



Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA TELEMÁTICA
TRABAJO FIN DE GRADO

**Aterrizaje de un drone sobre una
baliza móvil usando visión**

Autor: Andres de Jesús Hernández Escobar
Tutor: Jose María Cañas Plaza

CURSO ACADÉMICO 2014/2015

A los que...

Agradecimientos

Quiero agradecer...

Contenido

Resumen	IX
1. Introducción	1
1.1. Robótica	1
1.1.1. Historia	2
1.1.2. Aplicaciones actuales	4
1.1.3. Software en robots	5
1.2. Visión en robots	6
1.2.1. Aplicaciones actuales	6
1.3. Simulación	7
1.4. Robótica Aérea	8
1.4.1. Historia	8
1.4.2. Robótica Aérea en la Universidad Rey Juan Carlos	12
2. Objetivos	15
2.1. Tareas que cumplir	15
2.2. Requisitos	15
2.3. Metodología	15
2.4. Planificación	16
3. Infraestructura	19
3.1. Interfaz Gráfica de usuario	19
3.1.1. Qt	19
3.1.2. Matplotlib	20
3.2. ICE	20
3.3. JdeRobot	20
3.3.1. Teleoperadores uav_viewer e introrob_py	20
3.4. Gazebo	21
3.4.1. Plugins	22
3.5. Blender	23
3.6. Visualización virtual	23
3.6.1. AprilTags	23
3.6.2. OpenCV	24
3.7. Dispositivos físicos	24
3.7.1. MK802IV	24

3.7.2. Ar.Drone 2	24
4. Desarrollo de la infraestructura	25
4.1. Coche virtual y teleoperador	25
4.2. Mk802IV	25
4.3. Distance Viewer	25
5. Aterrizaje visual	27
5.1. Control visual	27
6. Conclusiones y líneas futuras	29
6.1. Conclusiones	29
6.1.1. Título de la primera subsección	29
6.2. Líneas futuras	29
Bibliografía	31

RESUMEN

En el presente Proyecto Fin de Carrera...

CAPÍTULO 1

INTRODUCCIÓN

En este primer capítulo, se introducirá al lector en el mundo de la robótica. Un campo muy conocido en la cultura popular y el género de ficción, pero muy poco acertado sobre su verdadera identidad. Muchos son los mitos y creencias que hay sobre los robots. Sin embargo, en este apartado se pretende mostrar el contexto y la verdadera aplicación que tienen los robots en el día a día. Al mismo tiempo, se definirá y se aportará información relativa con este Trabajo de Fin de grado, para una buena comprensión del contenido.

1.1. Robótica

La robótica es la ciencia y la técnica que está involucrada en el diseño, la fabricación y la utilización de robots. Un robot es una máquina que puede programarse para que interactúe con objetos y lograr un objetivo, como imitar el comportamiento humano o la sustitución de una persona en un entorno peligroso.

Para comprender mejor que es un robot, es necesario conocer que partes lo forman y los posibles comportamientos para los que ha sido diseñado.

Un sensor es un dispositivo eléctrico y/o mecánico que convierte magnitudes físicas (luz, electricidad, presión, etcétera) en valores medibles de dicha magnitud. Si utilizamos como analogía la anatomía humana, son equivalentes a los sentidos del cuerpo humano, como la vista o el oído. Generan la información, que una vez procesada, produce una acción, normalmente en los actuadores. Por ejemplo, con sensores de temperatura se puede medir el número de grados Celsius en una habitación.

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en energía mecánica. Un ejemplo es un motor eléctrico que transforma electricidad en un movimiento rotacional para girar una rueda. Hay diferentes tipos según su naturaleza, y limitan las acciones que puede realizar un robot con el mundo que le rodea. Siguiendo con la analogía anterior, el actuador se correspondería a los músculos y articulaciones que componen un cuerpo humano.

Por último, un robot está formado por procesadores, que obtienen datos de los sensores y que se encargan de materializar acciones en los actuadores. El software se encarga procesar los datos de entrada y generar un comportamiento mediante la consecución de acciones. Volviendo a la analogía con el ser humano, sería nuestro cerebro y nervios.

Los robots pueden ser o no autónomos. Por autonomía se entiende la habilidad para

tomar decisiones por uno mismo y llevarlas a cabo. Esto en un robot, es la capacidad de percibir la situación y actuar apropiadamente. Mediante acciones enviadas a los actuadores, se puede desplegar el comportamiento que realiza un robot. Éste, es definido por las decisiones tomadas durante el procesamiento de los datos de los sensores.

En caso de carecer de autonomía, se puede interaccionar con el robot mediante la teleoperación y la telepresencia. La teleoperación es la manipulación y envío de órdenes para ser ejecutadas por un robot que se encuentra en un lugar diferente a la persona. En medicina, se utiliza para realizar operaciones a través de unos brazos que ejecutan los movimientos enviados desde un lugar lejano. La telepresencia es la obtención de datos de los sensores de forma remota. En el espacio exterior, se aplica a la hora de enviar órdenes a satélites o robots como el Curiosity en Marte.

El comportamiento ha de ser en *tiempo real*, incluyendo la toma de decisiones y el análisis de diferentes situaciones. Además, *robusto* para evitar posibles accidentes o resultados no esperados. Otra forma de generar un comportamiento es por medio de sistemas distribuidos. En estos, los diferentes robots colaboran juntos y se comunican sin intervención humana para la obtención de un objetivo en común.

Uno de los objetivos para el futuro de la robótica, es la multitarea. Hoy en día un robot está diseñado para un número limitado de posibles trabajos o tareas, a diferencia de los seres humanos, los cuales nos adaptamos sin necesidad de cambiar nuestra naturaleza física. Esto será posible con la evolución de inteligencias artificiales más capaces, actualmente muy limitadas.

(a) *Curiosity* en Marte

(b) Teleoperación médica

Figura 1.1. Ejemplos de robots

1.1.1. Historia

El origen etimológico del término *robot*, comúnmente utilizado hoy en día, tiene como origen la obra con título *R.U.R*, que es la abreviatura de Rossum's Universal Robots. El escritor de origen checoeslovaco, Karel Čapek, inventó la palabra *roboťa* (labor, trabajo) y cuya raíz eslava *rabu* coincide con esclavo. La palabra robot a difundida gracias a numerosos autores de éxito en norteamérica, como por ejemplo Isaac Asimov.

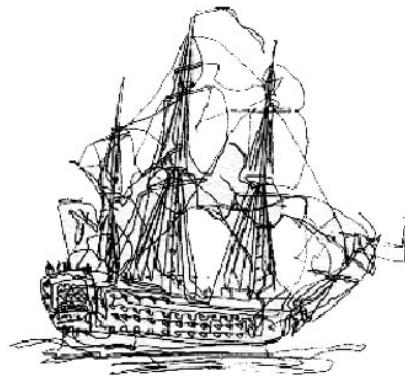
Aunque el origen de los robots es anterior a ésta novela de ciencia ficción de los años 20. Sus predecesores son los autómatas. La palabra *autómata*, de origen griego, se utiliza según la *Real Academia Española (RAE)* para designar a un instrumento o aparato

que encierra dentro de sí el mecanismo que le imprime determinados movimientos. Son artefactos, creados por el hombre, capaces de realizar tareas diarias y comunes para los hombres, o bien, para facilitar las labores cotidianas. Aunque no todos tenían una utilidad, algunas de ellos servían para entretenir a sus dueños.

Un ejemplo es el *hombre de palo*(1500) para el emperador Carlos V en España. Juanelo Turriano construyó este autómata con forma de monje. Andaba y movía la cabeza, ojos, boca y brazos. Conforme pasaron los siglos, la complejidad mecánica fue aumentando hasta llegar por ejemplo, al autómata de Maillardet(1800) en Londres. Con una memoria muy superior a cualquier máquina de la época, era capaz de dibujar cuatro dibujos y escribir tres poemas (dos en francés y uno en inglés) en un papel.



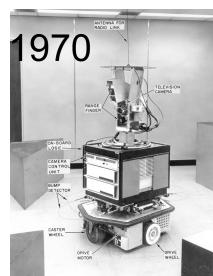
(a) Autómata



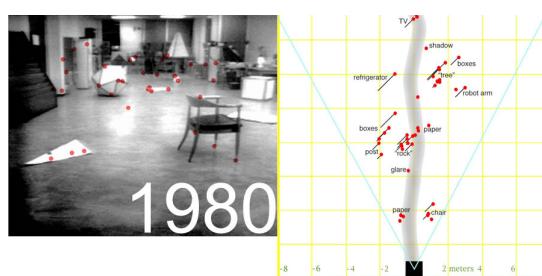
(b) Dibujo en papel

Figura 1.2. Autómata de Maillardet

Entre los años 1960 y 1980, la aparición de los primeros ordenadores revolucionan los autómatas, llegando por primera vez la robótica a las universidades. Un ejemplo es el Stanford Cart. Se utilizó para demostrar que no se podía controlar de forma fiable un vehículo no tripulado en la Luna. Utilizaba cuatro ruedas con motores eléctricos y una cámara. La siguiente evolución de este carrito fue con el logro del primer robot con ruedas y visión autónomo. Hasta que en 1980, utilizando dos cámaras, se comienza a reconstruir el espacio en 3D mediante la construcción de mapas.



(a) Stanford Cart



(b) Reconstrucción del espacio 3D en mapas

Figura 1.3. Robot de Stanford Cart

Más adelante, con la aparición de nuevos sistemas operativos y ordenadores mucho más potentes, las aplicaciones de la robótica son cada vez más extensas y se populariza la interacción entre robots y humanos. La compañía de juguetes LEGO lanza MINDSTORMS. Consistía en un equipo de desarrollo para robots, basado en piezas de LEGO como estructura y compuesto por un procesador y varios sensores y actuadores. Por otro lado, Sony pone al mercado el primer Aibo, el perro robot. Un juguete pensado para interaccionar con niños y simular el comportamiento de una mascota. Los humanoides también se popularizan como por ejemplo ASIMO. Este robot es presentado por primera vez en el año 2000 por la empresa japonesa Honda. Es una demostración del potencial de la robótica y un escapárate para el mundo de lo que se puede hacer. Actualmente sigue sirviendo como exhibición de la tecnología empleada por Honda y es capaz de acciones como dar la mano a otra persona, subir escaleras y coger un vaso de agua sin romperlo, entre muchas otras.

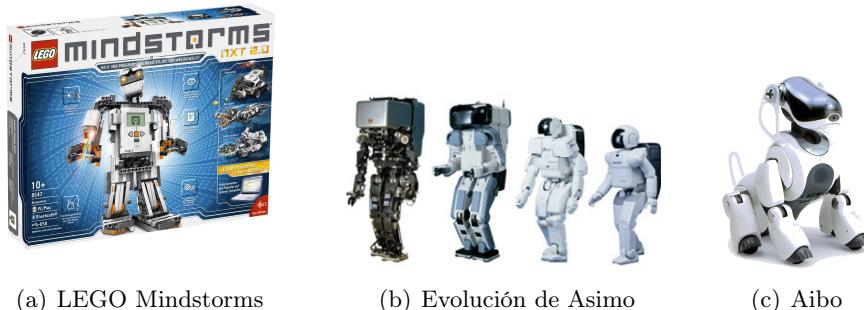


Figura 1.4. Robots en el siglo XX

1.1.2. Aplicaciones actuales

Hoy en día, las aplicaciones de los robots son muy diversas. Fuera y dentro de nuestro planeta, los robots permiten hacernos ver sitios en los que el hombre no puede llegar o en los que el hábitat es hostil. Como el Global Explorer ROV, que se ha sumergido en diferentes océanos para obtener imágenes nunca antes vistas por el hombre.

Sector industrial

Es uno de los sectores que más compra robots y se encuentra en constante crecimiento. China está a punto de pasar a EEUU en densidad de robots por trabajador y en España el uso de robots en empresas aumentó un 38 % en 2013¹.

Dentro de las aplicaciones encontramos:

- Operaciones de manipulación: Usan pinzas, colaboran con otros robots y/u operarios y desplazan objetos. Este uso es uno de los más extendidos en empresas. Por ejemplo en el montaje de coches, ayudando a operarios a desplazar objetos pesados como las puertas.

¹Datos obtenidos de International Federation of Robotics <http://www.ifr.org/>

- Soldadores. Se encargan de las tareas de soldadura de componentes. La compañía Asus ha creado un método de producción automático para sus tarjetas gráficas. En concreto, este proceso de soldadura mejora la calidad del producto y permite reducir el tamaño de sus tarjetas notablemente.¹
- Montaje. Las cadenas de montaje se vuelven más rápidas y eficientes. En las plantas de procesadores Intel, el proceso de montaje es uno de los más avanzados del mundo y se utilizan salas en las que el aire no es respirable para personas y garantizan un área de partículas externas muy baja, muy importante para la pureza de los procesadores.

Aplicaciones comerciales

Su uso está en constante crecimiento y el número de aplicaciones es muy variado, desde recreación pasando por la grabación profesional para cine. A continuación, se recogen algunas de las aplicaciones que tienen lugar en la actualidad:

- Limpieza doméstica: Incluye robots que limpian piscinas, hasta aspiradoras inteligentes. Este último caso es el de Roomba(Figura 1.5), de la compañía iRobot, que incorpora algoritmos de construcción de mapas, evasión de objetos o incluso detección de escaleras (para evitar posibles accidentes).
- Transporte de personas:El mayor representante de este tipo de productos es el Google Car(Figura 1.5). Este proyecto, en funcionamiento desde 2009, ofrece a personas con movilidad reducida o discapacitados, la utilización de automóviles, en concreto un coche. Está equipado con todo tipo de sensores que permiten la autolocalización, evitar accidentes y llegar al destino deseado. Dota de mayor autonomía a estas personas, mejorando su calidad de vida.
- Seguridad y automatización en vehículos: En esta categoría se recogen sistemas de seguridad que controlan la distancia de seguridad respecto de otros vehículos, luces inteligentes e incluso sistema de aparcamiento asistido. Compañías como BMW, Honda o Mercedes Benz apuestan cada día más por la utilización de estas tecnologías en la industria de automoción.
- Grabación aérea: La utilización de drones con la capacidad de volar, ha reducido considerablemente el coste de planos aéreos y simplificado el equipo necesario. Airdog(Figura 1.5) es un proyecto nacido de una camapaña Kickstarter que permite el seguimiento de actividades deportivas o recreativas a gran velocidad, de forma totalmente autónoma, y la grabación de las mismas.

1.1.3. Software en robots

El software que se encuentra en los robots, es el encargado de dotar de un comportamiento inteligente a los mismos. Se pueden distinguir tres tipos software en un robot. Uno, es el sistema operativo que se ocupa del procesador. El segundo es el que se encarga, por

¹ Introducing ASUS Auto-Extreme Technology: <https://www.youtube.com/watch?v=4gRpuurPsuc>

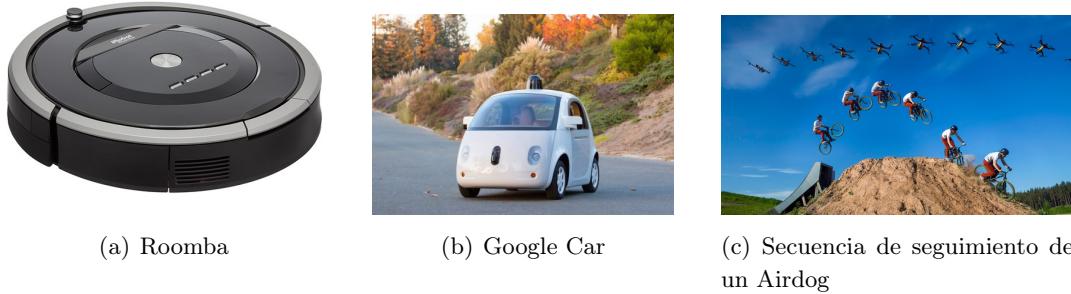


Figura 1.5. Aplicaciones en la actualidad

ejemplo, de calcular el movimiento necesario de cada actuador basándose en algoritmos. Se caracteriza porque puede contener diferentes niveles de lenguaje. Desde unos de alto nivel, como el que utilizaría cualquier programa que ejecuta en un ordenador, hasta otros de bajo nivel, como el código máquina. El último grupo es la colección de programas y funciones desarrolladas para utilizar los periféricos del robot. Dicha colección, abstrae los dos grupos anteriormente explicados en beneficio de una programación más fácil.

1.2. Visión en robots

La visión, al igual que en los seres humanos, es una gran fuente de información de nuestro entorno. Las personas utilizamos el espectro visible de la luz, que se corresponde con lo que llamamos colores. Además, percibimos el mundo que nos rodea como un mundo en tres dimensiones debido a que tenemos dos ojos. Esto nos permite obtener propiedades del entorno y poder desplazarnos en él sin preocupaciones. Los robots utilizan sensores de visión de todo tipo, capaces de ver en la oscuridad o distinguir diferentes temperaturas. Además, en los últimos años, los sensores de visión han disminuido en precio y su utilización se ha visto incrementada notablemente en el mundo de la robótica. A pesar de este incremento en su uso y de ser una fuente potencialmente rica en información, el flujo de datos puede llegar a ser muy grande y su procesado complicado.

El objetivo de la visión artificial es conseguir datos a través de las imágenes que recibe de una cámara. Para esto se aplican diferentes operaciones matemáticas con el fin de conseguir bordes, formas, colores, patrones en la imagen, etcétera.

1.2.1. Aplicaciones actuales

Actualmente, las aplicaciones de visión artificial son muy variadas, desde seguridad, como sistemas de detección de movimiento, pasando por el entretenimiento, como el sensor Kinect, hasta la accesibilidad para dotar de autonomía a personas que lo necesitan. Es muy común, hoy en día, que un robot incluya como parte de sus sensores una cámara.

- Navegación y construcción de mapas: Es una de las primeras aplicaciones a través de visión y permite a los robots la creación de mapas a través de la detección de bordes, formas o profundidad. Ésta información sirve también para poder navegar sobre sitios desconocidos o previamente han sido convertidos a un mapa. Un ejem-

plano hoy en día el *Darpa Robotics Challenge*², en el que la visión es un elemento crucial para poder esquivar objetos y realizar mapas del terreno.

- Autolocalización: Permite extraer información a un robot sobre la posición relativa respecto al resto del mundo que lo rodea, mediante el reconocimiento de patrones o balizas. Una técnica es el SLAM (*Simultaneous Localization and Mapping*), que permite la autolocalización al mismo tiempo que se realiza un mapa del entorno.

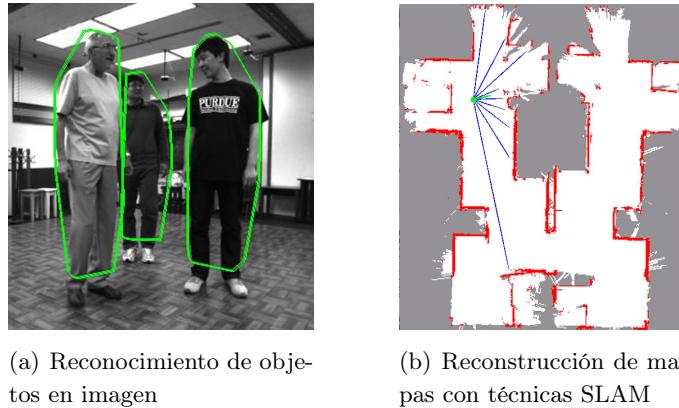


Figura 1.6. Visión artificial en robots

1.3. Simulación

Una parte importante del diseño del comportamiento de un robot, es la simulación. Proporciona un entorno virtual en el que somete a diferentes escenarios y situaciones a un modelo creado a partir del robot elegido. Permite probar algoritmos sin necesidad de utilizar uno real. Puede llegar a aportar información muy valiosa y familiarizarse con posibles situaciones. Además de prevenir accidentes como cualquier daño físico al objeto o herir a las personas cercanas. El problema que se presenta es que los resultados dependen de la precisión a la hora de caracterizar el modelo y mundo virtual. Una vez alcanzado el resultado deseado, se ha de adaptar nuestra aplicación robótica al mundo real.

Son necesarios motores de física como Bullet, que simplifican el cálculo y la representación de físicas reales en tiempo real para la interacción entre objetos.

Todo esto no sería posible sin un motor gráfico, que dibuje en pantalla todo lo que está ocurriendo. Se ocupan de darrealismo y representar en imágenes el mundo virtual con el que estamos trabajando. Un ejemplo es Ogre u OpenGL, que dan una gran variedad de opciones, como la renderización de texturas y sombras.

Para una mayor precisión del comportamiento en el mundo real, los simuladores incluyen la adición de funciones de ruido en sensores y actuadores. Esto permite la creación de comportamientos mucho más robustos.

²Web oficial de Darpa Robotics Challenge <http://www.theroboticschallenge.org/>

Un ejemplo es Gazebo³, un proyecto de software libre que incluye multitud de modelos y motores de física virtualizada. Ofrece una interfaz gráfica y control sobre los objetos y el mundo generado, además de la creación y modificación de actuadores y sensores personalizados. Por ejemplo, se pueden crear vehículos con diferentes sensores o casas con las que interactuar.

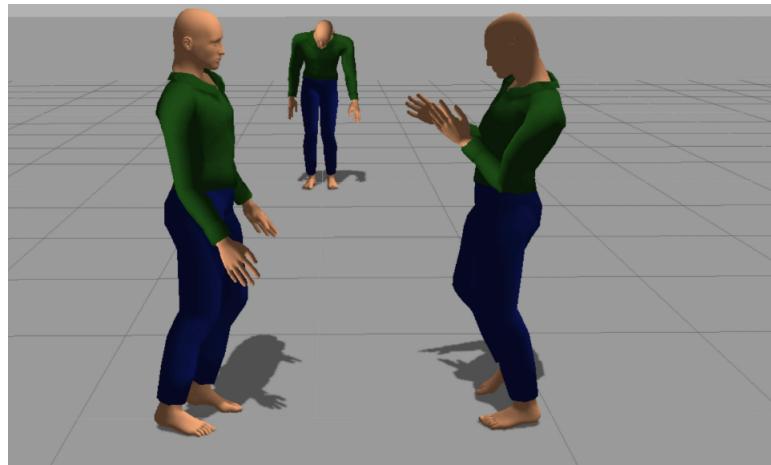


Figura 1.7. Personas simuladas en Gazebo

El desarrollo de Gazebo comenzó en 2002 en la Universidad de California del Sur por Andrew Howard y Nate Koenig. A partir del concepto de un simulador de alta precisión para poder simular robots en entornos exteriores bajo diferentes circunstancias. El nombre elegido fue Gazebo ya que es la estructura más cercana a un escenario de exteriores.

A pesar del origen de este nombre, hoy en día se utiliza para la simulación de interiores también. Es uno de los componentes oficiales en *Virtual Robotics Challenge* organizado por *DARPA Robotics Challenge*(DRC). Actualmente su desarrollo se encuentra bajo el *Open Source Robotics Foundation*(OSRF) con soporte de una variada y activa comunidad. Por último, es utilizado como herramienta de virtualización oficial para la plataforma *JdeRobot* de la Universidad Rey Juan Carlos.

1.4. Robótica Aérea

Es una de las ramas de la robótica de mayor auge actualmente y sus aplicaciones son cada vez más extendidas. Pertenecen a este área los *Unmanned Aircraft Vehicle*(UAV), en español *Vehículo Aéreo No Tripulado*(VANT), o también conocidos como *drones*. Se trata de un vehículo capaz de volar, que puede o no recibir órdenes del exterior. Incluyen multitud de diferentes sensores para mantenerse en vuelo, aterrizar o despegar.

1.4.1. Historia

Históricamente, el origen de los *UAV* ha sido en aplicaciones militares, como en otras áreas de investigación. Una vez ha sido suficientemente desarrollado comienzan las aplicaciones civiles y su aplicación comercial e industrial. En 1883, Douglas Archibald instaló

³Página web oficial de Gazebo : <http://gazebosim.org/>



Figura 1.8. Robot utilizado en DRC

un anemómetro, un instrumento para medir la velocidad del viento, a su cometa para poder medir la velocidad del viento a una altura de más de 350 metros. En 1887 instaló dos cámaras, de las cuales tomó fotografías una vez en el aire. Se consideran las primeras imágenes tomadas por un *UAV*. Más tarde se usaría esta técnica en la guerra entre España y los Estados Unidos en 1898 para obtener datos estratégicos.

Durante la primera y la segunda guerra mundial se utilizaron drones para la obtención de mapas sin poner en peligro al piloto. Más tarde, en 1995 se utilizaron en Bosnia para tareas de vigilancia o análisis de daños. Especialmente importante para el reconocimiento nocturno. El modelo se llamaba *Predator* y era lanzado desde Hungría.



Figura 1.9. UAV Predator.

Aplicaciones actuales

Actualmente, gracias al avance de la estabilización electrónica, los *UAV* han alcanzado tamaños mucho más reducidos, como el *Hummingbird*⁴ o colibrí en español, de DARPA. Además son mucho más ágiles y mecánicamente simples. Han aparecido numerosos usos comerciales y civiles, aunque no desaparece el interés militar. Compañías como *Amazon*, están trabajando en proyectos para conseguir crear un sistema de envío de compra a domicilio utilizando *drones*, en concreto *cuadricópteros*. En universidades como *University of Pennsylvania* están diseñando comportamientos basados en grupos masivos, creando formaciones en el aire y dotando capacidad de pensamiento en grupo a los drones.

⁴Más información en: <http://www.avinc.com/nano>

Otro de los usos es la exploración aérea, que incluye la inspección de embalses, líneas de alta tensión, campos agrícolas y la vigilancia. Este último caso es el de Alemania, que utiliza drones aereos para evitar el ataque de grafiteros a vagones de tren⁵. Uno de los campeonatos más recientes de programación para *UAV* es el *Mohamed Bin Zayed International Robotics Challenge*(MBZIRC)⁶. Con una recompensa de 5 millones de dólares, una de las pruebas consiste en localizar, seguir y aterrizar, coincidiendo con los objetivos principales de este Trabajo Fin de Grado.



(a) Amazon Prime

(b) Grupos masivos

Figura 1.10. Ejemplos de UAV civiles

Cuadricópteros

Existen diferentes tipos de *drones* en función del diseño y los componentes que los forman. Algunos son similares a los aviones, con alas y el mismo método de despegue y aterrizaje. Están pensados para largos períodos de tiempo y altas velocidades. Otros, buscan una excepcional maniobrabilidad y estabilidad aérea. En este caso utilizan rotores, al igual que los helicópteros. A este grupo pertenecen los *cuadricópteros*. Se caracterizan por ser un helicóptero multi-rotor de cuatro brazos en forma de cruz. Los rotores, se encuentran en el extremo de cada brazo.

Cuando los motores giran, las hélices situadas en ellos generan un fuerza de empuje vertical, llamada *sustentación*. Ésta es perpendicular al movimiento de la hélice y depende de la velocidad a la que gira. La suma de cada fuerza en cada rotor produce una resultante. Los diferentes movimientos que puede describir el cuadricóptero se encuentran recogidos en la ilustración de la Figura1.11. El color rojo indica que una potencia mayor ha sido aplicada, mientras que el verde, respresenta una potencia menor. Para evitar un fenómeno que en los helicópteros produce vueltas sobre sí mismo, la disposición de los motores sigue una forma de cruz, en la que cada par opuesto gira en el mismo sentido. Uno en el sentido de las agujas del reloj y el otro anti-horario.

Para que sea posible el despegue (Figura1.11,e), esta resultante ha de ser superior al peso del *UAV*. Si es igual, se consigue un estado de altitud fija (también conocida en inglés como *hovering*). Para aterrizar sería necesario una resultante menor que el peso del objeto (Figura1.11,f).

⁵ Alemania pone a prueba drones contra los grafitis: http://www.bbc.com/mundo/noticias/2013/05/130528_tecnologia_drones_graffiti_alemania_aa

⁶Página Web oficial del campeonato: <http://www.mbzirc.com/>

Para conseguir el giro conocido como *yaw* (Figura 1.11,g y h) o *guiñada*, es el giro del plano horizontal al *drone*. Para girar a la derecha se transmite más potencia al par de rotores que giran en sentido anti-horario. Si la potencia fuera superior en el otro par opuesto, giraría hacia la izquierda sobre sí mismo.

En el supuesto de que sólo uno de los motores aplicase más potencia que los demás, por ejemplo el delantero, el *cuadricóptero* se desplazaría hacia atrás, inclinando la parte trasera del vehículo hacia arriba. Esto se correspondería con el movimiento llamado *pitch* (Figura 1.11,a y b) o *cabeceo*.

Por último, si aumentamos la potencia en uno de los motores laterales, por ejemplo la derecha, el vehículo se inclinará y trasladará hacia la izquierda, provocando un movimiento conocido como *roll* (Figura 1.11,c y d) o *alabeo*.

Comportamiento de los rotores

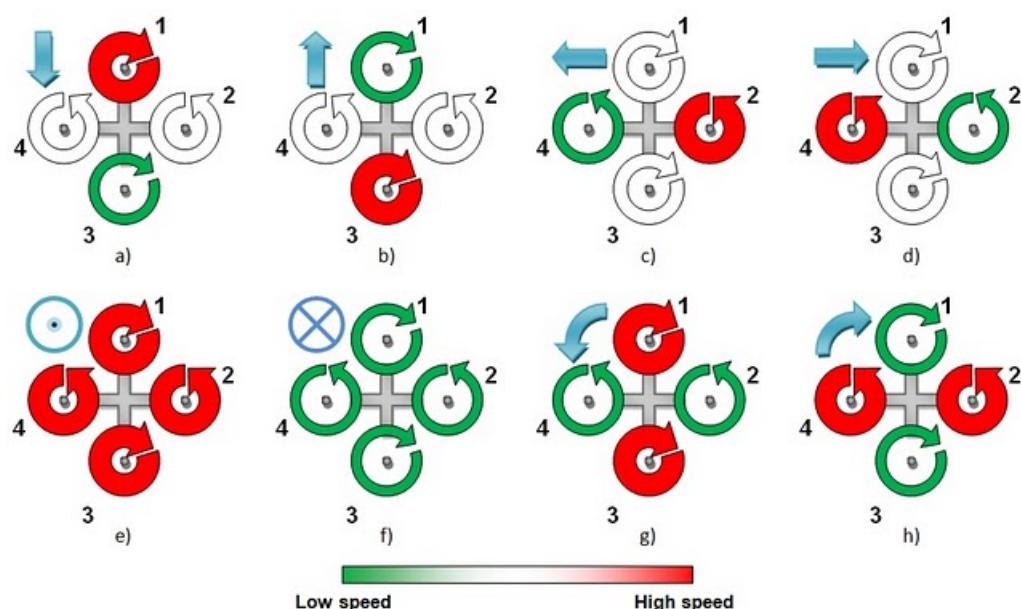


Figura 1.11. Relación entre la potencia de los rotores y el movimiento de un cuadricóptero

Los cuadricópteros pueden tener numerosos sensores desde acelerómetros, giroscopios, magnetómetros, ultrasonidos, incluso cámaras con resoluciones de hasta 4K. Todo ello se utiliza en combinación, para conseguir una mayor estabilización durante el vuelo. Dentro de los actuadores encontramos los cuatro rotores pero también se pueden añadir pinzas para cargar objetos o, en caso de uso militar, armas y sus respectivos gatillos.

Algunos de los fabricantes de drones más relevantes actualmente son:

- Parrot: Con modelos como el Ar.Drone 1 y 2 que acercaron a un gran público el uso de los drones.
- 3DRobotics: Diseñado para la obtención de planos aéreos estables.
- Erle: Está soportado oficialmente por Ubuntu.

- Airdog: Ya mencionado previamente en el apartado ??

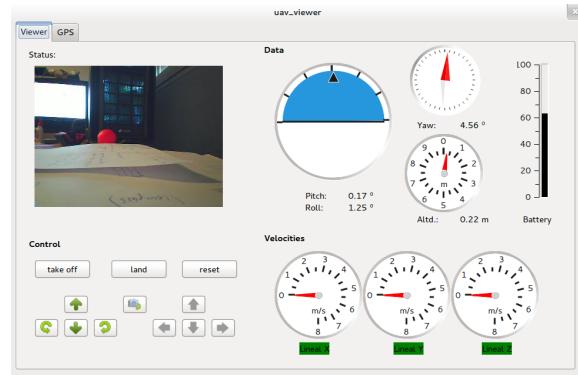
1.4.2. Robótica Aérea en la Universidad Rey Juan Carlos

El grupo de Robótica de la Universidad Rey Juan Carlos lleva años desarrollando proyectos relacionados con la navegación, visión, autolocalización y virtualización de entornos con robots. Gracias a la popularización y a la reducción en coste de los *drones* se comenzó en el año 2013 una nueva línea de investigación sobre los *UAV*. Los primeros proyectos han creado las bases sobre las que seguir investigando y han proporcionado la infraestructura necesaria. Éstos han sido integrados en la plataforma *JdeRobot*, en la cual participan profesores, alumnos y gente de la comunidad de software libre. Sirve como base para nuevos proyectos o la mejora de los actuales. Recientemente ha formado parte de el *Google Summer of Code 2015* en el que personas de todo el mundo, ofrecen su colaboración para mejorar un proyecto de software libre.

Entre estos proyectos se encuentra el Trabajo de Fin de Grado(TFG) *Navegación visual en robots aéreos* de Alberto Martín. Sus aportaciones a *JdeRobot* fueron la de un componente llamado *ardrone_server*, que crea una interfaz capaz de comunicarse con el *AR.Drone* de la compañía *Parrot*. En el mismo trabajo, incluye una herramienta, llamada *uav_viewer* cuya función es obtener la información de los sensores y controlar los actuadores de dicho *UAV*. Por último, se encuentra un componente de visión y navegación llamado *object_tracking*. Utiliza filtros de colores para el seguimiento de objetos a través de las imágenes recibidas por la cámara frontal y ventral del *drone*. El drone es capaz de un seguimiento autónomo de objetos, tanto en el suelo como en 3D.



(a) ArDrone sin y con protección de interiores



(b) Interfaz gráfica del componente *uav_viewer*

Figura 1.12. Simulación en exteriores

Daniel Yagüe en su Proyecto Fin de Carrera *Cuadricóptero AR.Drone en Gazebo y JdeRobot*, desarrolló un modelo para la plataforma *JdeRobot* en el simulador *Gazebo* del mismo *AR.drone* que utilizó Alberto Martín, anteriormente mencionado. Esto permite tanto la simulación de los datos sensoriales como de la virtualización realista de un comportamiento cercano a dicho *drone*. Adicionalmente, programó diferentes aplicaciones de navegación autónomas como el seguimiento de balizas por posición, seguimiento de carretera o el seguimiento de otro cuadricóptero. Para ello, crea unas interfaces incluidas en el entorno de *JdeRobot* que permiten el desarrollo de otras aplicaciones de navegación.

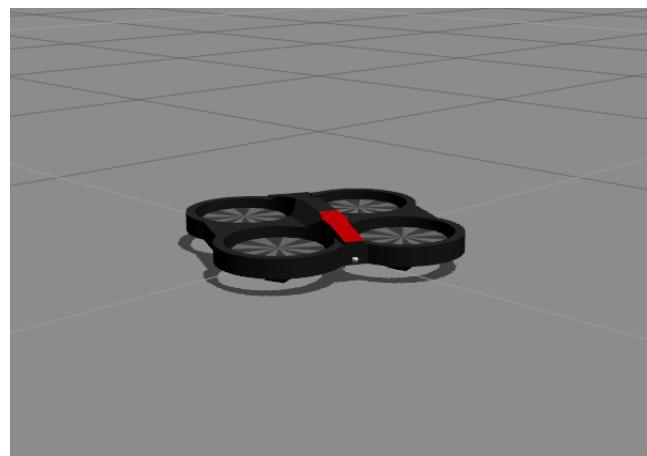


Figura 1.13. ArDrone simulado en Gazebo

Siguiendo con las bases aportadas por los dos proyectos previamente explicados, en este TFG se programará el comportamiento autónomo de aterrizaje de un drone sobre una baliza visual, móvil o estática. Se utilizará un drone simulado en Gazebo, empleando la infraestructura existente en JdeRobot para estos cuadricópteros. Con ello se pretende dar un salto adelante añadiendo un nuevo comportamiento a los ya disponibles.

En el próximo capítulo, se explicarán los objetivos y la metodología propuesta para resolverlos. En el tercer capítulo se expondrán con profundidad la infraestructura y herramientas utilizadas. En el cuarto, se describirá el desarrollo de todos los componentes que forman este proyecto. En quinto lugar, se realizarán distintos experimentos para observar los datos obtenidos en diferentes escenarios y validar experimentalmente la solución programada. Para terminar, unas conclusiones aportarán un visión global del conjunto y los conocimientos extraídos.

CAPÍTULO 2

OBJETIVOS

2.1. Tareas que cumplir

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación, que a través de la aplicación de una técnica de visión artificial, dote de un comportamiento autónomo a un *drone* en el que busque una baliza visual y aterrice sobre ella. Para llegar a él, ha sido fundamental dividir el proyecto en objetivos más pequeños, que sumados, dan forma a todo el conjunto.

- Diseño de un coche y desarrollo de un plugin, ambos para Gazebo, con una baliza visual en su techo.
- Diseño y desarrollo una aplicación de control visual para la localización, seguimiento y aterrizaje en un objeto preparado para la tarea. Se compone de una parte perceptiva, que recoge la información de las imágenes, y una parte de control y actuación, que generará los diferentes comportamientos del cuadricóptero.
- Validación experimental en el cuadricóptero simulado. Se realizarán varias pruebas para demostrar el correcto funcionamiento de la infraestructura y del control visual.

2.2. Requisitos

Se deberán satisfacer, adicionalmente, los siguientes requisitos:

- Los componentes y aplicaciones desarrollados han de estar integrados en la plataforma *JdeRobot*.
- El control del cuadricóptero simulado ha de ser fluido, de modo que pueda seguir objetos en movimiento a 5 cm/seg.
- Los componentes y aplicaciones desarrollados han de ser computacionalmente eficientes.

2.3. Metodología

Para poder materializar los objetivos y requisitos, previamente mencionados, es necesario aplicar algún método que defina las distintas etapas y estados. En este Trabajo de Fin

de Grado se aplica el método de desarrollo en espiral. Este modelo, creado por Barry Boehm en 1986, se utiliza frecuentemente en la ingeniería de software. Se basa en una serie de iteraciones en bucle. En cada ciclo, se realiza un conjunto de cuatro actividades:

1. **Determinar los objetivos:** Poner limitaciones definidas en forma de objetivos o requisitos. Dividir el proyecto en partes más pequeñas.
2. **Análisis del riesgo:** Estudiar los riesgos de cada uno de los objetivos que se abordan. Evaluar las alternativas posibles en caso de amenazas.
3. **Desarrollar y probar:** Verificación de la tarea actual. Al mismo tiempo, se realiza un análisis para encontrar nuevos factores de riesgo, como errores que se podrían arrastran a la próxima iteración.
4. **Planificación:** Establecer y definir las fases anteriores.



Figura 2.1. Representación del desarrollo en espiral.

2.4. Planificación

Durante el desarrollo de este proyecto, se han establecido reuniones periódicas con el tutor. En ellas, revisábamos los objetivos anteriormente fijados y los resultados obtenidos. Si alguno de los objetivos generaba algún problema o no se llegaba al resultado deseado, se aplazaban o se profundizaba en la raíz del problema. A continuación, se determinaban los subobjetivos de nuestro próximo encuentro.

Como parte de la evaluación de los objetivos propuestos, ha sido fundamental la utilización del *mediawiki*¹ de la plataforma *JdeRobot*. En el publicaba, a modo de cuaderno de bitácora, redactando los éxitos y progresos, haciendo uso además de contenido multimedia como imágenes o videos.

Para el seguimiento y almacenamiento del software desarrollado hemos empleado la herramienta Subversion. Todo el código relacionado con este proyecto se encuentra alojado en mi repositorio².

¹<http://jderobot.org/Andresjhe-tfg>

²<https://svn.jderobot.org/users/andresjhe/tfg>

- Familiarización con el entorno de *JdeRobot*: Incluye el estudio de las dependencias necesarias para la instalación del entorno. Estudio de las diferentes bibliotecas, interfaces y componentes. Aprendizaje y profundización de lenguajes de programación como Python y C++, así como la herramienta para las comunicaciones ICE. Así como el estudio y familiarización de la biblioteca April Tags para la detección de los mismos y la biblioteca de visión OpenCV.
- Familiarización con Gazebo. Se ha estudiado los plugins ya existentes en JdeRobot y a su vez, la creación otros nuevos a través de la API de Gazebo. El aprendizaje de Blender ha sido necesario para generar el modelo virtual de un vehículo. Por último, se ha desarrollado su respectivo plugin en Python para teledirigir y asignar rutas al vehículo anteriormente generado. Se ha desarrollado un componente integrado en JdeRobot que calcule y dibuje distancias a partir de la posición de modelos en el simulador Gazebo.
- Estudio del dispositivo MK802IV para una modificación del software para ser empleado como unidad de procesamiento junto al AR.Drone. Adicionalmente, se ha investigado diferentes medios para realizar la comunicación con el drone.
- Desarrollo de la aplicación que se encarga del control visual. En este punto se han creado diferentes mundos en Gazebo y se han diseñado distintos comportamientos en el vehículo.
- Validación experimental, en la que se validará en función de los resultados la solución programada.

CAPÍTULO 3

INFRAESTRUCTURA

En cada uno de los siguientes apartados se detallará la función de los programas o dispositivos necesarios para la elaboración de este proyecto. Debido a la naturaleza de la plataforma JdeRobot, el sistema operativo que se ha elegido para el desarrollo y ejecución de los componentes ha sido Linux. En concreto, la distribución Ubuntu 14.04.

3.1. Interfaz Gráfica de usuario

También conocida como *GUI*, es el software encargado de simplificar la interacción con los usuarios. Proporciona texto, imágenes y diferentes objetos gráficos para representar la información y e interactuar con las acciones que se ofrezcan en un programa.

3.1.1. Qt

Qt¹ es una biblioteca multiplataforma cuyo objetivo principal es el de facilitar el diseño de interfaces gráficas de usuario. Su modelo de desarrollo es el de software libre y de código abierto, bajo la licencia GPL v3 y LGPL v2.1. Está programada en C++, aunque puede ser utilizada junto a otros lenguajes de programación. Ofrece un *ambiente de desarrollo integrado*(IDE en inglés) llamado QtCreator, que facilita el proceso de desarrollo y el diseño gráfico del GUI. La versión utilizada en este proyecto es la 4.8.

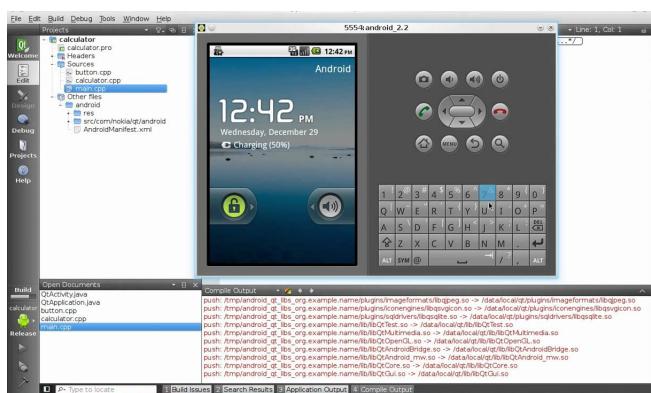


Figura 3.1. Qt Creator

¹<http://www.qt.io/>

PyQt

PyQt² es una colección de bibliotecas escritas en Python y C++ que complementan Qt. Mantiene la característica multiplataforma y es un partner reconocido oficialmente por el propio Qt. Se encuentra bajo la licencia GPL v3 y la *Riverbank Commercial License*. En este proyecto se ha utilizado para el desarrollo de componentes utilizando el lenguaje de programación Python. La versión utilizada en este proyecto es la PyQt5.

3.1.2. Matplotlib

Matplotlib³ es una librería en python para dibujar gráficas en dos dimensiones. Ofrece una interfaz muy cercana a la ofrecida por el programa *Matlab* y su utilización resulta muy familiar para usuarios que dominan este programa. Su principal ventaja es la simplificación a la hora de dibujar diferentes estilos de líneas, cambiar las propiedades de ejes, las propiedades del texto, etcétera.

3.2. ICE

ICE⁴ es un middleware orientado a objetos proporciona las herramientas, APIs y librerías necesarias para simplificar las comunicaciones entre cliente y servidor. En JdeRobot es la plataforma elegida como infraestructura para todos los procesos de comunicación entre componentes y plugins. Proporciona una capa transparente que se encarga de abrir y cerrar conexiones, la serialización de información, retransmisión de paquetes perdidos, etcétera. La versión utilizada en este proyecto es la 3.5

3.3. JdeRobot

En el mundo de la robótica, existen diferentes plataformas que simplifican y aportan las herramientas necesarios para el desarrollo de aplicaciones en robots. JdeRobot es una de ellas y consiste en una colección de aplicaciones robóticas, domóticas y de visión artificial. Estas aplicaciones están escritas en diferentes lenguajes como C++, Java o Python y su interoperación se realiza a través de interfaces ICE. En ella participan desarrolladores de diferentes niveles desde profesionales del sector, profesores, alumnos de la Universidad Rey Juan Carlos y de otras universidades de todo el mundo. Durante el ciclo de vida este proyecto ha servido como plataforma de exposición de los progresos y logros conseguidos. El código fuente es libre y está bajo la licencia GPLv3 y la documentación se encuentra protegido bajo la licencia de Creative Commons by-SA.

Durante el desarrollo de este TFG, componentes como uav_viewer, introrob_py han servido de referencia y han tenido una gran relevancia para el aprendizaje de la plataforma. La versión utilizada en este proyecto es la 5.2.4⁵

3.3.1. Teleoperadores uav_viewer e introrob_py

Ambos son componentes de JdeRobot y su función es la de ofrecer una GUI para enviar comandos tanto al cuadricóptero real, como al Ardrone creado por Daniel Yagüe. Los

²<https://riverbankcomputing.com/software/pyqt/intro>

³<http://matplotlib.org/>

⁴<https://zeroc.com/>

⁵<https://github.com/RoboticsURJC/JdeRobot>

comandos se envían a través de interfaces ICE. Estos permiten el desarrollo de programas independientemente del tipo de cuadricóptero (real o simulado) y transportan la información de los sensores. La principal diferencia entre ambos es el lenguaje en el que han sido desarrollados: introrob_py está escrito en Python mientras que uav_viewer está en C++.

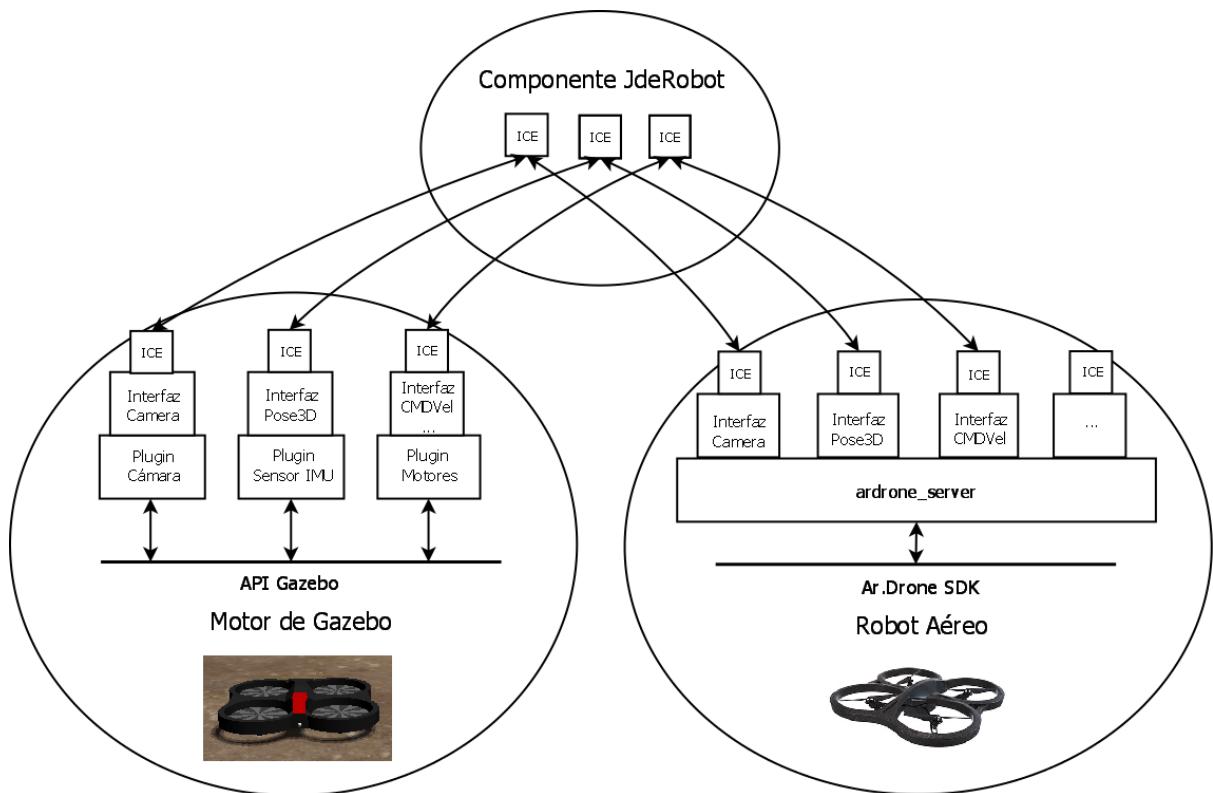


Figura 3.2. Interfaz de JdeRobot para el AR.Drone

3.4. Gazebo

La herramienta en la que se realizan todas las pruebas es el entorno de simulación Gazebo. Ya fue introducido en el apartado 1.3, por tanto, hablaremos de su estructura y las bibliotecas de las que se sirve este proyecto.

Gazebo se divide en dos programas:

1. **Gzclient:** Es la GUI a través la cual visualizar los resultados y comportamientos de nuestros modelos virtuales. Ofrece diferentes herramientas como la pausa de la simulación, toma de instantáneas, añadir objetos sobre la marcha, etcétera.
2. **Gzserver:** Este programa se encarga del control de la simulación, la generación del mundo virtual y el motor físico.

La estructura que define una simulación en Gazebo se compone de varios elementos:

- **Mundos:** Describen todos los elementos que contiene una simulación, incluyendo robots, luces, sensores y objetos estáticos. Están escritos en formato SDF (explicado más adelante) y usan la extensión *.world*.

- **Modelos:** Se generan a partir de archivos en formato SDF. Contienen las propiedades y los plugins necesarios para la simulación. Gazebo incluye una base datos de robots que se descarga a través de Internet. Estos modelos pueden ser insertados y facilitan la familiarización con el entorno.
- **Variables de entorno:** Se utiliza para localizar los diferentes archivos como mundos, modelos o plugins.
- **Plugins:** Son un mecanismo para interacción con Gazebo a través de su API⁶. Permiten la modificación de parámetros durante la simulación, con el fin de cambiar el comportamiento de los modelos.

Formato SDF

Algunos de los elementos anteriormente mencionados utilizan el formato SDF, el cual se basa en lenguaje XML para describir las propiedades de los mundos y modelos en Gazebo.

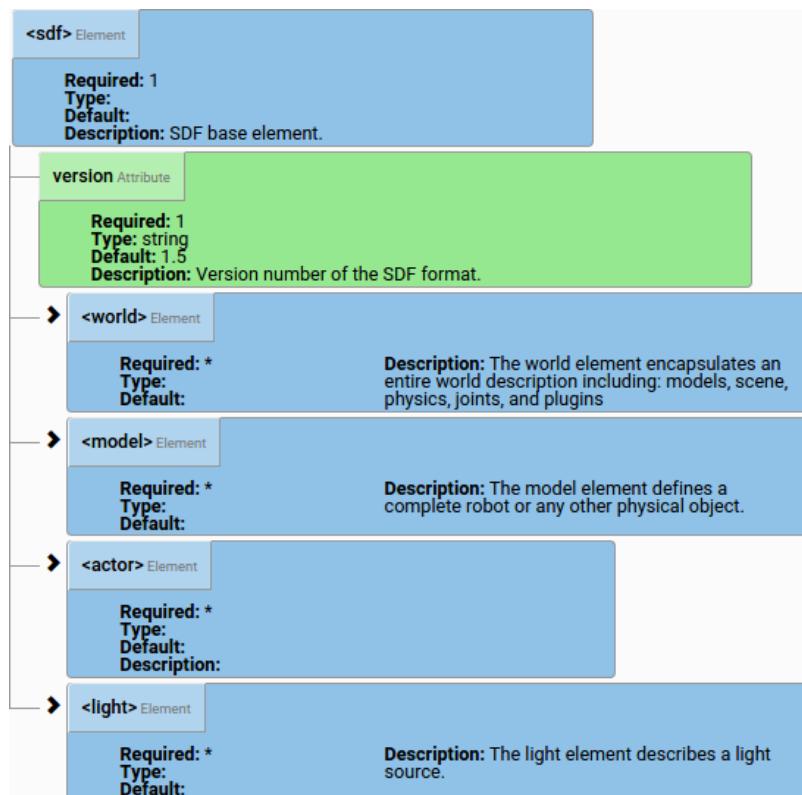


Figura 3.3. Ejemplo del formato SDF

3.4.1. Plugins

Los plugins permiten acceder a todas las funcionalidades de Gazebo a través de clases en C++ descritas en la su respectiva API⁷.

⁶<http://gazebosim.org/api.html>

⁷<http://gazebosim.org/api.html>

3.5. Blender

3.6. Visualización virtual

3.6.1. AprilTags

En este proyecto, se utilizará una librería basada en visualización virtual denominada *AprilTags*. Es un sistema de visualización fiduciaria. Estos sistemas se basan en símbolos diseñados para ser fácilmente reconocidos del resto del entorno. Puede detectar uno o varios símbolos en la misma imagen. Además de proporcionar información como la identificación y posición del símbolo dentro de una imagen. Es un sistema robusto cuyo funcionamiento es independiente del ángulo y diferentes situaciones de luminosidad en la imagen.

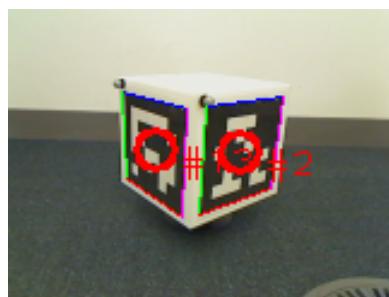


Figura 3.4. Ejemplo de detección en AprilTag

Sus aplicaciones son muy variadas, desde la captura de movimiento en objetos hasta sistemas de balizas en técnicas de navegación. Otra de sus aplicaciones es la de *realidad aumentada*, sustituyendo el símbolo por una imagen virtualizada.

Originalmente está escrita en C y Java pero Ed Olson de Massachusetts Institute of Technology(MIT) ha creado una adaptación en C++. El código⁸ es abierto y está protegido bajo la LGPLv2.1. Entre sus dependencias se encuentran OpenCV y Eigen3 que son conocidas bibliotecas de tratamiento de imagen en Linux.

Los símbolos se dividen en diferentes familias. Estas familias utilizan un número de bits y de distancia de Hamming diferentes.

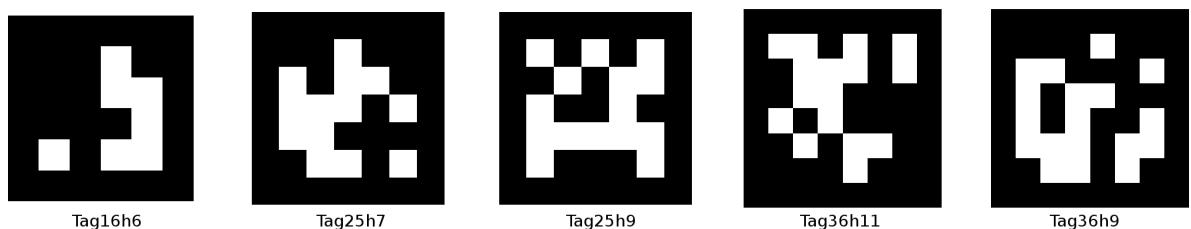


Figura 3.5. Ejemplos de familias en AprilTag

⁸<https://svn.csail.mit.edu/apriltags>

3.6.2. OpenCV

3.7. Dispositivos físicos

3.7.1. MK802IV

3.7.2. Ar.Drone 2

CAPÍTULO 4

DESARROLLO DE LA INFRAESTRUCTURA

4.1. Coche virtual y teleoperador

Diana

4.2. Mk802IV

4.3. Distance Viewer

CAPÍTULO 5

ATERRIZAJE VISUAL

5.1. Control visual

CAPÍTULO 6

CONCLUSIONES Y LÍNEAS FUTURAS

Incluyendo discusión de los logros principales alcanzados y posibles trabajos futuros.

6.1. Conclusiones

Bla, bla, bla,...

6.1.1. Título de la primera subsección

Bla, bla, bla,...

6.2. Líneas futuras

Bla, bla, bla,...

Bibliografía

- [1] J. Rice, *Mathematical Statistics and Data Analysis*, 2nd ed. Duxbury Press, 1995.
- [2] L. Pérez Lombard, J. Ortiz, and C. Pout, “A review on buildings energy consumption information,” *Energy and Buildings*, vol. 40, no. 3, pp. 394–398, Mar. 2008.
- [3] W. Huber, M. Lädke, and R. Ogger, “Extended floating-car data for the acquisition of traffic information,” in *World Congress on Intelligent Transport Systems*, Nov. 1999.
- [4] (2011) The MATLAB website. [Online]. Available: <http://www.mathworks.com/>
- [5] *Código Técnico de la Edificación*, Ministerio de Vivienda, Apr. 2009.

