prueba: copo de nieve con región de interés

El cálculo de la región de interés no es del todo perfecto, es decir, no toma el valor del punto real más extremo para su definición, pero la aproximación es muy buena con un error muy pequeño.

5.3. Detección manual del objeto

Se han hecho pruebas seleccionando la región de interés manualmente. Esto consistía en abrir una ventana con la imagen donde el usuario manualmente tenía que trazar un rectángulo sobre el objeto de interés para poder hacer el seguimiento visual. Fue reemplazado por la

detección completamente automática descrita en el capítulo 4.

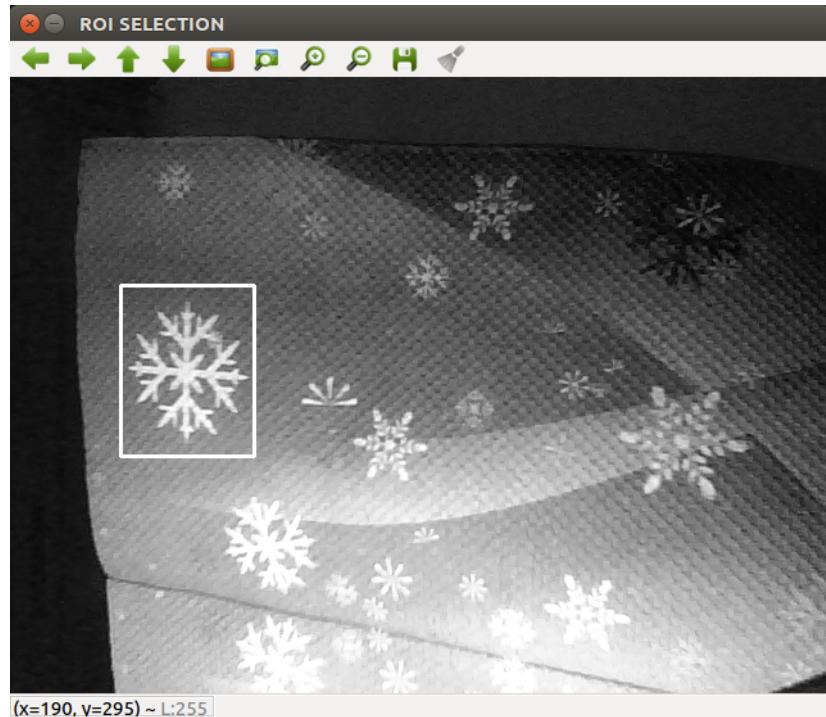


Figura 5.15: Objeto de prueba: copo de nieve seleccionando región de interés

5.4. Seguimiento desde el drone de objetos de colores

Además de estas pruebas con cámara real, se hicieron otras pruebas en el simulador Gazebo, realizando el seguimiento de un objeto con colores, que como se ha explicado antes basaba su búsqueda en si el objeto tenía el color relevante o no.

Los valores del color se obtenían de forma experimental gracias a la ayuda de un componente de JdeRobot llamado ColorTuner, con el cual se obtenían los valores que marcarían el rango en el espacio de color HSV con el cual se ajustaba el filtro que dejaría pasar los los colores que estén dentro de ese rango, para que, a continuación, la parte perceptiva detectase el objeto y pudiera darle la información necesaria a los algoritmos de control.

Esto es útil para simplificar la percepción y poder probar el seguimiento con estos objetos sencillos, antes de usar otros objetos más sofisticados que no tienen algo relevante, sólo textura.

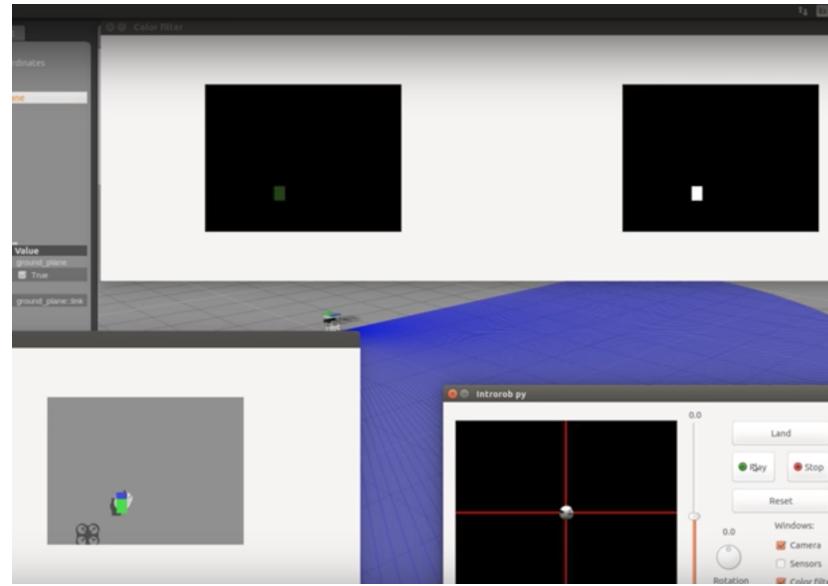


Figura 5.16: Objeto de prueba: robot terrestre con colores

Tal y como se aprecia en la figura 5.16, el filtro de color HSV está ajustado para el color verde. El filtro calcula que valores de la imagen están dentro del rango que esta ajustado y genera la máscara en blanco y negro (parte derecha superior de la figura). Esta máscara se aplica a la imagen, generando una imagen donde solo se puede ver el color que ha sido filtrado.

Capítulo **6**

Conclusiones y trabajos futuros

La visión artificial está ganando muchos adeptos en los últimos años debido al gran potencial que tiene. Lo interesante de esta tecnología es que algo tan normal para los humanos como es ver y procesar la información recibida por nuestros ojos, para las máquinas supone un reto. Gracias al desarrollo de este trabajo de fin de grado, se ha aprendido paso a paso cómo las máquinas son capaces mediante algoritmos de procesar una información recibida por una cámara y realizar una acción, así como la capacidad de tomar decisiones de manera autónoma.

6.1. Conclusiones

Los objetivos planteados en el capítulo 2 se han desarrollado tal y como se ha explicado en el capítulo 4. Estos algoritmos desarrollados son capaces de realizar una percepción visual aplicando desde filtros de color al seguimiento de puntos de interés en objetos con textura. Además, se han desarrollado también unos algoritmos capaces de realizar una búsqueda y un seguimiento de un objeto. Como se explica en el capítulo 5, los experimentos que se han hecho con el componente validan su funcionamiento, ya que cumple con los objetivos planteados.

Ha sido necesario aprender como ven las máquinas y que son capaces de hacer con esa información. De ahí que las primeras pruebas que se hicieran fuesen con objetos simples, formas y colores sencillos en lugar de objetos más complejos. Una de las partes más importantes del presente trabajo de fin de grado fue el aprendizaje del procesamiento básico de imágenes utilizando la biblioteca OpenCV y aplicar los datos del mismo a unos mecanismos de control.

Después de ese aprendizaje básico, las formas básicas de los objetos pasan a ser puntos en una imagen real gracias a las características que tienen las texturas. Aquí es donde este trabajo de fin de grado tiene su aportación a este campo. Aplicando las funciones de OpenCV, `cv2.goodFeaturesToTrack()` y `cv2.calcOpticalFlowPyrLK()` (véase la sección 4.2), se ha podido hacer un seguimiento visual de estos puntos que definen unas características, las cuales a su vez describen el objeto, por lo tanto, se hace un seguimiento visual del objeto.

Se han tratado los datos que las funciones de percepción visual ofrecían, y se han desarrollado algoritmos acordes al objetivo sobre el que se centraba el tema del trabajo, generando una salida a unos datos de entrada. El desarrollo de algoritmos para realizar la búsqueda, donde el drone se desplaza por el escenario siguiendo un patrón, y seguimiento del objeto dentro del escenario de Gazebo, gracias a los datos extraídos de la imagen (véase la sección 4.3) y haciendo uso de la función de control `cmdVel.sendCMDvel` como director de vuelo del drone, hace que se completen los objetivos marcados inicialmente.

En la parte experimental se valida que ahora el componente es capaz de seguir un objeto más complejo frente a los antecedentes que existían previos a este desarrollo, ya que ahora es capaz de seguir objetos con una textura frente a un objetos con un color llamativo. Esto hace que funcione con una mayor variedad de objetos, acercando el comportamiento a objetos más realistas. En esta fase se hicieron una gran cantidad de pruebas una por una de las partes que componen el componente en su versión final (véase el capítulo 5).

Además de que el desarrollo cumple con los objetivos inicialmente marcados, cumple también con los requisitos que se definieron. El componente funciona con la versión 5.5 de JdeRobot junto con la versión 7.4 de Gazebo. El procesamiento de las imágenes es rápido y permite al drone realizar correctamente el seguimiento (véase la sección 5.1). En ocasiones la velocidad de procesamiento es más lenta, depende de la capacidad que tenga la máquina sobre la que se está simulando y ejecutando el componente. Pese a que hay errores en el cálculo de los puntos (véase la sección 5.2), son mínimos, por lo que el componente es robusto y minimiza los errores ejecutándose de una manera correcta. Y por último, cumple con el requisito de ser capaz de seguir diferentes objetos (sección 5.1, figuras 5.8 y 5.9).

Durante el desarrollo del trabajo se han podido emplear diferentes bibliotecas funcionando con el lenguaje de programación Python, el cual se ha aprendido y se ha adquirido un buen nivel, pudiendo llevar a la práctica conceptos que previamente eran solo

teóricos, lo que ha permitido ampliar el conocimiento sobre visión artificial y el poder conocer más en profundidad los algoritmos empleados, logrando así desarrollar algoritmos de percepción para objetos complejos. Ademas, ha habido un aprendizaje de la plataforma JdeRobot, la cual ha permitido una ampliación importante del conocimiento sobre el mundo de la robótica y el control de robots y drones, y como estos se relacionan con el entorno.

En mi opinión, este aprendizaje ha sido muy útil y muy interesante para el desarrollo de nuevas tecnologías de visión artificial, ya no solo centradas en drones, si no en cualquier aspecto de la robótica que disponga de una cámara y una capacidad de proceso que permita ejecutar este tipo de algoritmos.

6.2. Trabajos futuros

Sin salirse de la temática de un drone persiguiendo a un objeto con textura, podemos desarrollar aún más este trabajo añadiendo una clasificación de características, es decir, haciendo que después de que este obtenga las características, clasifique los objetos en función de los datos que haya obtenido, para de este modo crear una base de datos con objetos predefinidos. Con esto se pueden desarrollar algoritmos de reconocimiento de objetos por patrones o tipos de objetos.

También cabe la posibilidad de desarrollar redes neuronales o con técnicas de SVM para que haya un aprendizaje más profundo para detección y clasificación de objetos, pudiendo dotar al drone de una capacidad para tomar decisiones más complejas.

En la parte del simulador Gazebo, se pueden desarrollar y se están desarrollando escenario más realistas, y empleando texturas dará una sensación de que esas pruebas están realizadas de la manera más próxima a la realidad y así poder hacer el seguimiento de personas y vehículos.

Respecto a los drones, la primera línea futura a seguir es llevar el desarrollo hecho en el simulador al mundo real y probarlo con el drone real ArDrone2 de Parrot. También se pueden desarrollar soporte para más modelos en el mercado, consiguiendo que haya más variedad de drones disponibles para ejecutar el componente. Actualmente, con JdeRobot, cabe destacar el trabajo de Diego Jiménez Bravo¹, que está desarrollando soporte para el modelo 3DR Solo

¹jderobot.org/Jimenez-tfg

Drone y el trabajo de Jose Antonio Fernández², cuyo tema se centra en desarrollar soporte en un avión.

Dejando de lado un poco el tema sobre el que se ha fundamentado el trabajo, y teniendo en cuenta que el tema de la infraestructura ha sido explicado aquí, un trabajo futuro es desarrollar imágenes de los componentes para que puedan ser plug and play, es decir, que tenga todo lo necesario para que el usuario pueda ejecutar los algoritmos sin que nada más sea necesario previamente, por ejemplo, poniendo esas imágenes en servicios cloud o haciendo despliegues mediante contenedores.

Por último, el que más abanico de posibilidades da para desarrollo al ser más general, es la visión artificial, en la cual se pueden desarrollar, por ejemplo, un reconocimiento facial y de gestos, y aplicar esta tecnología no solo a drones sino también a sistemas de seguridad y control de maquinaria mediante la cual un coche podría ser conducido solamente con la mirada, apuntando con los ojos allá donde el conductor desea ir. También se puede desarrollar un sistema de detección de enfermedades en la piel mediante un reconocimiento de la textura que estas presenten, otorgando a los pacientes un método de diagnóstico sin ni siquiera tener que asistir al médico. Podemos aplicar también el análisis de texturas en el mundo de las infraestructuras, por ejemplo, detectando en una pared si los ladrillos han sido correctamente colocados y si se encuentran en buen estado.

Esto son solo un ejemplo de las posibilidades de desarrollo que existen en el mundo de la visión artificial, de los drones y de los simuladores de entornos virtuales, pero las posibilidades de desarrollo de diferentes aplicaciones son casi infinitas.

²jderobot.org/Jafernandez