Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FINAL DE GRADO

TÍTULO DEL TFG: Aplicación dron en la lucha contra la caza furtiva de rinocerontes en África.

TITULACIÓN: Grado en ingeniería de Aeronavegación

AUTOR: Sandra Bordera Antequera

DIRECTOR: Esther Salamí San Juan

SUPERVISOR: Arnau García Terrades

FECHA: julio de 2017

Resumen

En la última década el número de rinocerontes africanos en libertad se ha visto reducido en un 28%. El factor más importante que está acabando con la especie es el aumento de la caza furtiva, que los persigue con la finalidad de obtener su cuerno y posteriormente venderlo en el mercado negro.

En la actualidad la mayor población de rinocerontes se encuentra en los parques protegidos de Sudáfrica, la mayoría de los cuales cuentan con grandes extensiones de terreno. Este proyecto está dirigido a desarrollar una herramienta dron de bajo coste capaz de proporcionar a los guardabosques información útil del terreno.

El objetivo ha sido crear una aplicación dron capaz de identificar puntos calientes en la zona sobrevolada y obtener la máxima información posible acerca de estos y de la zona que los rodea. Para cumplir con el objetivo, hemos utilizado tecnologías relacionadas directamente con la visión artificial y con el control dron a través de comandos. Gracias a ellos hemos creado una aplicación capaz de modificar el plan de vuelo del dron de forma totalmente autónoma.

En este documento se puede encontrar la definición detallada de la aplicación desarrollada, así como los procedimientos técnicos que hemos seguido. En último lugar se encuentra la validación tanto teórica como experimental de la aplicación creada.

Overview

Last decade the number of rhinoceroses living in freedom has been reduced by a 28%. The most important factor that is extinguishing the species is the increase of poaching, which pursues the rhinos in order to obtain their horn and later sell it on the black market.

Nowadays, the largest population of rhinoceroses can be found in South Africa's protected parks, most of them have large tracts of land. This project is aimed at developing a low-cost drone tool capable of providing the rangers with useful terrain information.

The objective was to create a drone application capable of identifying hot spots in the overflown area and get as much information as possible about them and the area around them. To achieve the objective, we have used technologies directly related to artificial vision and drone control through commands. Thanks to them we have created an application capable of modifying the flight plan of the drone completely autonomously.

In this document you can find the detailed definition of the application developed, as well as the technical procedures we have followed. At the end you can find the theoretical and experimental validation of the application created.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

1.1. Motivación

Durante la última década, la población global de rinocerontes se ha visto gravemente afectada debido al importante aumento de la caza furtiva en los parques naturales africanos. Entorno a esta actividad se encuentran numerosas organizaciones criminales organizadas, las cuales intentan a toda costa hacerse con el cuerno de los rinocerontes adultos de los que obtendrán grandes sumas de dinero en los mercados ilegales.

El tráfico de cuernos de rinoceronte se focaliza mayoritariamente en los mercados negros asiáticos, donde el precio de un único ejemplar puede llegar a alcanzar hasta los 53.000€[1]. El producto es altamente codiciado ya que en la medicina tradicional asiática se le atribuyen propiedades curativas y afrodisiacas además de ser uno de los ingredientes claves en las medicinas tradicionales chinas y vietnamitas [2].

Sudáfrica es el país más afectado por la caza furtiva ya que en sus parques viven alrededor de 17.000 ejemplares que representa el 80% de la población total africana. Durante el pasado año 2016 fueron asesinados 1.054 rinocerontes, no incluidos los que fueron mutilados, siendo este el cuarto año consecutivo en que la cifra supera los 1.000 ejemplares [2]. La siguiente figura 1 muestra los rinocerontes cazados por año desde 2006.

Figura 1. 1:Número de rinocerontes cazados ilegalmente en Sudáfrica. Fuente WWF.

Desde SANParks (*South African National Parks*) [3] se llevan a cabo numerosas operaciones contra la caza ilegal. Según el informe anual publicado por la asociación, el pasado año se intensificaron las labores y actividades de lucha contra la caza furtiva. El parque natural Kruger, el más importante del país, vio cómo se redujo el número de víctimas respecto el año anterior. Este hecho no se consideró como una victoria ya que la actividad se desplazó a otros parques con menos vigilancia[3]. De media, Sudáfrica pierde 3 rinocerontes al día debido a la caza ilegal [4].

El principal problema a los que los parques se enfrentan es el bajo presupuesto que les otorgan para afrontar la lucha contra la caza furtiva. A pesar de ello se conoce que la asociación dispone de una avioneta y 4 helicópteros operativos en las actividades de vigilancia de la fauna en peligro de caza ilegal.

El objetivo de nuestro proyecto será desarrollar una aplicación dron de bajo coste, que pueda proporcionar a lo guardabosques información útil acerca del estado del parque y de los animales que viven en él. Esta aplicación formará parte de un proyecto más amplio conocido como *Ranger Drone* dedicado a la implementación de los drones como herramientas útiles contra la caza furtiva. Este proyecto comenzó a desarrollarse en el año 2014, con la colaboración de estudiantes de la UPC y la empresa *start up* Hemav, cuya sede se encuentra en el Campus Del Baix Llobregat.

En el siguiente apartado explicaremos el estado del arte del uso de los drones tanto en la lucha contra la caza furtiva como en otros campos, en los cuales son considerados herramientas de gran interés.

Figura 1. 2:Rinocerontes del parque natural Kruger. Fuente web de KNP [5].

1.2. Estado del arte

1.2.1. Historia de los drones

Desde la aparición del primer dron durante la Primera Guerra Mundial su evolución no ha dejado de mejorar llegando hasta lo que hoy en día conocemos como los drones modernos. En un primer momento fueron desarrollados con fines bélicos siendo utilizados como bombas tripuladas o herramientas para afinar la artillería antiaérea [6]. Hoy en día su uso se extiende a una infinidad de tareas en un gran número de campos como la entrega de paquetería, labores de vigilancia, rodajes cinematográficos etc.

A continuación, mostramos el dron Radioplane-OQ-2, uno de los primeros en ser comercializado y vendido de forma masiva durante la Segunda Guerra Mundial.

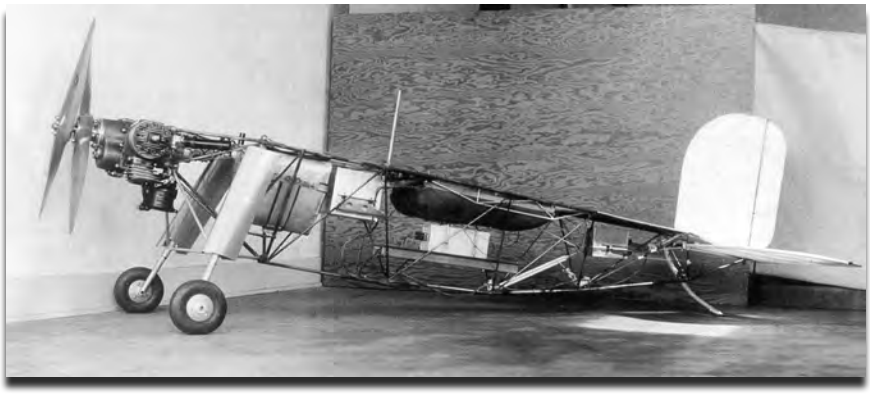

Figura 1. 3:OQ-2, Righter 2-GS-17 Photo: Righter Family Archives [7]

1.2.2. Usos de los drones en la actualidad

Actualmente los drones son usados en una gran variedad de aplicaciones. Su manejabilidad y capacidad de alcanzar lugares de difícil acceso en tiempos reducidos los convierten en herramientas útiles en labores de salvamento y vigilancia.

A continuación, hablaremos de algunas aplicaciones dron que se están implementado en la actualidad.

Drones en la lucha contra incendios medioambientales

Drones como el MQ-1 Predator son utilizados en las labores de extinción de fuegos forestales. En el año 2013 su uso fue decisivo en la lucha contra los incendios del parque natural Yosemite, California.

Fueron utilizados para recopilar información acerca del incendio en las zonas de difícil acceso, localizar las rutas por las que los bomberos podían acceder al fuego de forma segura, localizar los focos más activos y determinar las zonas donde el fuego podía considerarse controlado [8].

Drones aplicados en labores de salvamento acuático

En las labores de rescate en el océano el factor más apreciado es el tiempo y la rapidez con la que se actúa. Los drones son utilizados como ayuda en labores de búsqueda de víctimas y envío de salvavidas, que aumentan las posibilidades de supervivencia de las víctimas.

La empresa General Drones ha desarrollado un dron llamado Auxdron, el cual detecta víctimas en alta mar y facilita su localización a los guardacostas para realizar el rescate. Este dron transporta dos salvavidas a las víctimas, los cuales se hinchan con el contacto del agua. A continuación, mostramos una fotografía del dron de rescate marítimo Auxdron [9].

Figura 1. 4:Dron de salvamento marítimo desarrollado por la empresa General Drones

Drones aplicados en labores de rescate en montaña

En la actualidad encontramos el proyecto SHERPA, de ámbito internacional, cuyo objetivo es el desarrollo de plataformas tanto aéreas como terrestres que proporcionen ayuda a las labores de búsqueda y rescate en zonas montañosas de difícil acceso.

Por otra parte, asociaciones públicas como la agencia gallega de emergencias (Axenga) ha introducido en sus simulacros de rescate drones equipados con cámaras térmicas capaces de detectar personas extraviadas. Estos drones

permiten la carga de 8 Kg de peso de forma que pueden cargar con un pack de primeros auxilios que es lanzado cerca del lugar donde se ha detectado a la víctima[10].

Drones equipados con herramientas de primeros auxilios.

Otra aplicación muy interesante que está siendo estudiada actualmente es la conocida como dron-ambulancia. Al año alrededor de 80.000 personas sufren un paro cardiaco repentino, de los cuales únicamente el 8% sobreviven. Los especialistas alegan que es crucial actuar con rapidez en estas situaciones y en la mayoría de casos las ambulancias con el equipo de reanimación llegan demasiado tarde.

Por esta razón se ha desarrollado un prototipo de dron equipado con un desfibrilador capaz de llegar al lugar donde se ha enviado la alarma en menos de un minuto si este se encuentra en un radio de 12 Km de la posición inicial del dron. El dron también cuenta con sistema de comunicación a tiempo real que permite la comunicación entre un operario especializado y el acompañante de la víctima de forma que el acompañante pueda recibir instrucciones acerca del funcionamiento del desfibrilador [11].

Esta aplicación, a pesar de tener gran potencial, no se puede poner en uso debido a la actual legislación vigente que prohíbe el uso de los drones sobre zonas urbanas.

1.2.3. Drones contra la caza furtiva en África

Durante los últimos años se han desarrollado diferentes estudios acerca de la viabilidad y utilidad de implementar drones como herramientas contra la caza furtiva en los parques africanos.

Un equipo de investigadores del CSIC (Consejo Superior de Investigaciones Científicas) desarrolló en 2013 un dron de ala fija y de largo alcance capaz de recolectar información sobre el estado del parque. El dron fue diseñado con la finalidad de que fuese capaz de detectar cazadores furtivos, para ello fue equipado con dos cámaras, una térmica, que utilizaban en la noche, y una de alta resolución, que usaban durante el día. Este grupo de investigadores consiguió llevar su proyecto a Sudáfrica y probar su funcionamiento en el KNP (Kruger National Park). Los resultados obtenidos revelaron que a pesar haber conseguido información útil proporcionada por el dron era necesario un desarrollo más exhaustivo de las tecnologías aplicadas [12].

Uno de los problemas que presentaron fue que, durante el vuelo del dron, la cámara térmica no consiguió buenos resultados debido a la distorsión inducida conforme aumenta la altura de vuelo.

Por otra parte, la empresa UDS (UAV & *Drone Solutions*) también tuvo la oportunidad de probar su prototipo, en este caso en la reserva *Hluhluwe-iMfolozi,* la más antigua de Sudáfrica. La empresa operaba con drones de ala fija equipados con tecnología que elevaba su precio hasta los 12.000€ por dron [13].

Por desgracia, tras el periodo de prueba al que se sometieron los diferentes prototipos diseñados por los equipos de investigación y empresas de drones, los resultados no fueron los esperados y desde SANParks se decidió cancelar, por el momento, el programa de inserción de drones.

Tras el año de evaluación se rebeló que la tecnología dron no había podido hacer frente a las características del parque, el cual mantiene una alta temperatura durante todo el día lo que dificulta la identificación de animales con la cámara térmica ya que no son capaces de diferenciar una roca, que desprende el calor absorbido por el día, de un animal o un cazador.

Por otra parte, tampoco se consiguió diferenciar la presencia de los animales del parque con la presencia de los cazadores furtivos. [1]

1.2.4. Legislación UAS en Sudáfrica

Recientemente Sudáfrica ha desarrollado una serie de regulaciones aplicadas al uso de los drones. Estas regulaciones difieren según el carácter de la aplicación del dron, por una parte, nos encontramos la normativa que regula los drones de carácter privado, por otra parte, la normativa que regula los drones usados en carácter público. En este caso el dron debe ser registrado y debe ser operado en términos de la SACAR (*Souht African Civil Aviation Regulation*)

Algunas de las regulaciones aplicadas al uso de drones con ámbito privado son las siguientes [14]:

- No ser usado con fines económicos.
- No volar el dron más de 50m de LOS (*Line of Sight*)
- No volar sobre los 150m de altura respecto al suelo si no se ha obtenido un permiso previo de la SACAA (*Souht African Civil Aviation Authority)*
- No volar sobre zonas urbanas o aglomeraciones de personas.
- No volar un dron cuyo peso supere los 7Kg.
- No volar en zonas de espacio aéreo restringido ni prohibido.
- No volar en zonas de espacio aéreo controlado.

1.3. Propuesta de proyecto

Los guardabosques presentan un gran número de dificultades en sus labores de vigilancia. Entre otras cosas se encuentra la escasez de información acerca del estado del parque, no conocen el paradero de los animales ni su estado. En ocasiones se utilizan helicópteros y avionetas con el objetivo de recopilar este tipo de información.

Por desgracia el uso de estas herramientas puede llegar a ser muy costoso por lo que su implementación no es tan constante como debería. Por estos motivos hemos querido desarrollar una aplicación dron, de reducido coste económico, que pueda resultar de utilidad a los guardabosques en este tipo de labores.

Figura 1. 5:Helicóptero propiedad de SANParks [15].

El objetivo de nuestro proyecto es desarrollar una aplicación dron capaz de proporcionar información útil acerca del estado de la zona que está sobrevolando. Principalmente, el objetivo será detectar rinocerontes y conseguir la máxima información posible acerca del estado del animal y de la zona que lo rodea.

El dron que usemos necesitará estar equipado con una cámara térmica y un procesador, en el que instalaremos el código de la aplicación. La aplicación será capaz de procesar las imágenes térmicas, en tiempo real, y determinar si existen o no puntos calientes que puedan ser rinocerontes. Este procesador deberá a su vez disponer de un *router* con acceso a internet, el motivo será comentado en la sección siguiente.

Como ya hemos comentado, la tecnología de las cámaras térmicas actuales no proporciona una resolución nítida de la imagen cuando son tomadas a gran altura. La resolución de la imagen disminuye conforme crece la altura de vuelo, por otra parte, el área cubierta por imagen aumenta con la altura.

El objetivo de la misión es recolectar la mayor información posible, por lo que la altura de vuelo del dron deberá ser elevada de forma que las imágenes cubran la mayor área posible. Con la finalidad de obtener la máxima información acerca de los puntos calientes detectados, la aplicación creará una misión de observación entorno de los puntos calientes detectados. Esta misión de observación se realizará a una menor altura, la cual permita obtener una imagen más definida de los puntos calientes.

Con esto conseguiremos identificar si el punto caliente detectado se trata de un rinoceronte o cualquier otro animal, un humano o una falsa alarma. También conseguiremos información acerca de la zona que rodea el punto caliente.

Esta aplicación será capaz de modificar el plan de vuelo del dron de forma completamente autónoma lo cual lo convierte en un dron autónomo, es decir, su control dependerá de los resultados de las imágenes procesadas y no de las decisiones de un piloto.

Para poder llevar a cabo el desarrollo de nuestra aplicación necesitaremos implementar nuevas tecnologías como la visión artificial y el control autónomo de drones.

1.4. Descripción de la misión

La aplicación se desarrollará con la finalidad de realizar una misión de vigilancia concreta. Para una mayor eficiencia de la misión, el dron debería caracterizarse por tener un gran alcance, es decir, que sea capaz de recorrer grandes distancias.

La misión consistirá en dirigir al dron hacia un punto concreto, escogido por el guardabosques. Durante el trayecto de ida hacia este punto, el dron será capaz de analizar el estado de la zona sobrevolada en busca de rinocerontes.

Cabe destacar que la aplicación ha sido diseñada con el objetivo de ser lo más adaptable posible a los requisitos de los guardabosques. También se pretende que se adapte a las características de la cámara térmica con la que se equipe el dron.

Es por esto que la aplicación solicita diferentes datos al usuario antes de comenzar con la misión. La siguiente figura muestra los datos que deben ser introducidos en la misión.

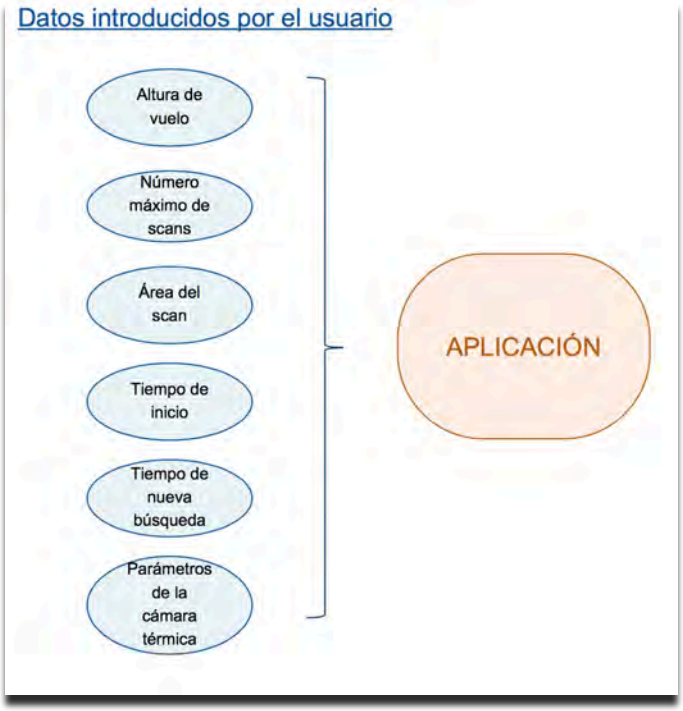

Figura 1. 6: Datos de la aplicación introducidos por el usuario.

A continuación, definiremos la misión de una forma más detallada.

La misión comenzará con el despegue del dron a la altura de vuelo introducida previamente por el usuario. Una vez el dron haya alcanzado la altura deseada, la aplicación lanzará un servidor web (gracias al *router* del procesador), en el cual aparecerá un mapa señalizado con el Home del dron (lugar de despegue). En un futuro, se pretende personalizar este mapa, añadiendo una circunferencia con centro en Home y radio igual al alcance máximo del dron, el cual dependerá de las características del dron que se use en la misión (duración de las baterías y las cualidades técnicas de la plataforma).

El usuario podrá acceder a este servidor desde cualquier dispositivo. Gracias al mapa mostrado, el guardabosques podrá seleccionar el punto al que el dron debe dirigirse, es decir, el punto final de la misión que a partir de ahora llamaremos Punto de Misión, el cual deberá encontrarse dentro del radio de alcance máximo señalizado. Las coordenadas de este punto serán enviadas a la aplicación vía internet.

La aplicación recibirá las coordenadas del Punto de Misión, seguidamente ordenará al dron que se dirija hacia esta localización. El dron seguirá las instrucciones marcadas por la aplicación y comenzará su desplazamiento hacia el punto.

Una vez comience el desplazamiento, la aplicación esperará un tiempo pertinente (tiempo de inicio introducido por el usuario) para comenzar con la

labor de búsqueda de puntos calientes. Esta opción se ha introducido por motivos de eficiencia. Ya que el dron se dirige de forma autónoma, queremos que la modificación de la ruta se deba al hallazgo de un punto caliente alejado del lugar donde ha despegado, que podría estar demasiado cerca de zonas de vigilancia frecuentadas por gente. Usamos la variable tiempo en lugar de distancia debido a la facilidad de implementación en el código, ya que únicamente es necesario ordenar a la aplicación que se mantenga en *stand by* durante ese tiempo y no necesita realizar cálculos que puedan entorpecer su funcionamiento.

Transcurrido este tiempo de inicio, la aplicación comenzará a procesar las imágenes térmicas capturadas, a su vez cambiará el *ground spee*d del dron y lo establecerá a 1m/s. Gracias a mantener esta velocidad, no muy grande, el procesador tiene tiempo de procesar la imagen e introducir la misión de observación (si detecta punto caliente) antes de que el dron avance demasiado y deba retroceder para realizar la misión de observación.

Figura 1. 7:Esquema, inicio de procesado de imágenes.

Si la aplicación detecta un punto caliente en la imagen que está procesando creará una misión de observación entorno al punto caliente, al cual llamaremos Punto de Scan. Esta misión de observación consistirá en un *scan* de 8 puntos el cual analizará la zona alrededor del Punto de Scan. Esta misión de observación se realizará a una altura que permita la obtención de imágenes térmicas con resolución suficiente para identificar el objeto caliente, si es un rinoceronte, una persona o simplemente una falsa alarma.

La misión de observación será creada en función del área que quiere escanear el usuario. La aplicación creará el *scan* y lo introducirá al plan de vuelo del dron, que de forma inmediata modificará su dirección y comenzará a descender dirigiéndose a al primer punto que define la misión de observación. Cabe decir que la aplicación únicamente procesará imágenes durante el estado de búsqueda, es decir, no se procesará imagen durante la misión de observación.

Una vez acabada la misión de observación, el dron recuperará la altura de vuelo introducida por el usuario y retomará su rumbo hacia el Punto de Misión, desde el último punto del *scan*. La aplicación esperará otro margen de tiempo antes de volver a procesar las imágenes en busca de más puntos calientes. Este tiempo de espera también será introducido por el usuario. Dejando este margen de tiempo podemos separar las zonas sobre las que se ha realizado una misión de observación y así conseguir una mayor eficiencia de la misión. Ver figura 4.

El usuario deberá introducir el número máximo de misiones de observación que pueden ser realizadas en la misión. Este número dependerá del rango máximo que pueda alcanzar la plataforma que se esté usando. Cuanto mayor sea el rango máximo del dron usado mayor número de misiones de observación podrán ser realizadas por misión.

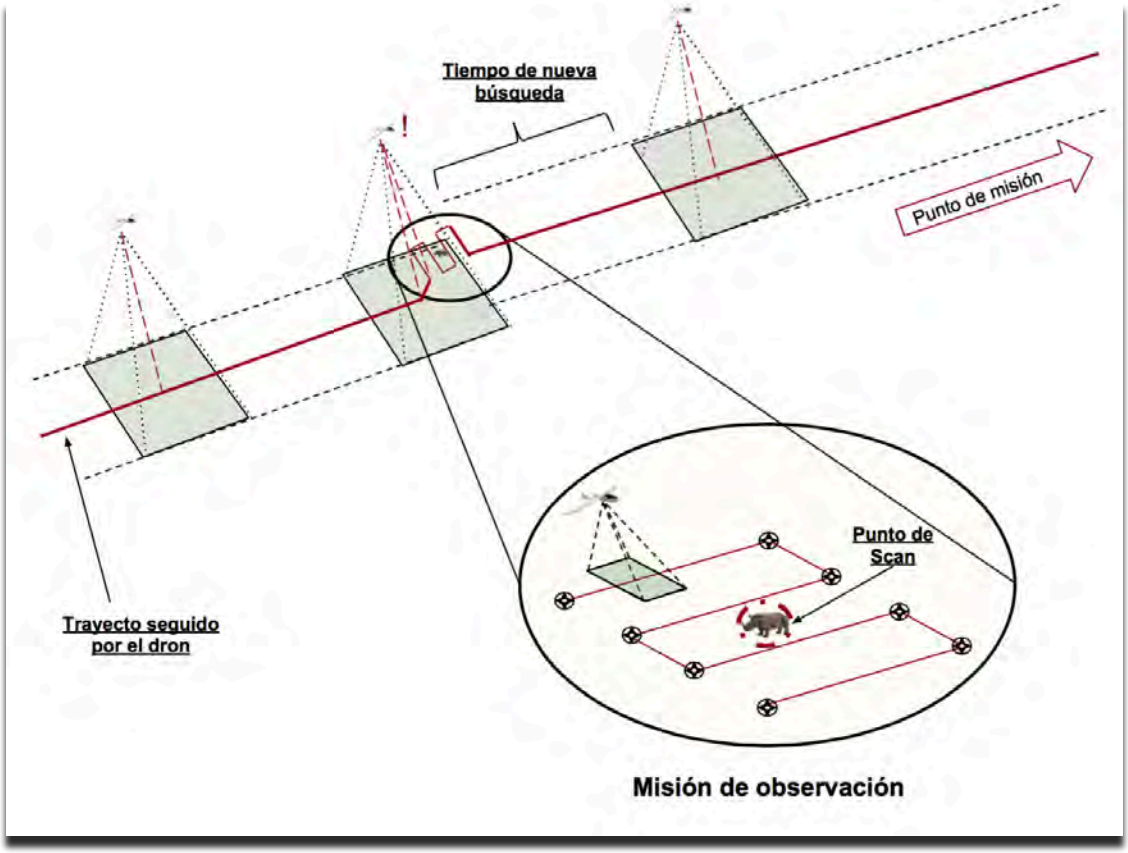

Figura 1. 8:Detección de punto caliente, realización de la misión de observación.

Una vez el dron alcance el Punto de Misión, la aplicación cambiará su modo de vuelo de “AUTO” (automático) a “RTL” (retorna a casa y aterriza) por lo que el dron se dirigirá al lugar dónde ha despegado y aterrizará.

CAPÍTULO 2: Desarrollo de la aplicación dron

El capítulo 2 está dedicado a definir de un modo más detallado las características técnicas de la aplicación que hemos desarrollado. Este capítulo a su vez está dividido en dos sub-apartados independientes.

En el primer sub-apartado hablaremos de las principales herramientas software que hemos utilizado en el desarrollo de la aplicación. El segundo apartado lo hemos dedicado a la explicación más detallada del procedimiento seguido por el código de la aplicación, tanto en el procesado de imagen como en la modificación del plan de vuelo de forma autónoma.

2.1. Herramientas software

En este sub-apartado hablaremos sobre las herramientas software más importantes que hemos usado. Introduciremos el lenguaje de programación Python y las principales herramientas que hemos necesitado para conseguir que el procesador se comunique con el piloto automático. Finalmente hablaremos de CherryPy y Mapbox, que nos han permitido utilizar un servidor web en nuestra aplicación e introducir un mapa personalizado para que el usuario pueda seleccionar el punto final de la misión (Punto de Misión) de forma manual.

2.1.1. Lenguaje de programación

El lenguaje de programación utilizado en nuestra aplicación es Python. El intérprete de Python y su gran variedad de bibliotecas se puede descargar de forma gratuita desde su página oficial. El uso de Python es gratuito incluso en programas con fines empresariales, es por esto que muchas grandes empresas como Google, Yahoo y NASA lo utilizan en sus códigos [16]

Este lenguaje apareció por primera vez en los años 80 y hoy en día se ha convertido en uno de los más conocidos y utilizados. Su popularidad se debe principalmente a la sencillez y velocidad con la que se desarrollan los programas, la gran variedad de librerías que contiene y el gran número de plataformas que le dan soporte, Python puede ser usado en las plataformas más importantes como Windows, Linux y MacOS. Python es considerado un lenguaje de programación general, ya que con él se pueden crear todo tipo de programas, desde aplicaciones hasta páginas web.

Para trabajar con Python de una forma sencilla y práctica hemos necesitado utilizar PyCharm, un IDE (*Integrated Development Environment*) especializado en el lenguaje de programación Python que nos ha proporcionado diferentes herramientas para el desarrollo de la aplicación. Una de estas herramientas

proporcionadas por PyCharm es el debugger, que nos facilita la revisión del funcionamiento del programa. PyCharm también utiliza el autocompletado de código, que nos puede ahorrar mucho tiempo en el desarrollo del programa [17].

2.1.2. Visión artificial

Este apartado está dedicado a describir la tecnología más importante aplicada en este proyecto, la visión artificial. Posteriormente ofrecemos una descripción detallada de OpenCV, la librería open-source usada para implementar la visión artificial en nuestra aplicación.

2.1.2.1. Introducción a la visión artificial

La visión artificial es una disciplina científica que forma parte de la inteligencia artificial, esta utiliza métodos para procesar e interpretar las imágenes del mundo real y convertirlas en valores numéricos para que pueda ser analizado por un procesador, su objetivo es imitar el proceso de visión del ojo humano [18].

Su capacidad de procesado y análisis de imágenes la convierte en una herramienta ideal para la industria, realizando labores que mejoran la eficiencia y la calidad de ciertos trabajos que hasta ahora se consideraban costosos o repetitivos.

Como ya hemos comentado, la visión artificial ha comenzado a tomar importancia en muchos procesos industriales, algunos ejemplos en los que se utiliza la visión artificial como herramienta principal del proceso son: [19]

- Clasificación del producto por el tamaño y color.
- Verificación de posicionamiento del producto en cajas.
- Detección de defectos de impresión.
- Control de calidad de productos envasados.
- Control de tiempo del ciclo de producción.

Por otra parte, es una herramienta en desarrollo en campos como la medicina (detección de tumores, aterosclerosis, etc.), la criminología (reconocimiento de huellas dactilares, de matrículas, etc.) o la seguridad.

2.1.2.2. Librería OpenCV

En la actualidad existen gran variedad de softwares especializados en visión artificial. Algunos de ellos son: LTI, RAVL ,VLX, OpenCV, etc.
En nuestra aplicación hemos usado la librería OpenCV, debido por una parte a su compatibilidad con el sistema operativo que hemos usado (Mac IOS) y a su

acceso completamente gratuito y por otra parte a sus características técnicas como su velocidad de cálculo en los algoritmos más utilizados (como redimensionar una imagen). Como podemos ver en la figura 2.1, el tiempo de cálculo empleado por OpenCV es mucho menor que el empleado por otras librerías.

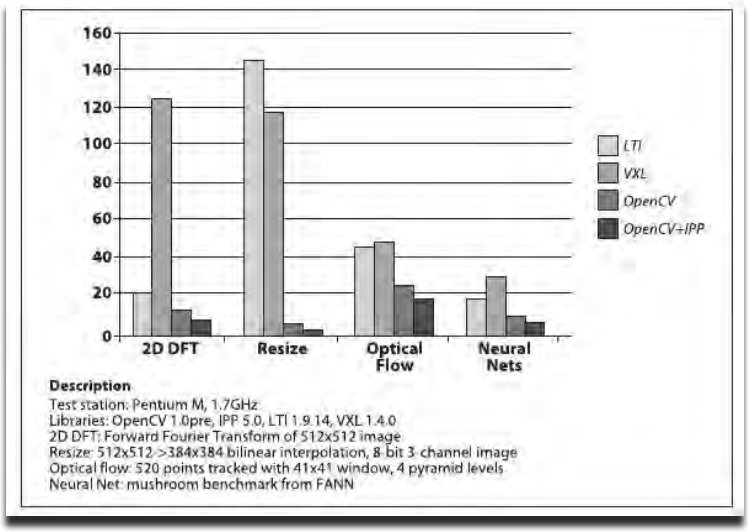

Figura 2. 1: Comparativa de velocidades de algoritmos entre diferentes librerías de visión artificial. “Learning OpenCV: Computer Vision with the OpenCV Library”

OpenCV cuyas siglas significan *Open Source Computer Vision Library*, es una librería open source por lo que cualquier usuario o empresa puede tener acceso ilimitado. Se conoce que es ampliamente utilizada tanto en grupos de investigación como a nivel profesional de empresa. Google, Microsoft y Honda son algunas de las empresas que usan dicha librería [20].

La librería fue desarrollada en un primer momento por la compañía de electrónica Intel, actualmente su mantenimiento corre a cargo de Willow Garage. [21]

OpenCV pone a disposición de los usuarios más de 2500 algoritmos optimizados, entre ellos destacan los algoritmos de detección de objetos (HOG) centrados en la detección de personas. Una ventaja de esta librería es que está preparada para trabajar a tiempo real, factor muy importante en nuestra aplicación.

La librería es compatible en diferentes lenguajes de programación como C++, C, Python, Java y MATLAB y puede ser usada por Windows, Linux, Android y Mac OS.

2.1.3. Comunicación con el piloto automático.

En este apartado vamos a describir las herramientas software utilizadas en la comunicación con el piloto automático. Hablaremos de MAVLink; protocolo

utilizado comúnmente en la comunicación entre el GCS (ground control station) y el piloto automático y de DroneKit; librería Python encargada de la comunicación con el piloto automático y de gestionar el protocolo MAVLink.

2.1.3.1. Protocolo MAVLink

La principal característica de la aplicación que estamos desarrollando es su capacidad de dirigir en tiempo real los movimientos del dron, para ello es necesario que el procesador sea capaz de intercambiar cierta información con el piloto automático, bien sea en forma de orden (dirígete a este punto) o bien sea solicitando información relevante para la aplicación (dime a qué altura vuelas).

El protocolo que usaremos para la comunicación entre el procesador on-board y el piloto automático es conocido como MAVLink, cuyo nombre completo es Micro Air Vehicle Communication Protocol.

MAVLink es un protocolo de comunicación específicamente diseñado para drones [22], utiliza una librería que tiene toda la información necesaria para el intercambio de información con el piloto automático. Esta información está desarrollada en lenguaje XML por lo que el protocolo puede ser usado en diferentes lenguajes de programación. Una ventaja que presenta el protocolo MAVLink es la posibilidad de modificar o incluso crear mensajes personalizados, de forma que se adapten a las necesidades de la aplicación que se esté desarrollando.

2.1.3.2. DroneKit

DroneKit es una API (Application Programming Interface) creada por 3D Robotics, fue desarrollada con la finalidad de ofrecer a los desarrolladores una nueva forma de crear aplicaciones para UAVs [23].

Las aplicaciones desarrolladas con DroneKit utilizan el protocolo MAVLink para comunicarse con el piloto automático, esta API es capaz de gestionar el protocolo a niveles superiores que otras librerías utilizadas para este fin, como pyMavlink. Por otra parte, las aplicaciones desarrolladas con DroneKit utilizan enlaces de baja latencia para comunicarse con el piloto automático.

Una de las ventajas que presenta DroneKit frente a otras APIs creadas con el mismo objetivo es que DroneKit está ambientada a la comunidad, por lo tanto, se puede encontrar infinidad de documentación y multitud de ejemplos prácticos para el aprendizaje autónomo de desarrollo de aplicaciones con esta librería de Python.

A pesar de que la librería ofrece la mayoría de funciones que una aplicación dron existen ciertas necesidades que no puede satisfacer por sí misma. En este caso es necesario trabajar con otras librerías especializadas que sean compatibles

con DroneKit [24]. En nuestro caso era necesario que la aplicación tuviese acceso directo a la cámara, acceso no proporcionado por DroneKit, por lo tanto, esta función ha sido derivada a OpenCV, librería que ha sido descrita en el apartado 2.1.2.

2.1.4. Herramientas de aplicación web.

En el siguiente apartado hablaremos CherryPy, una librería de Python dedicada al desarrollo de aplicaciones web y de Mapbox, una plataforma dedicada a proporcionar mapas a una gran cantidad de aplicaciones web.

2.1.4.1. CherryPy

CherryPy es un entorno de desarrollo de aplicaciones web (open source) basado en el protocolo HTTP y desarrollado en lenguaje de programación Python. Este f*ramework* permite crear aplicaciones web de forma sencilla ya que el proceso sería equivalente al desarrollo de un programa orientado a objetos. Las características del f*ramework* lo hacen accesible a desarrolladores con pocos o ningún conocimiento acerca de los protocolos internos de internet.

Por otra parte, CherryPy incluye un servidor web de serie capaz de soportar grandes cargas de tráfico. Esto hace posible que se puedan utilizar las aplicaciones web en cualquier dispositivo que tenga Python instalado. [25]

CherryPy es usado en una gran variedad de aplicaciones y páginas web conocidas, Netflix sería la más destacada. [26]

2.1.4.2. Mapbox

Otra parte importante de nuestra aplicación es la utilización de un mapa para la selección manual del punto al que el dron debe dirigirse (Punto de Misión).

Mapbox es una plataforma open source dirigida a los desarrolladores de páginas o aplicaciones web que necesitan implementar un mapa en sus respectivos trabajos [27]. Al igual que el resto de herramientas que hemos usado Mapbox es open source, por lo tanto es una herramienta al alcance de todos los usuarios. Una empresa importante que dan uso a esta herramienta es Pinterest [28].

Para poder utilizar Mapbox es necesario crear una cuenta personal en su plataforma. Ser usuario de Mapbox te permitirá utilizar sus mapas, modificarlos y personalizarlos para introducirlos en tu aplicación en la forma que deseas.

2.2. Desarrollo del código

Hemos dedicado este sub apartado a explicar de forma más detallada las partes de código encargadas de la realización de tareas más importantes. Con esto también se pretende explicar el funcionamiento del código y cómo se adapta a las necesidades requeridas por la aplicación.

En la siguiente imagen podemos ver el diagrama de estados de la aplicación.

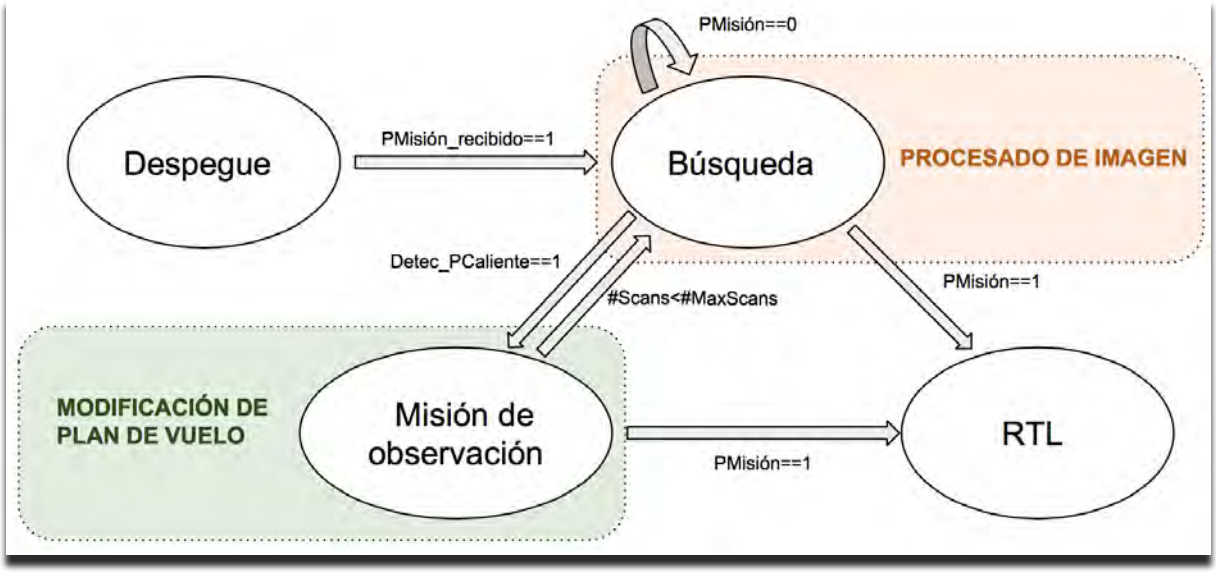

Figura 2. 2:Diagrama de estados de la aplicación.

Como podemos ver, la aplicación tiene cuatro estados, es importante comentar en qué consisten estos estados y que tareas se llevan a cabo cuando la aplicación se encuentra en ellos.

Despegue: En este estado la aplicación se inicializa y ordena al dron que despegue, una vez el dron ha alcanzado la altura deseada la aplicación lanza un servidor CherryPy, al cual podemos acceder desde cualquier ordenador, y genera un mapa Mapbox desde el cual el usuario puede escoger el Punto de Misión. La aplicación cambia de estado cuando recibe las coordenadas del Punto de Misión.

Búsqueda: Durante este estado la aplicación accede a la cámara del dron y procesa las imágenes en busca de puntos calientes. Es posible que durante el trayecto hacia el Punto de Misión no se detecten puntos calientes. Si se alcanza el Punto de Misión la aplicación pasará al estado RTL (Return To Launch) para volver a casa. Si por el contrario se detecta algún punto caliente la aplicación cambiará el estado a Misión de Observación.

Misión de observación: Cuando la aplicación se encuentra en este estado significa que ha encontrado un punto caliente o un conjunto de ellos. En este estado se crea la misión de observación alrededor del Punto Scan y se añade al plan de vuelo del dron, el cual mantiene como punto final el Punto de Misión.

Una vez realizado el scan, mientras el dron se dirige al Punto de Misión, la aplicación puede volver al estado búsqueda, siempre que el número de misiones de observación que se han realizado no supere el máximo definido por el usuario. Por otra parte, si ya se han realizado todas las misiones de observación permitidas, la aplicación esperará a que el dron alcance el Punto de Misión y cambiará al estado RTL.

RTL: Este es el estado final y solo se alcanza cuando el dron ha llegado al Punto de Misión. Una vez la aplicación ha entrado en este estado se cambiará el modo de vuelo del dron a RTL (Return to Launch), por lo tanto, el dron volverá a su Home, aterrizará y se desarmará.

Es importante señalar que el código únicamente procesará imágenes durante el estado de búsqueda de puntos calientes, es decir, el código no tomará ni tratará imágenes mientras está realizando la misión de observación o cuando se está dirigiendo al Punto de Misión tras haber completado el número de misiones de observación solicitadas por el usuario.

Como podemos ver en la figura 2.2, hemos diferenciado los estados encargados del procesamiento de imágenes y de la modificación del plan de vuelo. A continuación, comentaremos la metodología utilizada tanto en el procesado de imágenes como en la modificación del plan de vuelo, en la imagen siguiente podemos ver las funciones que hemos desarrollado dirigidas a cada campo.

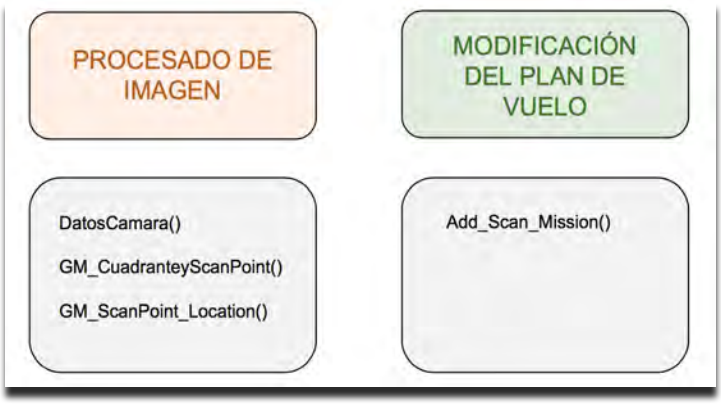

Figura 2. 3: Funciones desarrolladas en el ámbito de procesado de imagen y modificación del plan de vuelo.

2.2.1. Procesado de imagen

Como podemos ver en la figura 2.3, hemos desarrollado tres funciones exclusivamente relacionadas con el procesado de imagen. A continuación, vamos a describirlas individualmente y explicaremos la relación que hay entre ellas.

2.2.1.1. DatosCamara()

Los parámetros de la cámara influyen directamente con las características de nuestra aplicación. El FoV (*Field of View*) o campo de visión y la resolución por frames de la cámara son los que determinan el área capturada por cada imagen que tome la cámara a una altura determinada.

La siguiente imagen muestra el esquema de inputs y outputs de la función.

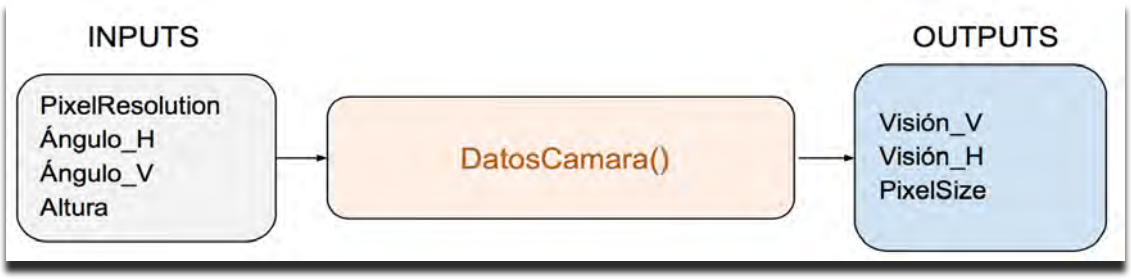

Figura 2. 4:Esquema de inputs y outputs de la función DatosCamara().

Como podemos ver en la imagen anterior, la función tiene como inputs: la resolución de la cámara (en pixel), los ángulos que determinan el FoV y por último la altura de vuelo.

Los outputs que retorna la función son: Distancia horizontal capturada, Distancia vertical capturada y un vector con el tamaño de píxel horizontal y vertical.

Las distancias horizontal y vertical capturadas hacen referencia a las dimensiones reales (en metros) que un frame puede abarcar a una altura determinada.

Por otra parte, el tamaño de píxel hace referencia a los metros representados en las dimensiones de un píxel.

Las siguientes ecuaciones nos permiten calcular las distancias horizontal y vertical capturadas por una cámara a una altura h.

$$Visión\ V = 2 \times sen(\frac{V^{\underline{o}}\ FoV}{2}) \times h \quad (2.1)$$

$$Visión\ H = 2 \times sen(\frac{H^{\underline{o}}\ FoV}{2}) \times h \quad (2.2)$$

Para conocer el área capturada en m^2 por la cámara a dicha altura únicamente se debe hacer la multiplicación del resultado obtenido de las ecuaciones (2.1) y (2.2).

El tamaño de píxel, es en realidad, el resultado que más nos interesa calcular ya que gracias a él podremos traducir distancias en píxel a distancias reales (en metros).

En las siguientes ecuaciones podemos ver cómo se calcula el tamaño de píxel, tanto en vertical como en horizontal.

$$Pixel\ Size\ V = \frac{Visión\ V(m)}{Resolución\ V(píxel)} \quad (2.3)$$

$$Pixel\ Size\ H = \frac{Visión\ H(m)}{Resolución\ H(píxel)} \quad (2.4)$$

2.2.1.2. GM_CuadranteyScanPoint

Esta función es la encargada de procesar la imagen. La función toma las imágenes de la cámara térmica y retorna las coordenadas en pixel del Punto de Scan junto con el cuadrante donde se encuentra. La siguiente imagen muestra el esquema de inputs y outputs de la función.

Figura 2. 5:Esquema de inputs y outputs de la función GM_CuadranteyScanPoint()

La metodología seguida por la función consiste en tomar una imagen de la cámara, separarla en los cuatro cuadrantes y analizar por separado estas cuatro imágenes. En la siguiente figura, mostramos los cuatro cuadrantes de una imagen.

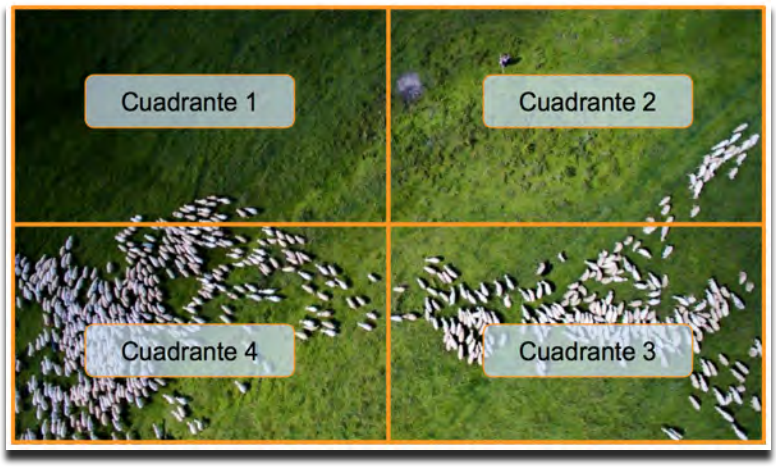

Figura 2. 6:Ejemplo de la separación por cuadrantes de una imagen.

El motivo por el que se decidió dividir la imagen en cuadrantes fue simplificar el proceso de búsqueda. El código cuantifica la cantidad de puntos calientes por cuadrante y calcula el centro de este conjunto. Una vez procesados todos los cuadrantes retornará como Punto Scan el centro del conjunto de puntos calientes del cuadrante con mayor número de puntos calientes.

La parte más importante de la función es la que se encarga de tratar la imagen y resaltar las zonas con mayor temperatura. Para facilitar el procesado de imagen se decidió que las imágenes térmicas fuesen tomadas basándose en un rango de colores grises. El color negro interpreta las zonas con menor temperatura y el blanco con mayor temperatura. A continuación, podemos observar una imagen térmica basada en escala de grises.

Figura 2. 7:Foto térmica de un rinoceronte, Fuente: HEMAV.

Se realizó un estudio acerca de los tipos de filtros que podíamos implementar en el código, el principal objetivo era evitar la aparición de falsas alarmas, es decir, considerar como punto caliente zonas de la imagen que no lo eran.

Tras analizar el resultado proporcionado por un gran número de filtros (y combinaciones entre ellos), se decidió usar el siguiente conjunto de filtros ya que su combinación fue la que produjo el menor número de falsas alarmas.

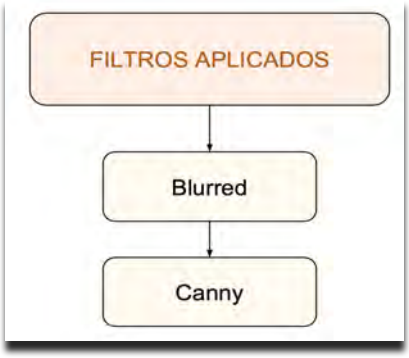

Figura 2. 8:Filtros aplicados en el procesado de la imagen.

El filtro GaussianBlur:

El filtro GaussianBlur tiene como objetivo desenfocar la imagen.

Filtro Canny:

El filtro Canny utiliza un algoritmo capaz de identificar los contornos de los objetos de la imagen. Utiliza un procedimiento llamado umbralización de histéresis el cual decide si un píxel forma parte del contorno basándose en su intensidad, se necesita que el usuario defina los parámetros de umbral máximo y umbral mínimo, si el pixel se encuentra dentro de este rango será considerado como parte del contorno, en otro caso será descartado. Es recomendable que los umbrales mantengan una relación de 2:1 o 3:1.

En nuestra aplicación hemos usado un umbral mínimo de 30 y un umbral máximo de 90. Como podemos ver en la figura 2.11 los píxeles considerados como parte del contorno se representan en blanco mientras que los que no se encuentran dentro del rango se representan en color negro.

El filtro Canny únicamente se puede aplicar a imágenes que hayan sido previamente procesadas con el filtro gray (convertir la imagen a escala de grises) y el filtro Blur. [29]

En las figuras 2.9, 2.10, 2.11 podemos observar una imagen térmica en su estado original y el resultado tras aplicar los filtros seleccionados.

Figura 2. 9:Imagen original

Figura 2. 10:Imagen tras aplicar el filtro GaussianBlur.

Figura 2. 11:Resultado tras aplicar los filtros GaussianBlur y Canny.

Como podemos ver en la figura anterior, el resultado es un conjunto de siluetas que definen las zonas con mayor temperatura de la imagen

El siguiente paso que hemos tomado ha sido la detección de estas siluetas, con la función `cv2.findContours`, la cual retorna los puntos (almacenados en vectores) que forman los contornos [30], un vector de puntos por cada contorno.

Tras determinar las zonas con más temperatura de la imagen y conocer los puntos que la definen se realiza el último cálculo para determinar si se trata de un punto caliente válido o no. Este último paso consiste en comparar el área del contorno con el área vista que tendría un rinoceronte desde la altura en la que hemos tomado la fotografía. De esta forma podemos filtrar de una forma más objetiva la información que recibimos de las fotografías térmicas. Si el área del contorno se encuentra entre el área mínima y el área máxima de un rinoceronte vista a la altura de vuelo se considerará como punto caliente.

Una vez se ha comprobado que el contorno es un punto caliente se utiliza la función `cv2.boundingRect,` esta función analiza el conjunto de puntos del contorno y retorna dos puntos de coordenadas que equivalen a los puntos abajo izquierda y arriba derecha de un rectángulo hipotético que delimitaría el contorno.

Conociendo estas coordenadas simplificamos el cálculo del Punto Scan. Como ya hemos comentado anteriormente cada cuadrante se analiza por separado, cada vez que encontramos un punto caliente en el cuadrante que estamos analizando se comparan las coordenadas de su rectángulo hipotético con las pertenecientes a los puntos calientes encontrados anteriormente y almacenamos las coordenadas X e Y máximas y mínimas encontradas. De esta forma el código conseguirá definir un rectángulo que recoja todos los puntos calientes del cuadrante que está analizado.

Tras procesar todos los cuadrantes la función retornará el cuadrante donde existan más puntos calientes y las coordenadas del centro del rectángulo hipotético que agrupa todos los puntos calientes encontrados en dicho cuadrante, estas coordenadas equivalen a las coordenadas del Punto Scan.

La siguiente figura muestra los puntos calientes detectados en la imagen original

Figura 2. 12:Puntos calientes detectados tras el procesado de la imagen.

Podemos observar que se ha identificado como puntos calientes alguna señalización de la imagen (puntos horizontales del centro de la imagen), cabe decir que esta señalización se puede eliminar modificando los parámetros de la cámara térmica. En la siguiente figura se muestra la estructura del código que acabamos de describir.

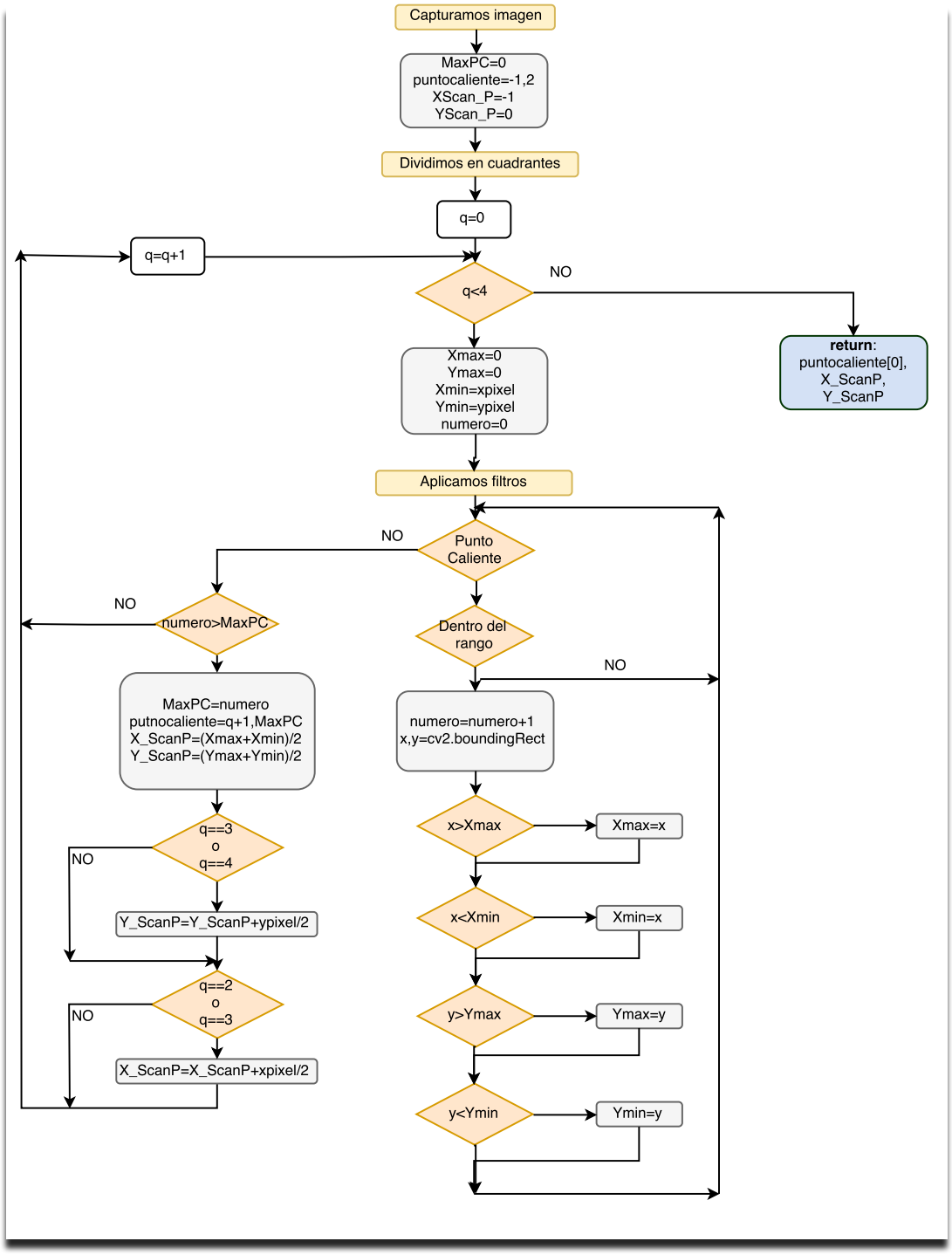

Figura 2. 13:Esquema de la función GM_CuadranteyScanpoint()

2.2.1.3. GM_ScanPoint_Location()

El objetivo de esta función es calcular las coordenadas geográficas del Punto Scan, para ello necesitamos los parámetros calculados en las funciones anteriores.

La función tiene como inputs: el tamaño del píxel vertical y horizontal calculados en la función `datoscamara()`, la resolución de la cámara, el Punto Scan (en coordenadas píxel), el cuadrante y la posición geográfica del dron es ese momento (Location Centre). Ver en la siguiente figura el esquema de inputs y outputs de la función.

Figura 2. 14:Esquema de inputs y outputs de la función GM_ScanPoint_Location.

Nuestra función usa el algoritmo `get_location_metres(Location, distEste, distNorte)`de la librería DroneKit. Esta función retorna las coordenadas geográficas de un punto situado a una distancia distEste y distNorte (en metros) respecto del punto Location.

El primer paso que realiza nuestra función es calcular estas distancias (este y norte en metros) entre el centro de la imagen (de la cual conocemos sus coordenadas geográficas) y el Punto Scan. Para ello primero calculamos las distancias en pixel, estas distancias en pixel se multiplican por el tamaño de píxel correspondiente, es decir, la distancia norte en pixel se multiplica con el tamaño de pixel vertical y la distancia este en pixel se multiplica por el tamaño de píxel horizontal. Con este cálculo conseguimos traducir las distancias en pixel a distancias reales en metros.

Tras calcular las distancias en metros la función utiliza el algoritmo de DroneKit `get_location_metres()`, con ello obtenemos un punto de coordenadas geográficas situado a la distancia Norte y distancia Este de la posición del dron, llamaremos a este punto PuntoB.

Con la finalidad de encontrar las coordenadas reales del Punto Scan es necesario rotar este PuntoB sobre la posición real del dron tantos grados como el *heading* (rumbo) que está siguiendo el dron en ese preciso momento.

Para rotar el PuntoB hemos creado la función `rotarPuntoScan()`, la cual tiene como inputs; el PuntoB, el Location (posición actual del dron en coordenadas geográficas) y el heading del dron.

El método utilizado para la rotación del PuntoB ha sido la rotación simple de un punto sobre otro punto desplazado del origen, donde el punto origen es la latitud y longitud 0º0'0" N y 0º0'0" E, el punto de rotación son las coordenadas de la posición del dron y el punto que rotamos es el PuntoB.

A continuación, podemos encontrar dos figuras que representan el resultado del Punto de Scan, antes y después de ser rotado.

Para ambas imágenes se ha simulado un punto caliente en el primer cuadrante de la imagen analizada, el heading del dron ha sido 90º en ambos casos.

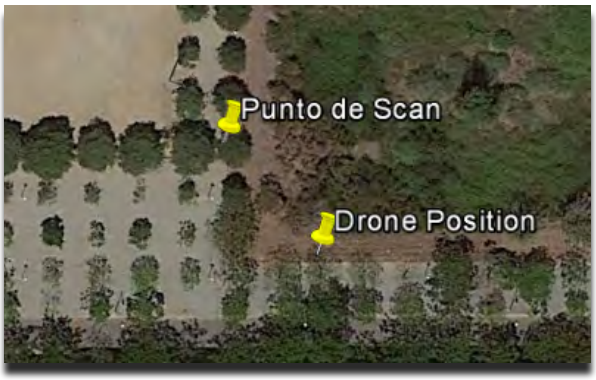

Figura 2. 15:Punto de Scan no rotado. Punto B.

Figura 2. 16: Punto de Scan rotado.

La siguiente figura muestra la arquitectura que sigue la función.

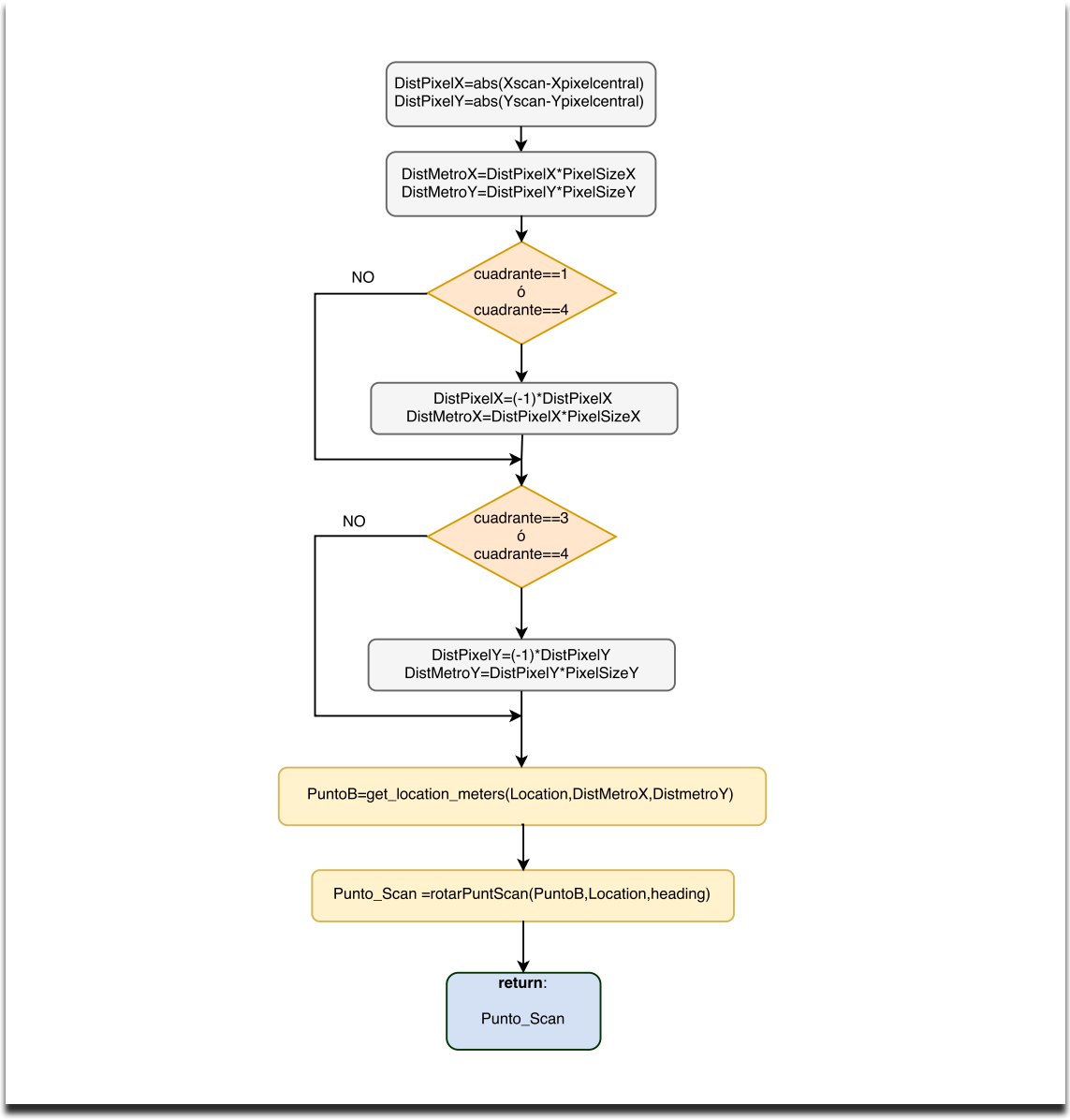

Figura 2. 17:Esquema de código de la función GM_ScanPoint_Location().

2.2.2. Modificación del plan de vuelo.

El control autónomo del dron es la función más llamativa que proporciona nuestra aplicación. En este apartado comentaremos cómo nuestro código es capaz de interactuar con el dron y modificar su plan de vuelo a través del envío de órdenes.

A continuación describiremos la función `add_scan_mission()` cuyo trabajo es por una parte diseñar una misión de observación y definir sus parámetros, los

cuales dependen de las características de la cámara y por otra parte crear un nuevo plan de vuelo que contenga la misión que hemos diseñado e introducirla en el plan de vuelo del dron.

En la siguiente figura podemos observar un esquema de la misión de observación que vamos a realizar

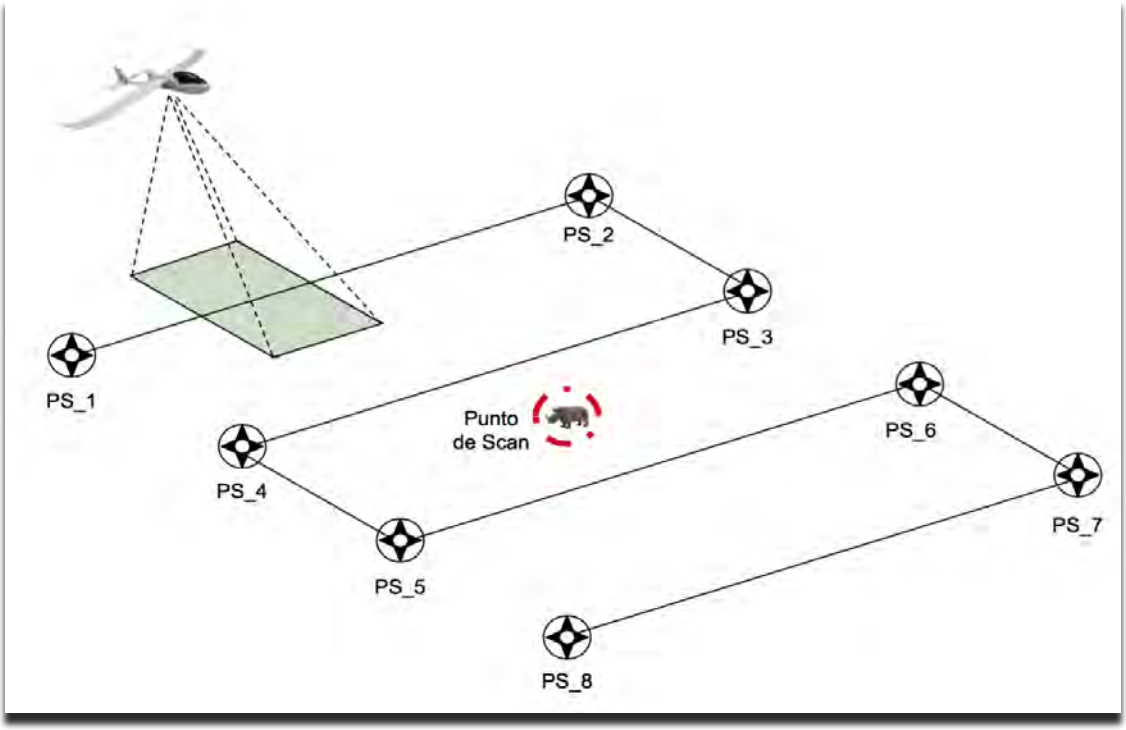

Figura 2. 18:Misión de observación.

2.2.2.1. Add_Scan_Mission()

Como ya hemos comentado, el objetivo de esta función es crear una misión de observación e introducirla en el plan de vuelo del dron. Dividiremos este apartado en dos secciones: La sección de diseño de la misión de observación, y la sección de creación del nuevo plan de vuelo y comunicación con el dron.

Diseño de la misión de observación

Con la finalidad de recoger toda la información posible acerca de los alrededores del Punto de Scan hemos decidido que la misión de observación consista en un escaneado de la zona, para ello hemos diseñado un scan de ocho puntos centrado en el Punto Scan. La siguiente figura se muestra la misión de observación.

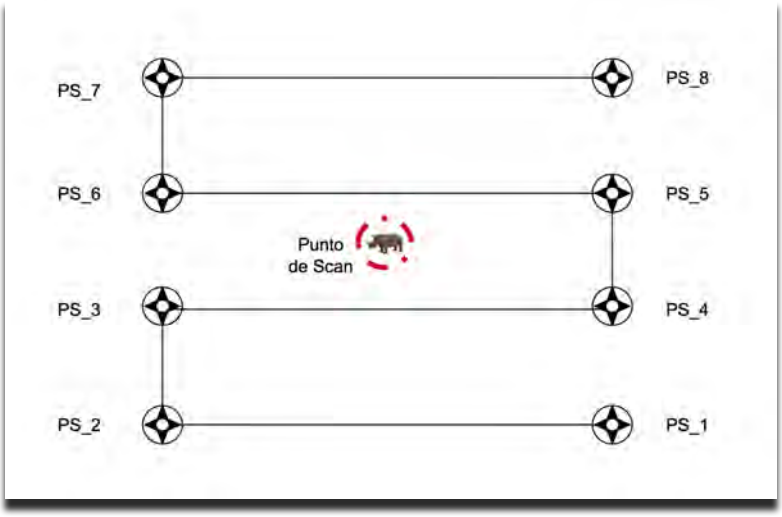

Figura 2. 19:Scan de 8 puntos realizado en la misión de observación.

Como podemos ver, los waypoints (PS) están numerados según el recorrido del dron.

Como ya hemos comentado, la finalidad de realizar la misión de observación es obtener más información acerca de los puntos calientes que hemos detectado. Para ello se ha decidido reducir la altura de vuelo durante el escaneado, de esta forma podemos conseguir mayor resolución de imagen a pesar de conseguir menos área cubierta por frame.

En cuanto a las dimensiones del Scan, por una parte, queremos que la aplicación sea lo más personalizable posible, pero garantizando la coherencia de los resultados obtenidos. Por eso hemos permitido que el usuario sea el encargado de decidir el área cubierta por el scan, pero hemos fijado la altura del scan.

Queremos que la distancia horizontal cubierta por frame sea alrededor de 8 metros, de forma que podamos capturar un rinoceronte (3 - 4 metros de longitud) dentro de un frame. Teniendo en cuenta esta condición fijamos las dimensiones del scan: 24 metros de altura y base igual al área introducida por el usuario dividida entre estos 24 metros. Ver en la siguiente figura las dimensiones del scan y del frame, representado en color verde.

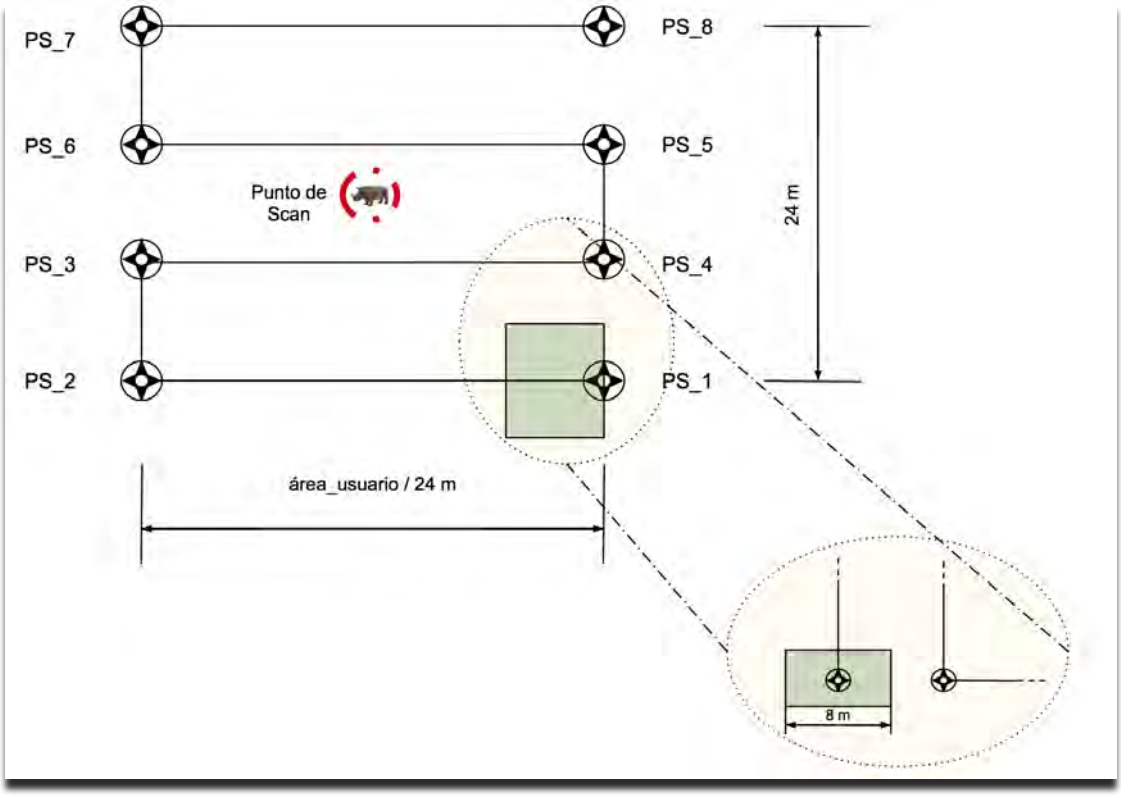

Figura 2. 20:Dimensiones del scan.

Como podemos observar en la figura 2.20, no se ha definido la distancia vertical capturada por frame, esta dependerá de los parámetros de la cámara y de la altura de vuelo, ver en el apartado 2.2.1.1.

Por otra parte, también debemos determinar la altura de vuelo durante el scan. Esta altura dependerá de los parámetros de la cámara y de la condición que hemos añadido, 8 metros de distancia horizontal capturada por frame. Podemos hallar la altura deseada utilizando ecuación la cual hemos introducido en el apartado 2.2.1.1:

$$Visión\ H = 2 \times sen(\frac{Hº\ FoV}{2}) \times h \qquad (2.5)$$

Despejando la altura y sustituyendo $Visión\ H$ por los 8 metros.

$$h = \frac{8}{2\ x\ sen(\frac{Hº\ FoV}{2})} \quad (2.6)$$

Creación del nuevo plan de vuelo y comunicación con el dron

Una vez conocemos todos los parámetros de la misión (dimensiones del scan y altura de vuelo) podemos comenzar a definir el nuevo plan de vuelo.

Esta función, al igual que la función `GM_ScanPoint_Location()`, utiliza el algoritmo de DroneKit `get_location_metres(Location,distE,distN)`, en este caso para obtener las coordenadas geográficas de los puntos que forman el scan.

DistE equivale a la mitad de la base del scan, DistN equivale a la mitad de la altura del scan, es decir 12 metros. El cálculo de los puntos del scan se puede ver en la imagen siguiente figura, la cual es un extracto del código de la aplicación, donde aSize hace referencia a DistEste y bSize a DistNorte

```
PS_1 = get_location_metres(aLocation, aSize, -bSize)
PS_2 = get_location_metres(aLocation, -aSize, -bSize)
PS_3 = get_location_metres(aLocation, -aSize, -bSize / 3)
PS_4 = get_location_metres(aLocation, aSize, -bSize / 3)
PS_5 = get_location_metres(aLocation, aSize, bSize / 3)
PS_6 = get_location_metres(aLocation, -aSize, bSize / 3)
PS_7 = get_location_metres(aLocation, -aSize, bSize)
PS_8 = get_location_metres(aLocation, aSize, bSize)
```

Figura 2. 21:Calculo de los puntos que forman el scan.

Estos puntos que hemos obtenido formarán un scan sobre el Punto de Scan, este scan estará orientado siempre al norte. Con la finalidad de hacer más eficiente la misión hemos decidido rotar este conjunto de puntos sobre su centro, Punto de Scan, tantos ángulos como el heading del dron en ese momento. Con esta rotación conseguimos evitar que el dron se desplace innecesariamente además de introducir la misión de observación de una forma más integradora.

Al igual que en la función `GM_ScanPoint_Location()` la rotación se realiza mediante una rotación sobre un punto desplazado del origen, donde el origen son las coordenadas geográficas 0º0’0” N y 0º0’0”E, el centro de rotación es el Punto de Scan y los puntos que rotan son los puntos PS, que forman el scan.

En las siguientes imágenes podemos ver un ejemplo del resultado del scan, si no lo rotamos y si lo rotamos, figuras 2.22 y 2.23 respectivamente. En ambos casos se ha simulado que el punto de misión es el mismo y el punto caliente se encuentra en el primer cuadrante. Las imágenes han sido extraídas del Mission Planner.

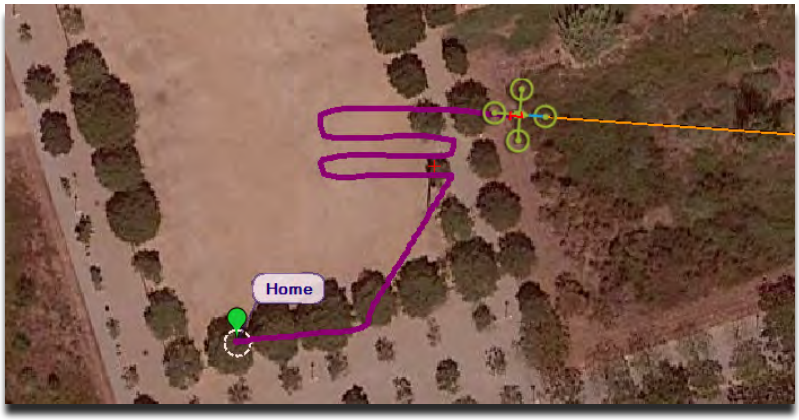

Figura 2. 22: Scan no rotado según el heading.

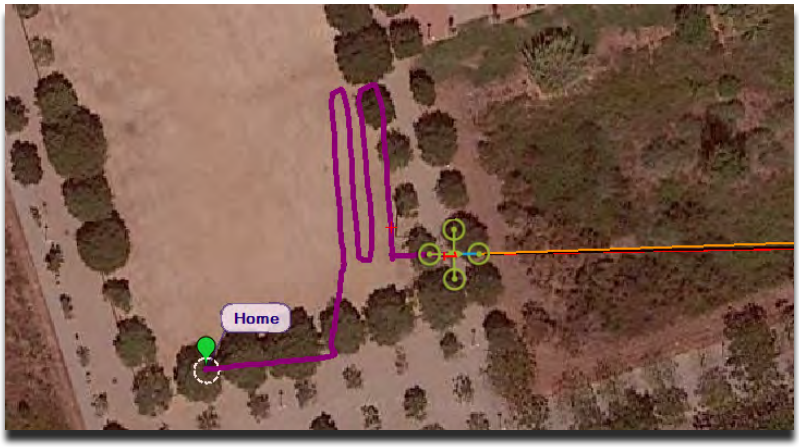

Figura 2. 23: Scan rotado según el heading.

Como podemos observar en las imágenes, ambos scans se realizan sobre el mismo Punto de Scan. En la primera imagen se genera un scan orientado al norte, en la segunda imagen el scan se orienta al heading del dron.

Una vez que conocemos las coordenadas de los puntos que forman la misión de observación podemos crear el nuevo plan de vuelo del dron, para ello utilizaremos las funciones de la librería DroneKit encargadas de la comunicación con el dron.

Es importante destacar que el plan de vuelo de un dron es en realidad un conjunto de comandos, órdenes, que dirigen su comportamiento.

La modificación del plan de vuelo de un dron comienza con la eliminación de los comandos existentes. Tras la limpieza de los comandos existentes se añaden nuevos comandos los cuales pueden ser de diferentes tipos según su finalidad; como por ejemplo realizar un despegue, un aterrizaje u ordenar al dron que se dirija a un waypoint concreto a una altura determinada. Estos comandos formarán el nuevo plan de vuelo y su organización dependerá del orden de

creación.

Una vez hemos creado el nuevo plan de vuelo con la adición de los nuevos comandos es necesario importar estos nuevos comandos al dron.

A continuación, mostramos el extracto de código correspondiente a los pasos seguidos. En la figura 2.24 podemos ver el código encargado de eliminar el plan de vuelo existente. En la figura 2.25 el código encargado de introducir nuevos comandos. En la figura 2.26 el código encargado de importar el plan de vuelo.

```
print " Clear any existing commands"
cmds.clear()
```

Figura 2. 24: Código encargado de eliminar los comandos que forman el plan de vuelo.

```
cmds.add(
    Command(0, 0, 0, mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT, mavutil.mavlink.MAV_CMD_NAV_WAYPOINT, 0, 0, 0,
            0, 0, 0, Scan1[0], Scan1[1], alturavuelo_scan))
```

Figura 2. 25: Código encargado de crear un nuevo comando para el plan de vuelo.

```
print " Upload new commands to vehicle"
cmds.upload()
```

Figura 2. 26: Código encargado de importar el nuevo plan de vuelo.

Como podemos ver en la figura 2.25, el código que introduce un nuevo comando consta de diversos parámetros. A continuación, comentaremos en qué consisten los parámetros más importantes, señalizados en la imagen.

Frame

El primer parámetro de interés determina el frame de referencia usado para las coordenadas de posición (x,y,z). En nuestro caso usamos `mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT`, el más utilizado comúnmente y que usa el sistema de coordenadas globales WGS84 para la latitud y la longitud, pero determina la altitud en relación al home (*home altitude* = 0). [31]

Tipo de comando

Este parámetro es el que determina el tipo de comando que vamos a introducir al plan de vuelo, como ya hemos comentado existen diferentes clases de comandos. En nuestro caso queremos añadir un scan el cual está compuesto por ocho waypoints, en la figura 2.25, se representa la adición de un waypoint al plan de vuelo (en concreto el primer punto del scan). Para introducir un waypoint utilizamos el parámetro `mavutil.mavlink.MAV_CMD_NAV_WAYPOINT`. Para crear un despegue usamos el tipo de comando `mavutil.mavlink.MAV_CMD_NAV_TAKEOFF`, para crear una orden de RTL (retorno a home y aterrizar) usaremos el tipo de comando `mavutil.mavlink.MAV_CMD_NAV_RETURN_TO_LAUNCH`. [32]

Parámetros 5 (Latitud), 6(Longitud) y 7(Altitud)

Los parámetros son usados en relación al tipo de comando que hemos escogido. En nuestro caso queremos añadir un punto de navegación al plan de vuelo por lo que únicamente utilizaremos los parámetros 5, 6 y 7 que hacen referencia a la latitud, longitud y altitud del nuevo punto de navegación. Como ya hemos comentado la altitud depende del frame que hemos seleccionado.

Si el dron se encuentra en modo AUTO (automático) cambiará su ruta una vez se haya importado el nuevo plan de vuelo.

El plan de vuelo que hemos creado constará de los 8 puntos que definen la misión de observación más el Punto de Misión, que se añadirá como último punto. De esta forma el dron se dirigirá al punto de misión tras finalizar el scan de la misión de observación.

Tras explicar detalladamente el funcionamiento interno de la función podemos hablar del esquema de código que sigue.

A continuación, mostramos el esquema de inputs y outputs de la función.

Figura 2. 27:Esquema de inputs y outputs de la función Add_Scan_Mission().

Como podemos ver no existen outputs, esto es debido a que la finalidad de la función es únicamente crear e importar un plan de vuelo.

En la siguiente figura se muestra la estructura de código que sigue la función.

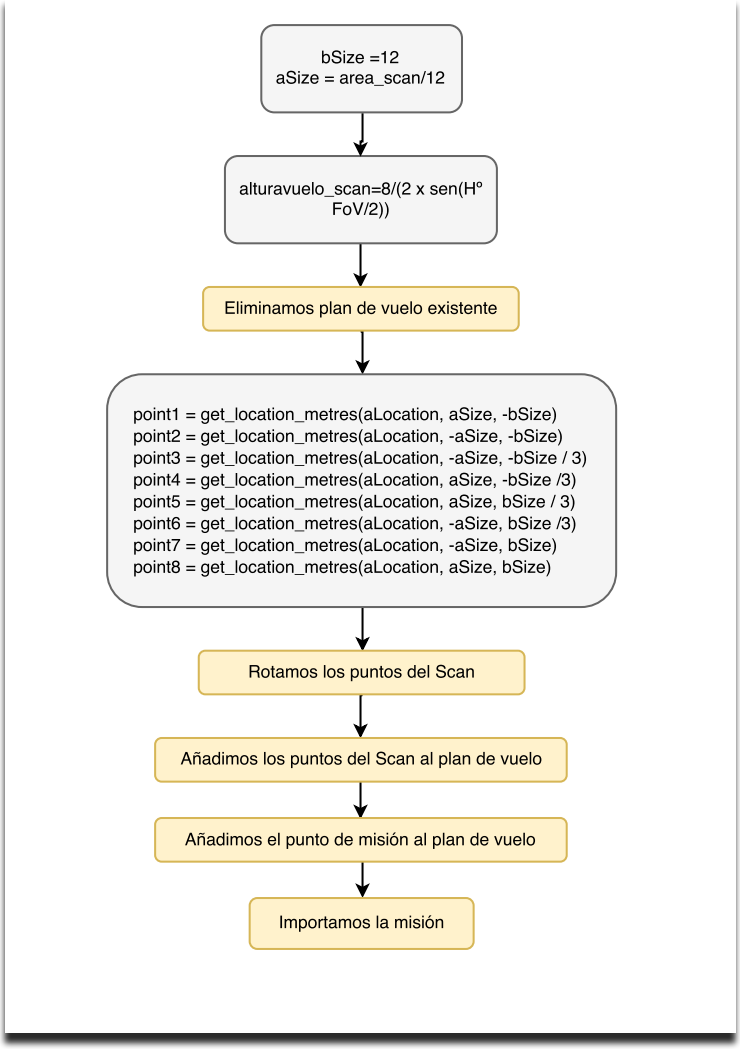

Figura 2. 28: Esquema de código que sigue la función Add_Scan_Mission().

CAPÍTULO 3: Validación de la aplicación.

En este capítulo vamos poner a prueba el funcionamiento de la aplicación desarrollada.

Realizaremos dos pruebas, una teórica y una experimental, con la finalidad de verificar el correcto funcionamiento de la aplicación y determinar sus capacidades, posibles errores y futuras mejoras.

La prueba teórica se realizará sin necesidad de utilizar un dron real, usaremos el simulador de dron ampliamente conocido como SITL (Software In The Loop). Esta prueba tiene como objetivo verificar el funcionamiento de la aplicación antes de ser utilizada en un dron real, de esta forma podremos corregir los errores que puedan poner en peligro el dron que usaremos en la validación experimental.

Por otra parte, en la prueba experimental equiparemos un dron real con los dispositivos necesarios para permitirle utilizar la aplicación. Tras equipar el dron, realizaremos un vuelo experimental en el cual pondremos a prueba las características de la aplicación. Este vuelo experimental se realizará con el dron sin hélices, es decir, nosotros mismos nos encargaremos de su movimiento teniendo en cuenta las modificaciones del plan de vuelo introducidas por la aplicación, las cuales conoceremos gracias a la herramienta Mission Planner.

3.1. Validación teórica

Como ya hemos comentado, la validación teórica consistirá en poner a prueba la aplicación mediante la simulación de un dron.

En este apartado hablaremos del software de simulación que ha sido utilizado, propondremos los datos de la misión que queremos simular, (tipo de cámara, altura de crucero etc.) y realizaremos una prueba de la aplicación mostrando el comportamiento del dron simulado.

3.1.1. Software utilizado para la simulación.

Con la finalidad de realizar la prueba teórica hemos utilizado los siguientes softwares: un simulador de drones basado en SITL (*Software in the loop*), MAVProxy y Mission Planner.

El simulador de drones nos ha proporcionado la capacidad de ver representado el comportamiento de un dron real ante diferentes estímulos, en nuestro caso las órdenes generadas por nuestra aplicación. Con la finalidad de ver los movimientos del dron simulado hemos usado Mission Planner, aplicación ampliamente conocida por ser la más usada en términos de monitoreo de

drones. Por último, hemos necesitado utilizar Mavproxy, encargado de la comunicación entre estos dos softwares.

3.1.1.1. Simulador de drones SITL

Como ya hemos comentado, utilizaremos un simulador de drones basado en la técnica “software in the loop”. Este tipo de simulación está basada en una máquina de estados, la cual nos permite conocer el comportamiento de un sistema ante cualquier entrada o evento externo[33].

El simulador SITL que vamos a implementar utiliza el software ArduPilot, software que es utilizado por los pilotos automáticos. Gracias a este hecho, SITL permite al usuario disponer de todas las plataformas de simulación que soporta ArduPilot. Entre ellas podemos encontrar plataformas de aeronaves y vehículos de tierra. Este simulador también nos permite simular otro tipo de sistemas como, por ejemplo, gimballs de cámara, sensores como un LIDAR, etc. [34]

En nuestro caso hemos decidido poner en marcha la simulación desde los mismos scripts del código de la aplicación. Para ello hemos necesitado instalar la librería DroneKit-SITL.

3.1.1.2. Mission Planner

Mission Planner es una aplicación de control de tierra completamente operativa para plataformas que utilicen pilotos automáticos basados en el software ArduPilot. Mission Planner es de código *open source.* Esta aplicación sirve como estación de tierra, el usuario puede conectar la telemetría del dron a la aplicación y controlar sus movimientos a través de esta. [35]

En esta prueba teórica únicamente utilizaremos Mission Planner con la finalidad de obtener unos resultados más visuales del comportamiento del dron, dirigido por la aplicación que hemos desarrollado.

3.1.1.3. MAVProxy

Al igual que Mission Planner, MAVProxy es un programa de estación de tierra GCS (*Ground Control Station*). El funcionamiento de MAVProxy está basado en la utilización de líneas de comandos, tanto para recibir información como para el envío de órdenes. MAVProxy permite al usuario reenviar la información del UAV a otros softwares de estación de tierra.

Utilizaremos MAVProxy como medio de comunicación entre el vehículo simulado por SITL y el Mission Planner. [36]

3.1.1.4. Procedimiento de simulación.

Como ya hemos comentado, la simulación del dron en SITL se realizará desde el script del código de la aplicación. El vehículo simulado estará conectado en la dirección tcp:127.0.0.1.5763.

Tras comenzar la simulación usaremos MAVProxy con la finalidad de crear una conexión entre el vehículo simulado y el Mission Planner. Esta comunicación se crea a través del comando que vemos en la siguiente imagen.

```
python mavproxy.py --master tcp:127.0.0.1:5763 --out 127.0.0.1:14550
```

Figura 3. 1:Comando MAVProxy encargado de crear la comunicación entre SITL y Mission Planner.

Como podemos ver, el comando creado ejecuta MAVProxy, lo conecta con el puerto TCP donde está conectado el vehículo que estamos simulando y determina un nuevo puerto de salida de información, en nuestro caso el 127.0.0.1:14550.

Los mensajes que contienen la información sobre el estado del dron están siempre estructurados según el protocolo MAVLink. Por otra parte, la comunicación entre MAVProxy y SITL se realiza con el protocolo TCP, mientras que la comunicación entre MAVProxy y el Mission Planner ser realiza con protocolo UDP.

A continuación, podemos ver una figura donde está representado el esquema de las comunicaciones de la simulación.

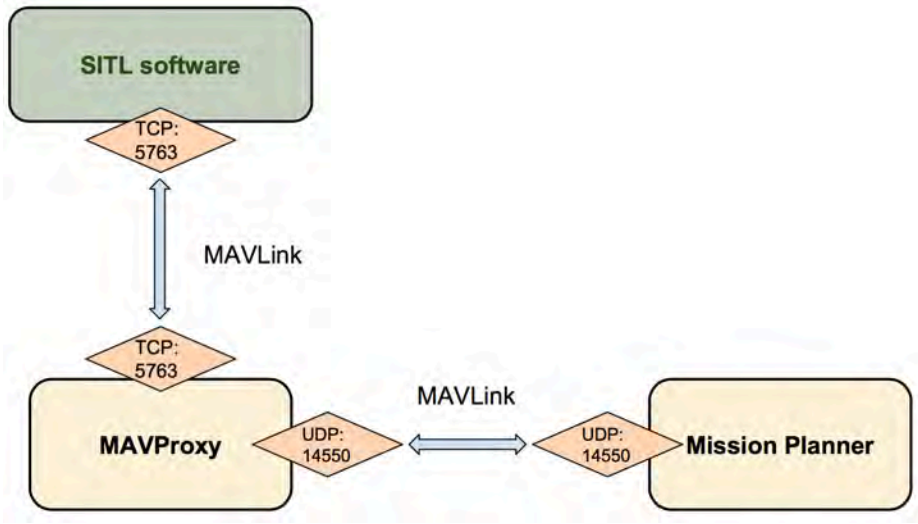

Figura 3. 2: Esquema de la comunicación entre SITL y Mission Planner.

3.1.2. Descripción de los parámetros de la misión.

Como ya hemos comentado, la aplicación ha sido desarrollada con la finalidad de adaptarse tanto a las preferencias del usuario como al tipo de cámara térmica con la que equipamos el dron.

Como hemos visto en la figura 1.6, el usuario tiene el permiso de modificar algunos de los parámetros de la misión, en la siguiente tabla encontramos los datos que hemos escogido para la realización de la prueba teórica.

Tabla 3. 1:Parámetros de la misión, prueba teórica.

Altura de vuelo (durante la búsqueda)	100 m
Número máximo de scans	2
Área del scan	576 m^2 (simulamos un scan cuadrado)
Tiempo de inicio	10 s
Tiempo de búsqueda	15 s
Parámetros de la cámara térmica	Simulación de (TAU 2 Flir 336 13 mm) FoV 25ºH x19ºV Resolución: 336x256 pixeles

Durante la prueba teórica no fue posible contar con una cámara térmica. Decidimos simular un video térmico grabado con una cámara TAU 2 Flir. Este video consiste en 10 segundos de video negro (no se detectan puntos calientes) y 5 segundos donde aparecen dos puntos calientes en la esquina superior izquierda del primer cuadrante.

Las imágenes del video que hemos creado tienen una resolución de 336 x 256 pixeles, como tendría un video térmico grabado por la cámara TAU 2 Flir. Gracias a este video hemos podido comprobar el funcionamiento de la aplicación. En la siguiente imagen podemos ver un frame del video que hemos creado, en él aparecen los puntos calientes que hemos simulado.

Figura 3. 3:Frame del video creado donde aparecen puntos calientes simulados.

3.1.3. Simulación y resultados obtenidos.

Tras determinar los parámetros de la misión e introducirlos en la aplicación, podemos poner en marcha el sistema.

En este apartado mostraremos cómo funciona la aplicación y cómo es la interacción con el usuario.

La aplicación comenzará su funcionamiento cerciorándose del estado del dron, tras comprobar que todos los sistemas del dron están operativos le ordenará que se arme y posteriormente realice el despegue hasta la altura que ha introducido el usuario.

Una vez el dron ha alcanzado la altura deseada, la aplicación genera un servidor web, al cual podemos acceder desde cualquier dispositivo conectado en la misma red. Este servidor genera un mapa de Mapbox, desde el cual el usuario es capaz de seleccionar manualmente el Punto de Misión. En la siguiente imagen podemos ver el servidor y mapa, al cual hemos accedido a través de nuestro teléfono móvil.

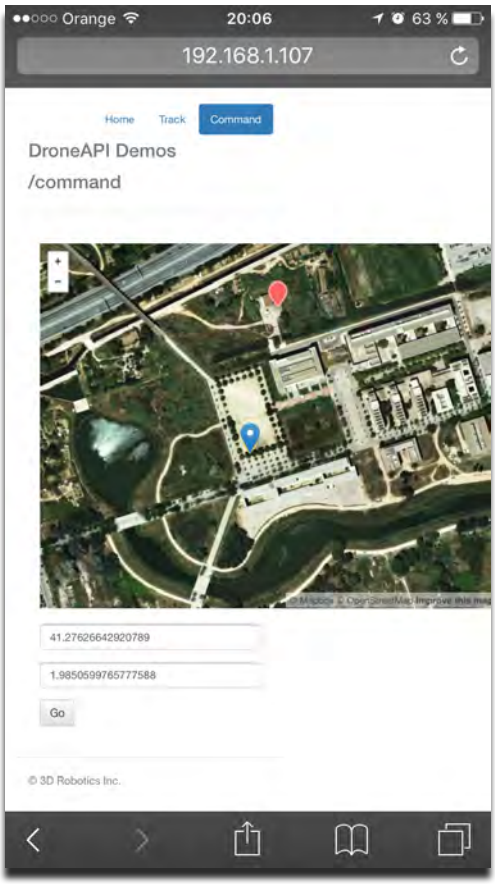

Figura 3. 4: Servidor web de la aplicación, acceso a través de un dispositivo móvil.

Las coordenadas del Punto de Misión son enviadas a la aplicación, acto seguido el dron comienza el desplazamiento hacia este punto. Pasados los 10 segundos pertenecientes al de Tiempo de inicio, la aplicación comenzará con el procesado de imagen térmica.

Cuando la aplicación detecta los puntos calientes, vistos en la figura 3.3, calcula la posición del Punto de Scan y crea de forma automática una misión de observación. En esta simulación hemos escogido un área de scan de 576 $\mathbf{m^2}$, esto genera un scan cuadrado. (Por defecto la altura del scan son siempre 24m, la base del scan serán 24 m para conseguir el área requerida). En la siguiente imagen, que hemos extraído del Mission Planner, podemos ver el dron realizando la misión de observación. Como podemos observar, el dron ha reducido notablemente su altura con la finalidad de conseguir un frame de mayor resolución, por otra parte, esta altura garantiza una distancia horizontal capturada por frame de 8m.

Figura 3. 5: Imagen extraída de Mission Planner, dron realizando la misión de observación.

Tras terminar con la misión de observación, el dron recupera la altura de vuelo impuesta por el usuario. Pasado el tiempo de nueva búsqueda, la aplicación vuelve a procesar el video térmico en búsqueda de más puntos calientes. Este proceso se repite tantas veces como el usuario haya determinado con el parámetro número máximo de scans. Una vez el dron alcanza el Punto de Misión, la aplicación ordena al dron que retorne a casa y aterrice. En la siguiente imagen, podemos observar el dron en modo RTL.

Figura 3. 6:Foto extraída de Mission Planner, el dron ha alcanzado el Punto de Misión y retorna a casa.

Como podemos observar, el dron está en modo RTL.

3.2. Validación experimental

La validación experimental ha tenido como objetivo comprobar el funcionamiento de las características básicas del sistema. Por desgracia, no disponemos de ninguna cámara térmica, tampoco de ningún módulo 3G que nos permita la conexión a internet. Por ello hemos tenido que modificar la misión, en lugar de una cámara térmica hemos usado una webcam, por lo que hemos modificado el código de la aplicación de manera que detecte el color rojo. A su vez hemos añadido la opción de guardar el frame en el cual se ha detectado el objeto. Por otro lado, hemos eliminado la parte del código que lanzaba un servidor web. Ahora el punto de misión será introducido antes del inicio de la aplicación, junto con los parámetros de la cámara y los tiempos de inicio y nueva búsqueda.

Con la prueba experimental queremos comprobar que:

- La aplicación es capaz de reaccionar a la presencia de objetos que destaquen sobre el resto de la imagen (en este caso objetos rojos).
- El procesado de la imagen es lo suficientemente rápido para crear la misión de observación sobre el punto deseado.
- La aplicación es capaz de comunicarse correctamente con el piloto automático del dron.
- La aplicación es capaz de dirigir el dron de forma autónoma.

3.2.1. Componentes del sistema

3.2.1.1. Piloto automático Pixhawk

El piloto automático que hemos usado en la prueba experimental es el Pixhawk 4. Uno de los más conocidos y usados. Esta controladora es fabricada originalmente por Robotics 3D.

3.2.1.2. Procesador RaspberryPi 3

El procesador que hemos usado ha sido la RaspberryPi 3. Este procesador dispone de una CPU de cuatro-núcleos ARM Cortex-A53 a 1,2 GHz, Wifi integrado (802.11n) y Bluetooth 4.1. El motivo de escoger este procesador ha sido su facilidad de aprendizaje y la compatibilidad de uso con el piloto automático que hemos escogido. [37]

Como ya hemos comentado anteriormente, la aplicación que hemos desarrollado necesita estar instalada en el procesador. A su vez, debe existir una comunicación directa entre el piloto automático y la RaspberryPi. A continuación, mostramos el esquema de conexión entre estos dos dispositivos.

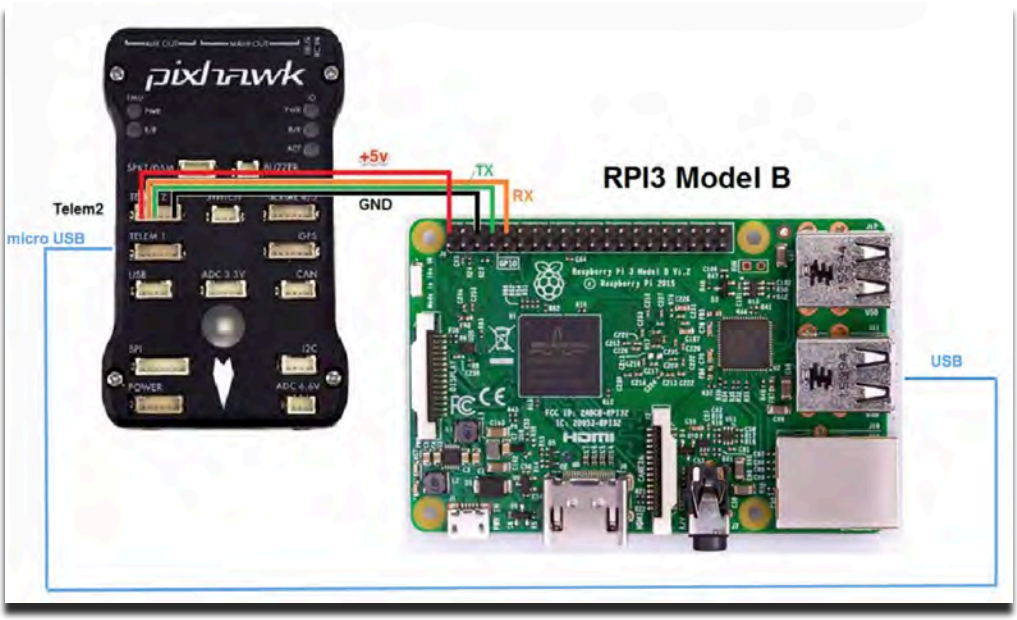

Figura 3. 7: Conexión Pixhawk - RaspberryPi 3 [38]

Como podemos ver, tenemos dos opciones de comunicación. La primera de ellas consiste en una comunicación mediante la conexión de los puertos UART. Por otra parte, podemos realizar la comunicación a través de los puertos USB de los dos dispositivos.

En ambos casos es necesario habilitar los puertos de comunicación serie de la RaspberryPi. Habilitamos el puerto serie ttyS0 si la comunicación se realiza mediante los puertos UART. Si por el contrario realizamos la conexión USB el puerto serie que utilizaremos será el ttyACM0. En nuestro caso hemos realizado la comunicación mediante la conexión USB.

3.2.1.3. Webcam

En cuanto a la cámara, hemos usado una Webcam Logitech C110, cuya resolución del frame es 640 x 480 pixeles, por otra parte FoV es de 68º diagonal, es decir, 45º horizontal y 58º vertical.[39]

3.2.1.4. Plataforma dron

Hemos usado el dron DJI F 550, multicóptero de 6 motores. En la siguiente figura podemos ver una imagen del dron equipado con todos los elementos mencionados.

Figura 3. 8: Plataforma equipada.

3.2.1.5. Esquema del sistema

En la siguiente figura mostramos un esquema de la conexión y comunicación de los dispositivos utilizados. Cabe destacar que no se han representado los componentes básicos del dron como son la telemetría, los motores, el GPS o la brújula.

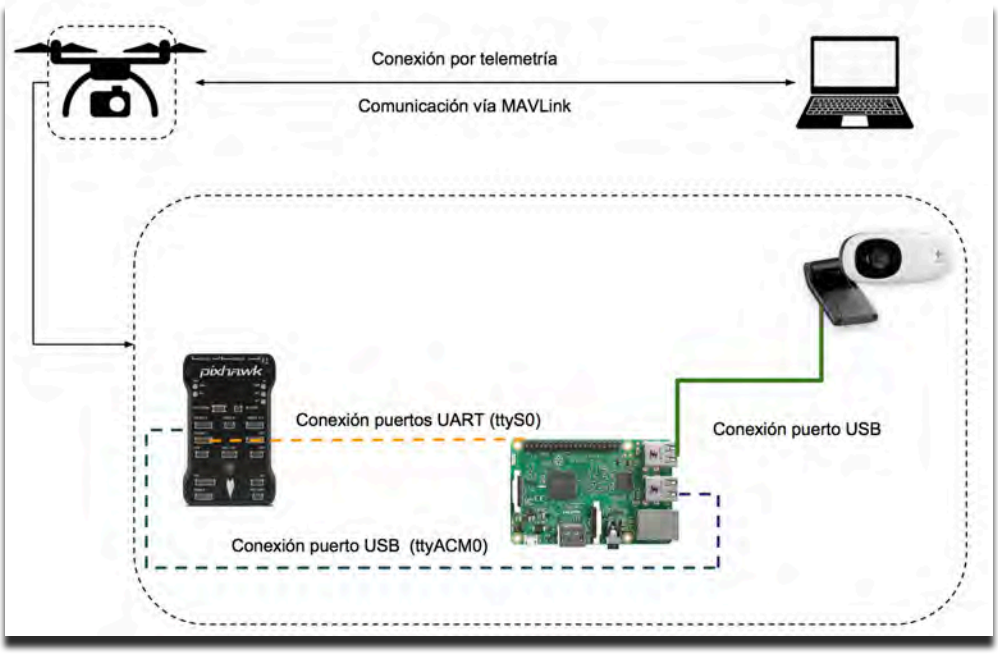

Figura 3. 9: Esquema de comunicación y conexión del sistema.

La comunicación entre el dron y la estación de tierra se realiza a través de las antenas de telemetría, una embebida en el dron, la otra conectada a un puerto COM del ordenador.

3.2.2. Descripción de la misión experimental

Como ya hemos comentado, no disponemos de los recursos necesarios que nos permitan comprobar el funcionamiento de la aplicación diseñada. Es por esto que hemos tenido que adaptar la misión a los componentes de los que disponemos.

La prueba experimental se realizará con el dron DJI F550 que hemos equipado. Como ya hemos comentado, la prueba se realizará con el dron sin hélices, nosotros mismos nos encargaremos del movimiento del dron teniendo en cuenta las modificaciones de la misión introducidas por la aplicación. De esta forma seremos capaces de comprobar el funcionamiento de la aplicación sin la necesidad de realizar un vuelo real.

En la prueba experimental el objetivo será detectar un objeto rojo y realizar una pequeña misión sobre este objeto. Hemos configurado la aplicación de forma que el Punto de Misión se encuentre a una distancia de 10 m al Este y 10 m al Norte del Home del dron.

En cuanto a la altura de vuelo, hemos decidido que la altura de vuelo durante la búsqueda del objeto rojo sea de 1,5 m.

Con la finalidad de simplificar la prueba también hemos modificado la misión de observación. Cuando la aplicación detecte el objeto rojo generará una pequeña misión entorno al punto. Esta misión consistirá en un cuadrado de 1 m^2 formado por 4 waypoints entorno al objeto. Por otra parte, hemos decidido que la altura de vuelo durante la misión de observación sea de 0,5 m.

Como hemos visto en el apartado anterior, la cámara tiene un FoV de 45º Horizontal y 58º Vertical. Esto implica que a una altura de 1,5 m será capaz de capturar un área de m^2 (1,45 m vertical x 1,14 m horizontal) . En cuanto al objeto rojo, utilizaremos una cartera roja.

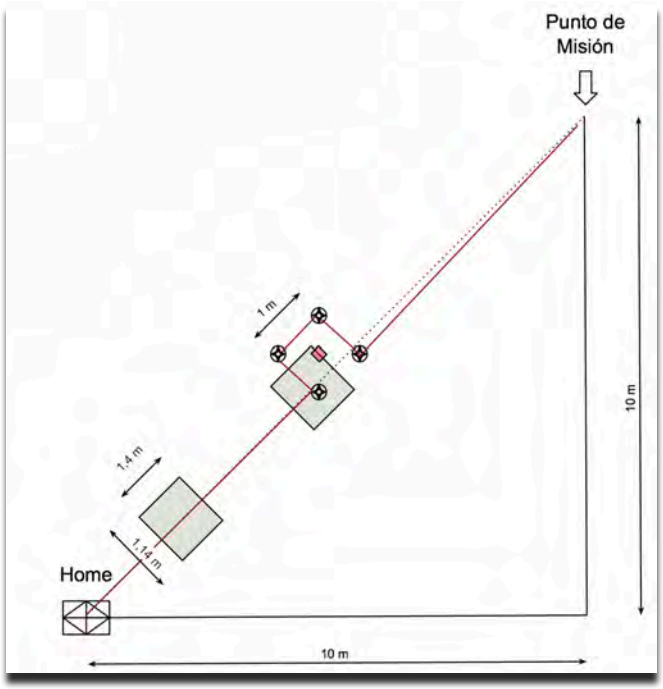

Figura 3. 10: Misión realizada en la prueba experimental.

3.2.3. Realización de la prueba

La prueba experimental se ha realizado dentro del Campus del Baix Llobregat, concretamente tras el edificio RDIT, el único lugar del campus donde se recibe señal GPS.

Con la finalidad de comprobar el correcto comportamiento de la aplicación hemos conectado una pantalla portátil a la RaspberryPi de esta forma podemos ver los mensajes creados por la aplicación. En la siguiente imagen podemos ver donde hemos situado el Home para la prueba y por consiguiente donde se encuentra el punto de misión.

Figura 3. 11: Home y punto de misión en la prueba experimental.

Tras simular el despegue a 1,5 m cambiamos el modo del dron a AUTO y comenzamos a desplazarnos en dirección al punto de misión, tres segundos después (tiempo de inicio) la aplicación comienza a procesar las imágenes. Tras recorrer unos cuantos metros la aplicación detecta el objeto rojo que hemos colocado previamente y modifica el plan de vuelo del dron, añadiendo la misión de observación.

En la siguiente imagen podemos ver el nuevo plan de vuelo, el cual ha sido introducido por la aplicación tras detectar el objeto, en este caso mostramos el mensaje creado por la aplicación. Se ha detectado el objeto en el cuarto cuadrante.

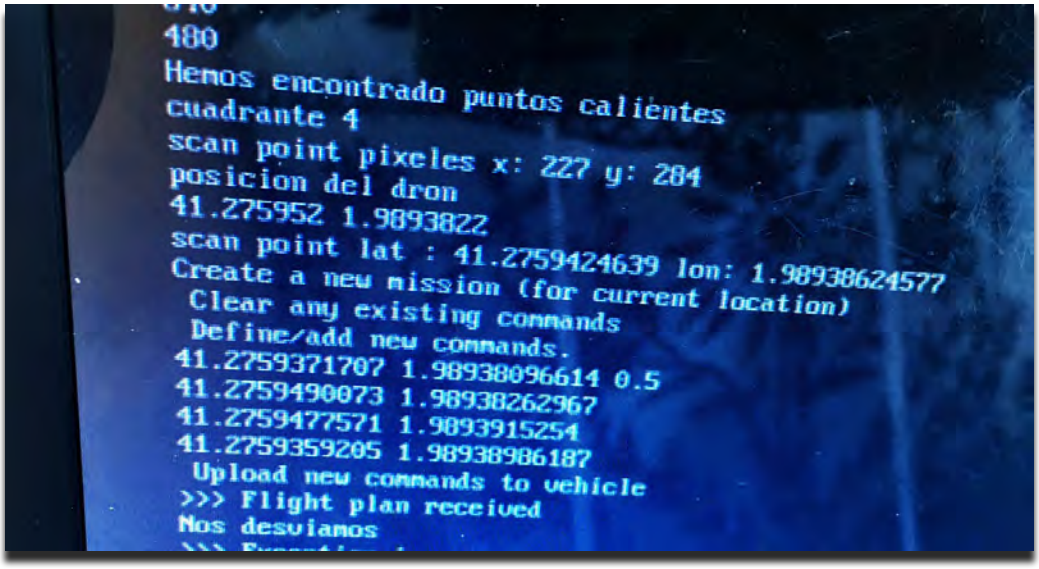

Figura 3. 12: Plan de vuelo creado e importado por la aplicación.

A continuación, mostramos el frame que ha guardado la aplicación procesado a posteriori con el mismo algoritmo que utiliza la aplicación.

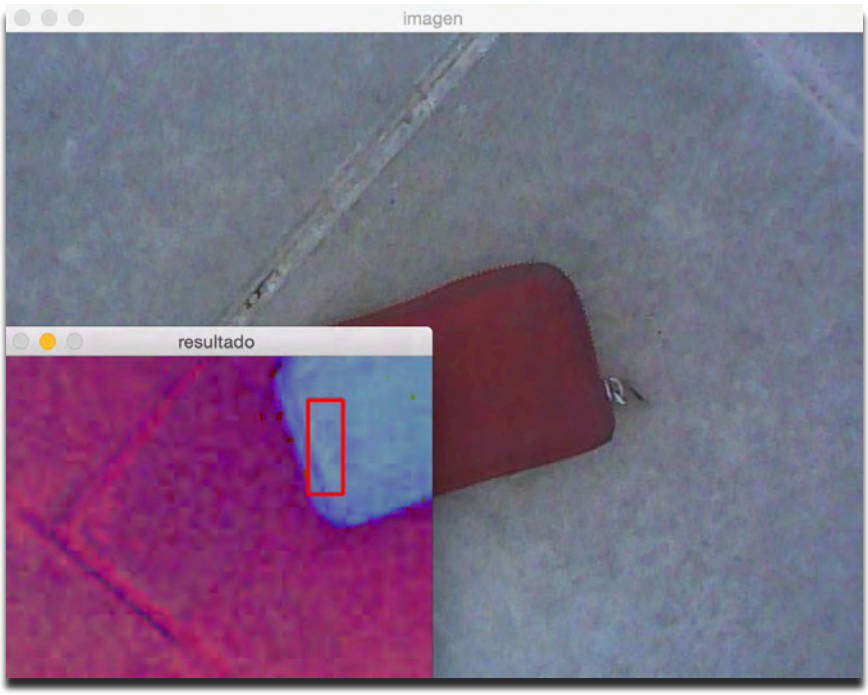

Figura 3. 13: Imagen tomada por la aplicación procesada.

Podemos observar que efectivamente la aplicación ha detectado un objeto rojo en el cuarto cuadrante del frame. Podemos ver que la detección no ha sido tan precisa como esperábamos ya que únicamente ha considerado una pequeña zona del monedero como un objeto rojo, esto se debe a una mala elección de los parámetros en el reconocimiento del color.

Con la prueba experimental hemos podido comprobar que la aplicación cumple con los objetivos propuestos. Por una parte, el sistema ha sido capaz de acceder a la cámara durante el estado de búsqueda, procesar las imágenes y determinar la presencia del objeto rojo. Por otra parte, la aplicación ha sido capaz de comunicarse con el piloto automático de forma eficaz tanto para obtener información como para importar el nuevo plan de vuelo. Hemos podido comprobar que la misión de observación se ha creado según el diseño propuesto, un scan de 1m cuadrado entorno al Punto de Scan. A su vez, hemos podido comprobar que existe un pequeño retardo entre la importación del plan de vuelo, ordenado por la aplicación, y la modificación definitiva del plan de vuelo del dron. En cualquier caso, este retardo no interfiere de forma negativa en el comportamiento de la aplicación.

Es importante comentar que los resultados obtenidos han sido satisfactorios teniendo en cuenta las características de la prueba que hemos realizado, una simulación de vuelo con un dron sin hélices. Consideramos que sería necesario realizar una prueba con un vuelo real con la finalidad de probar las limitaciones de la aplicación en una situación más realista.

CAPÍTULO 4: Conclusiones.

Las características de los drones como su manejabilidad y su capacidad de recolectar y/o retransmitir información los hacen herramientas útiles para realizar ciertas labores que se escapan de las capacidades de los humanos, como actividades de recolección de información en zonas de difícil acceso o labores de búsqueda de emergencia.

La lucha contra la caza furtiva es sin duda un escenario idóneo para la aplicación de los drones en labores de recolección de información, pueden resultar herramientas sustituyentes o complementarias al uso de helicópteros en labores de vigilancia del parque.

La implementación de visión artificial puede resultar de gran interés ya que es capaz de dotar al dron de un cierto grado de inteligencia artificial.

Cabe decir que los resultados proporcionados por la aplicación se ven directamente relacionados con las características tanto de la plataforma como de la cámara térmica embebida en el dron. Se recomienda utilizar cámaras térmicas con el máximo FoV y resolución posible, de esta forma conseguiremos capturar más área por frame sin sacrificar la calidad de la imagen.

Consideramos que la aplicación desarrollada cumple con los objetivos propuestos: la realización de una aplicación dron de bajo coste, compatible con cualquier tipo de plataforma y equipo térmico, capaz de modificar el propio plan de vuelo del dron con la finalidad de obtener mayor calidad de la información recolectada.

Por otra parte, consideramos que sería conveniente introducir diferentes mejoras con la finalidad de crear un producto final de mayor calidad. En trabajos futuros se pretenderá trabajar los siguientes aspectos de la aplicación:

- Mejorar el algoritmo de detección de puntos calientes.
- Introducir un algoritmo de diseño de la misión de observación.
- Preparar la aplicación ante la pérdida de señal GPS.
- Preparar la aplicación ante cambios de relieve en el terreno.
- Introducir un algoritmo de procesado de imágenes durante la misión de observación con la finalidad de detectar si el punto caliente se trata de un rinoceronte.
- Introducir las tecnologías desarrolladas previamente por el proyecto Ranger Drone capaces de enviar alarmas en tiempo real en caso de identificar la presencia de un rinoceronte o intruso.

Por último, comentar que la aplicación desarrollada podría extrapolarse a otros campos diferentes a la lucha contra la caza furtiva. Sus características la convierten en una herramienta útil en labores de búsqueda y reconocimiento del terreno.

Bibliografía

[1] *"Los drones fracasan en la mayor reserva de rinocerontes del mundo | Ciencia | EL PAÍS." [Online]. Available: http://elpais.com/elpais/2017/03/31/ciencia/1490957765_514434.html.*

[2] *"'Drones' contra la caza furtiva | Ciencia | EL MUNDO." [Online]. Available: http://www.elmundo.es/ciencia/2014/01/11/52d07306268e3ec6058b456a.html.*

[3] *SANParks, "SANParks Annual Report 2015/16," pp. 1–213, 2015.*

[4] *"South African rhino poaching numbers show need for urgent action | Stories | WWF." [Online]. Available: https://www.worldwildlife.org/stories/south-african-rhino-poaching-numbers-show-need-for-urgent-action.*

[5] *"Rhino Poaching Update | Rhino Conservation | Kruger Park News." [Online]. Available: http://www.krugerpark.co.za/krugerpark-times-e-6-rhino-poaching-update-25237.html.*

[6] *"Mundo Drone: Historia de los Drones." [Online]. Available: http://mundrone.blogspot.com.es/p/historia-de-los-drones.html.*

[7] *"Remote Piloted Aerial Vehicles - The Radioplane Target Drone." [Online]. Available: http://www.ctie.monash.edu.au/hargrave/rpav_radioplane3.html.*

[8] *"California utiliza sus drones en apoyo a la lucha contra incendios | Drones Argentina." [Online]. Available: http://www.drones-argentina.com.ar/2013/09/20/california-utiliza-sus-drones-en-apoyo-a-la-lucha-contra-incendios/.*

[9] *"Auxdron: dron diseñado para salvamento acuático. | Dronepedia." [Online]. Available: http://www.dronepedia.es/blog/auxdron-dron-disenado-para-salvamento-acuatico.*

[10] *"Los drones salvan vidas en la montaña." [Online]. Available: http://www.lavozdegalicia.es/noticia/ourense/laza/2015/10/30/drones-salvan-vidas-montana/0003_201510O31C9993.htm.*

[11] *"Presentan un prototipo de «drone ambulancia» capaz de transportar un desfibrilador." [Online]. Available: http://www.lavozdegalicia.es/noticia/sociedad/2014/10/28/presentan-prototipo-drone-ambulancia-capaz-transportar-desfibrilador/00031414525508575685409.htm.*

[12] *M. Mulero-Pázmány, R. Stolper, L. D. van Essen, J. J. Negro, and T. Sassen, "Remotely Piloted Aircraft Systems as a Rhinoceros Anti-Poaching Tool in Africa,"* PLoS One*, vol. 9, no. 1, p. e83873, Jan. 2014.*

[13] *"Drones contra la caza furtiva en Sudáfrica | ToDrone." [Online]. Available: http://www.todrone.com/drones-combatir-caza-furtiva-sudafrica/.*

[14] *"General Information." [Online]. Available: http://www.caa.co.za/Pages/RPAS/Remotely Piloted Aircraft Systems.aspx.*

[15] "An Aerial View of Kruger National Park | Nelspruit." [Online]. Available: http://showme.co.za/nelspruit/lifestyle/an-aerial-view-of-kruger-national-park/.

[16] "1. Introducción — Tutorial de Python 3.6.0 documentation." [Online]. Available: http://docs.python.org.ar/tutorial/3/real-index.html.

[17] "PyCharm :: Features." [Online]. Available: https://www.jetbrains.com/pycharm/features/.

[18] R. Klette, Concise Computer Vision*. .*

[19] "Aplicaciones | Visión Artificial: Especialistas en sistemas de visión artificial | INFAIMON." [Online]. Available: http://www.infaimon.com/es/menu/aplicaciones.

[20] "About - OpenCV library." [Online]. Available: http://opencv.org/about.html.

[21] "What is OpenCV?" [Online]. Available: http://opencv-srf.blogspot.com.es/2010/09/what-is-opencv.html.

[22] "MAVLink: protocolo de comunicación para drones." [Online]. Available: http://www.xdrones.es/mavlink/.

[23] "Welcome to DroneKit-Python's documentation!" [Online]. Available: http://python.dronekit.io/.

[24] "DroneKit | 3DR Solo Development Guide." [Online]. Available: https://dev.3dr.com/concept-dronekit.html.

[25] "CherryPy-Tutorial-Spanish | Scratchpad | Fandom powered by Wikia." [Online]. Available: http://scratchpad.wikia.com/wiki/CherryPy-Tutorial-Spanish.

[26] "Foreword — CherryPy 10.2.3.dev3+g9d2628a.d20170603 documentation." [Online]. Available: http://docs.cherrypy.org/en/latest/intro.html#success-stories.

[27] "About | Mapbox." [Online]. Available: https://www.mapbox.com/about/.

[28] "Pinterest: Scaling Beautiful Maps | Mapbox." [Online]. Available: https://www.mapbox.com/blog/pinterest-scaling-beautiful-maps/.

[29] "OpenCV: Canny Edge Detection." [Online]. Available: http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html.

[30] "Structural Analysis and Shape Descriptors — OpenCV 2.4.13.2 documentation." [Online]. Available: http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours. [Accessed: 25-May-2017].

[31] "DroneKit-Python API Reference." [Online]. Available: http://python.dronekit.io/automodule.html.

[32] "MAVLink Mission Command Messages (MAV_CMD) — Mission Planner documentation." [Online]. Available: http://ardupilot.org/planner/docs/common-mavlink-mission-command-messages-mav_cmd.html.

[33] "Simulador de drones basado en SITL | Simulador de dron." [Online]. Available: http://www.xdrones.es/simulador-de-drones-simulador-de-dron/.

[34] "SITL Simulator (Software in the Loop) — Dev documentation." [Online].

Available: http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[35] "Mission Planner Overview — Mission Planner documentation." [Online]. Available: http://ardupilot.org/planner/docs/mission-planner-overview.html.

[36] "MAVProxy Developer GCS — Dev documentation." [Online]. Available: http://ardupilot.org/dev/docs/mavproxy-developer-gcs.html#mavproxy-developer-gcs.

[37] "Raspberry Pi 3 tiene un procesador de 64-bit y WiFi integrado - Engadget en español." [Online]. Available: http://es.engadget.com/2016/02/29/raspberry-pi-3/.

[38] "Communicating with Raspberry Pi via MAVLink — Dev documentation." [Online]. Available: http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html.

[39] "FOV." [Online]. Available: https://community.logitech.com/s/question/0D53100006f2ET2CAM/fov.