UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Instituto Universitario de Microelectrónica Aplicada

Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación

Trabajo Fin de Máster

Programación y optimización del vuelo de un dron orientado a la agricultura inteligente

Autor: Carlos Herrera Falcón

Tutor(es): José Francisco López Feliciano
Pablo Sebastián Horstrand Andaluz

Fecha: Julio 2017

t +34 928 451 086

f +34 928 451 083

iuma@iuma.ulpgc.es

www.iuma.ulpgc.es

Campus Universitario de Tafira

35017 Las Palmas de Gran Canaria

Versión: 2.0 – 14 junio 2011

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Instituto Universitario de Microelectrónica Aplicada

Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación

Trabajo Fin de Máster

Programación y optimización del vuelo de un dron orientado a la agricultura inteligente

HOJA DE FIRMAS

Alumno/a:	Carlos Herrera Falcón	Fdo.:
Tutor/a:	José Francisco López Feliciano	Fdo.:
Tutor/a:	Pablo Sebastián Horstrand Andaluz	Fdo.:

Fecha: Julio 2017

t +34 928 451 086

f +34 928 451 083

iuma@iuma.ulpgc.es

www.iuma.ulpgc.es

Campus Universitario de Tafira

35017 Las Palmas de Gran Canaria

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Instituto Universitario de Microelectrónica Aplicada

Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación

Trabajo Fin de Máster

Programación y optimización del vuelo de un dron orientado a la agricultura inteligente

HOJA DE EVALUACIÓN

Calificación: .....

Presidente Fdo.:

Secretario Fdo.:

Vocal Fdo.:

Fecha: Julio 2017

t +34 928 451 086

f +34 928 451 083

iuma@iuma.ulpgc.es

www.iuma.ulpgc.es

Campus Universitario de Tafira

35017 Las Palmas de Gran Canaria

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ACRÓNIMOS

Acrónimos	
ENABLE-S3	European Initiative to Enable Validation for Highly Automated Safe and Secure Systems
IUMA	Instituto Universitario de Microelectrónica Aplicada
CEI	Centro de Electrónica Industrial de la Universidad Politécnica de Madrid
SDK	Software Development Kit
IDE	Integrated Development Environment
FOV	Field Of View
GPS	Global Positioning System
GLONASS	Sistema global de navegación por satélite de la Federación Rusa
ESA	European Space Agency
HELICOiD	HypErspectraL Imaging Cancer Detection

MEMORIA DESCRIPTIVA

Capítulo 1. Introducción

1.1 Antecedentes

La Unión Europea aprobó en 2016 un proyecto de investigación dotado con más de 34 millones de euros, en el cual participa el Instituto Universitario de Microelectrónica Aplicada (IUMA) de la Universidad de Las Palmas de Gran Canaria. **ENABLE-S3** (acrónimo del proyecto, de su nombre en inglés "European Initiative to Enable Validation for Highly Automated Safe and Secure Systems") está compuesto por un total de 74 socios de 15 países distintos, de los cuales 8 participantes son españoles. El IUMA, junto con el Centro de Electrónica Industrial (CEI) de la Universidad Politécnica de Madrid, son los dos únicos centros universitarios españoles que participan en este macro-proyecto, cuya duración estimada es de 3 años. En paralelo, este proyecto se ha visto complementado con fondos del Ministerio de Economía y Competitividad del Gobierno de España, a través de las acciones de Programación Conjunta Internacional del Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, en su convocatoria 2015-2.

ENABLE-S3 tiene un carácter claramente industrial, y su objetivo prioritario es el desarrollo de sistemas altamente automatizados, autónomos y seguros. En los últimos años se han venido desarrollando distintas iniciativas destinadas a la creación de coches inteligentes sin conductor, que permitan disminuir el número de accidentes y aumentar el grado de confort y la seguridad de sus ocupantes y de los peatones en general. No en vano, la mayoría de los accidentes de tráfico se deben a errores de los conductores, y no a defectos de los propios vehículos. Uno de los principales problemas a la hora de diseñar este tipo de sistemas, es que se necesita más de 1 millón de kilómetros de verificación hasta validar el producto, por lo que resulta deseable contar con sistemas que agilicen este proceso de verificación. Muchos de los avances que ha experimentado la industria del automóvil autónomo pueden ser transferidos a otros dominios industriales objeto de ENABLE-S3: el sector aeroespacial, el ferroviario, el sector hospitalario, el marítimo y el de la agricultura.

La principal aportación del IUMA en este proyecto es la de dotar de tecnología hiperespectral a varias de estas aplicaciones, mejorando de esta forma su funcionalidad y prestaciones. La tecnología hiperespectral [1] surgió hace ya varias décadas en aplicaciones de satélites de observación de la tierra. Se basa en obtener una imagen, de una misma zona espacial, en distintas longitudes de onda del espectro electromagnético. En una imagen convencional se capta solo una pequeña fracción del espectro: la correspondiente al rango del visible. Este rango se ve ampliado con la tecnología hiperespectral a otras longitudes de onda, como puede ser el infrarrojo o el ultravioleta. A partir de estos datos se puede obtener información físico-química del objeto que se está observando, y este es el principal beneficio que nos otorga esta tecnología, no solo la de detectar objetos sino la de identificar su composición. Uno de los mayores retos a los que nos enfrentamos a la hora de trabajar con esta tecnología es que tratamos con imágenes mucho más complejas, y por lo tanto requieren de un procesamiento también más complejo.

El IUMA lleva años desarrollando esta tecnología en sus laboratorios, y en la actualidad está transfiriendo mucho del conocimiento generado hacia diversas aplicaciones reales. Así, en el proyecto europeo HELICoiD, el IUMA aplica la tecnología hiperespectral para facilitar a los neurocirujanos la detección de los bordes de un tumor en el cerebro durante una operación, de forma que resulte más sencillo determinar qué zona debe ser extraída [2][3]. En otro proyecto financiado por la Agencia Espacial Europea (ESA), el IUMA desarrolla un sistema electrónico que permite comprimir los datos de una imagen hiperespectral antes de ser enviados desde una satélite a la tierra, para conseguir de esta forma una transmisión más rápida y eficiente [4][5].

En el proyecto ENABLE-S3 el IUMA está aplicando la tecnología hiperespectral en dos sectores: el espacio y la agricultura de precisión (o agricultura inteligente). En el primer caso, se pretende mejorar los actuales sistemas de compresión de imágenes hiperespectrales basándonos en estándares aprobados por las diversas agencias internacionales del espacio (como NASA en EEUU, CNSA en China, CNES en Francia o DLR en Alemania). Se desarrollarán además sistemas de navegación autónomos para satélites espaciales basados en el procesado de imágenes.

Este Trabajo de Fin de Máster es una aportación al desarrollo del otro caso de uso del proyecto ENABLE-S3 en el que está involucrado el IUMA, el destinado a la agricultura de precisión [6][7]. Esta área de investigación se basa en aprovechar el potencial tecnológico actual para disminuir el esfuerzo del agricultor y aumentar la productividad en los campos agrícolas. Además de las connotaciones tecnológicas derivadas de este tipo de proyecto, hay otras de tipo socio-económico y medio ambiental. En cuanto al primer factor, el socio-económico, hay que indicar que según el Banco Mundial en los últimos 50 años se ha perdido casi la mitad de la tierra cultivable por persona (en el caso de España, hemos pasado de 0,53 hectáreas por persona en 1961 a 0,26 hectáreas en 2014) [8], a lo que se une el aumento de población que en el año 2030 pasará a ser de 8.500 millones de personas. Indudablemente esto afectará en gran medida a toda aquella población que tenga una gran dependencia del sector agrícola, y todo lo que sea mejorar los recursos y aumentar la productividad de sus tierras beneficiará a su economía y a su bienestar. Por otro lado, el uso de recursos hídricos para abastecer a los campos de cultivo, así como el de pesticidas para evitar la llegada de plagas o eliminarlas, hace necesaria una estrategia que permita optimizar su utilización, disminuyendo el impacto negativo que pueda tener sobre el medio ambiente y la salud. Los riesgos que los pesticidas tienen para la salud (sobre todo en niños, adolescentes y mujeres embarazadas), hacen que debamos afrontar este problema de forma inteligente para evitar consecuencias fatales como pueden ser el provocar cáncer o acarrear consecuencias para los sistemas reproductivo, inmunitario o nervioso.

ENABLE-S3 pretende crear una prueba de concepto encaminada a desarrollar una maquinaria agrícola de recogida de cosecha (trigo, uvas, etc) que sea autónoma y automática, de forma que se dirija por sí sola a aquellas zonas de un campo de cultivo en el que se haya detectado previamente que el grado de madurez es óptimo para realizar la recogida, o el grado de humedad está por debajo de un límite y haya que aumentar el riego, o que comienza a producirse un cambio en la calidad de las plantas debido a la llegada de una plaga y por lo tanto haya que echar pesticidas de forma selectiva en determinadas áreas. Para ello es necesario disponer de información (en forma de coordenadas GPS) de aquellas zonas con este tipo de necesidades, transmitir dichas coordenadas a la maquinaria agrícola y hacer que la misma realice su trabajo de forma automática.

Existen varias formas de acometer este reto. Si bien en la actualidad varias empresas afrontan este problema haciendo uso de una red de sensores distribuida por los campos agrícolas, la opción que se sigue en ENABLE-S3 es la de disponer de un único sensor

hiperespectral embarcado en un dron que realice vuelos frecuentes a lo largo de la zona de cultivo. Para realizar la captura de las imágenes hiperspectrales, el IUMA dispone en sus laboratorios de varias cámaras, una de las cuales tiene unas características ideales para ser embarcada en un dron, debido a su pequeño volumen y su bajo peso: la FX10 de Specim (www.specim.fi). Esta cámara dispone de 224 bandas que van desde los 400 nm hasta los 1000 nm (el rango del visible está entre los 400 y los 700 nm), y tiene un peso de solo 1.4 Kg. Conjuntamente con esta cámara se ha adquirido un dron de altas prestaciones, el Matrice 600 de la empresa Dji (www.dji.com), capaz de transportar hasta un máximo de 6 Kg durante un tiempo máximo de 20 minutos. Ambas infraestructuras se muestran en la Figura 1.

Figura 1. DJI Matrice 600 y cámara hiperespectral Specim FX10

A la hora de afrontar las tareas que conlleva la agricultura de precisión, en una primera fase se busca delimitar una zona en la que se quiera hacer el análisis, para posteriormente poner a volar el dron capturando imágenes, las cuales una vez procesadas permitan extraer información relativa a distintas características de la cosecha (madurez, enfermedades, humedad, etc.) y transmitir esa información al agricultor y/o a la maquinaria agrícola. Esta tarea se realizaría a ser posible cada día a primera hora de la mañana, antes de proceder a las tareas agrícolas (de recogida, por ejemplo) necesarias. En este TFM se desarrolla un programa en lenguaje Objective-C encaminado a que el dron realice esta tarea de forma óptima y autónoma, barriendo un área de cultivo y tomando una serie de imágenes que una vez compuestas den una visión general de toda el área analizada. La característica principal es que el código debe tener en cuenta las limitaciones impuestas por el tiempo de la batería, y por lo tanto debe hacer una buena elección de dos parámetros que afectan directamente al tiempo de vuelo, como son la velocidad y la altura a la que debe volar. La altura a su vez afecta a la resolución espacial de la imagen que se desee obtener. Todos estos códigos y procesos son independientes del tipo de sensor que vaya embarcado en el dron, ya que se trata de un TFM encaminado a la optimización del vuelo y no al procesamiento de imágenes, si bien, como se verá, se incluyen algunos aspectos relativos a la captura de imágenes con una cámara RGB.

1.2 Objetivos

Como ya se ha mencionado anteriormente, este TFM se sitúa en la primera fase del proyecto de implementación de tecnología hiperespectral en agricultura inteligente que se lleva a cabo en el IUMA. Esta primera fase se centra en llevar a cabo la primera toma de contacto con los drones que se emplearán en dicho proyecto, buscando una solución eficaz para la automatización y optimización de su vuelo para explorar un campo agrícola mientras se capturan imágenes del mismo.

Durante este TFM se trabajará con un dron Phantom 4 de la empresa Dji, un modelo de gama inferior al Matrice 600 que portará la cámara hiperespectral en las fases futuras. Los objetivos que se busca completar en este trabajo son:

- Desarrollar una aplicación para dispositivos móviles en lenguaje Objective-C, que optimice el vuelo del dron de tal forma que este sea capaz de explorar un área agrícola determinada de forma automática, buscando siempre la solución que consiga la mayor resolución posible. Para ello, se debe determinar la trayectoria a seguir y la configuración de vuelo necesaria.

- Desarrollar una interfaz de usuario sencilla que permita definir el área a explorar por el dron y configurar el proceso del vuelo. El área a explorar podrá ser introducida señalándola en un mapa mostrado en un dispositivo móvil o bien introduciendo sus coordenadas espaciales.

- Tener en cuenta todos los parámetros que afectarán a la resolución y capacidad de vuelo, como la altura, el FOV (Field Of View) y el nivel de batería, limitando el vuelo a una velocidad máxima. Esto implica que se dará prioridad a minimizar la altura a la que se vaya a realizar el vuelo, respetando siempre el tiempo de vuelo que permita el nivel de batería disponible. De esta forma, se obtendrá la máxima resolución espacial posible, a costa de emplear un mayor tiempo de vuelo.

- Automatizar el proceso por medio del cual se obtendrán imágenes con la cámara del dron cada cierto intervalo de tiempo, de forma que a posteriori se podrán componer todas las imágenes en una única imagen de gran resolución espacial. Los parámetros que afectarán al intervalo de tiempo con el cual se toman las imágenes son la altura y velocidad de vuelo del dron, el FOV de la cámara y el porcentaje en el que dos imágenes consecutivas se superponen para facilitar el proceso de *mosaicing*.

- Evaluar, analizar y modificar las herramientas y utilidades de Dji que cumplan con los requisitos del trabajo y puedan ser adaptadas al mismo

- El software desarrollado para el Phantom 4 deberá ser exportable a otros productos de la empresa Dji, especialmente al Matrice 600 al ser la plataforma de vuelo que será utilizada en el proyecto ENABLE-S3, por lo que para su desarrollo se emplearán únicamente las utilidades del SDK compatibles con ambos sistemas.

1.3 Estructura del documento

El trabajo que se expone en este documento se encuentra desglosado en cuatro capítulos, además de las referencias bibliográficas consultadas.

En el presente capítulo, el primero de ellos, se introducen los antecedentes y motivaciones que han llevado a la realización del trabajo y, además, se desarrollan los objetivos del mismo.

El segundo capítulo trata sobre los conocimientos necesarios para el desarrollo del trabajo. Aquí se introducen los recursos de Dji con los que se ha trabajado y el entorno de desarrollo empleado para la generación del software final. También se introduce un software de orto rectificación de imágenes que se empleará para generar la composición de imágenes en los resultados.

En el tercer capítulo se exponen las soluciones adoptadas para la automatización y optimización del vuelo. Se explica la estructura del software desarrollado, detallando el funcionamiento de cada uno de sus módulos, haciendo especial hincapié en el proceso de optimización del vuelo y todas las consideraciones relacionadas con el mismo.

Finalmente, en el cuarto capítulo se recogen los resultados obtenidos y las conclusiones, donde se comprobará el grado de consecución de los objetivos planteados y se proponen algunas líneas de trabajo futuras.

Capítulo 2. Conocimientos previos y herramientas empleadas

Tal y como se mencionó en el capítulo anterior, el objetivo principal de este Trabajo de Fin de Máster es la realización de una aplicación que permita la captura de imágenes en vuelo, optimizando una serie de parámetros. Para ello, ha sido necesario adquirir conocimientos y aprender el manejo de una serie de herramientas que serán detallados en este capítulo.

En primer lugar, se han empleado las herramientas de Dji, que van desde el propio dron hasta un simulador para realizar comprobaciones y test, así como sobre el entorno de desarrollo software donde se ha realizado la programación e implementación del código, el Xcode. También se introduce un software de ortorectificación de imágenes que se empleará para generar la composición de imágenes en los resultados (OpenDroneMap).

2.1 Herramientas de Dji

2.1.1 Phantom 4

El Phantom 4 (Figura 2) es el dron que se ha empleado para el desarrollo de este proyecto. Se trata de un dron cuadricóptero equipado con una cámara RGB de 12 megapíxeles, batería inteligente y sensores de proximidad para evitar colisiones. Este dispositivo puede ser programado mediante el kit de desarrollo de software proporcionado por el fabricante, característica que lo hace óptimo para este proyecto. El control remoto permite manejarlo a distancias de hasta 5 km a la vez que se retransmite en tiempo real la vista desde la cámara en HD, todo ello gracias al enlace de baja latencia por el que están conectados (DJI Lightbridge).

Figura 2. Phantom 4 de Dji

Su batería inteligente de 5350 mAh gestiona el consumo de energía de tal forma que se pueden llegar a registrar tiempos de vuelo de hasta 28 min en condiciones óptimas. El controlador de vuelo que integra, permite localizar el dron con coordenada GPS gracias a su uso de los satélites de GPS y GLONASS disponibles. También es el responsable de procesar la información de los sensores y gestionar el mecanismo de detección y evasión de obstáculos.

En cuanto a la cámara, es capaz de capturar imágenes de 12 megapíxeles y grabar video con resolución 4K a 120 fotogramas por segundo. Está unida al dron a través de un gimbal, que se muestra con más detalle en la Figura 3, capaz de realizar giros en los 3 ejes para mantener la cámara estable y apuntando en la misma dirección a pesar de las distintas vibraciones que sufre el dron.

El sensor es de 1/23" y 12 megapíxeles eficaces, siendo el tamaño exacto de la imagen de 4000x3000 píxeles y pudiéndose elegir entre formatos JPEG y RAW. Estos datos junto con las características de la lente, de 20 mm (35 mm formato equivalente) y con un campo de visión (FOV) diagonal de 94º, permitirán conocer la resolución espacial de las imágenes en función de la altura a la que se encuentre la cámara.

Figura 3. Cámara y gimbal del Phantom 4

Existe una app específica del fabricante que permite conectarse al dron para visualizar las imágenes capturadas por la cámara y controlar todos los parámetros relacionados con la configuración de la cámara y el control de vuelo, denominada DJIGo.

En la Tabla 1 se encuentran agrupadas las principales características del dron y sus componentes.

Tabla 1. Características del Phantom 4

Specifications	
Aircraft	
Weight (Battery & Propellers Included)	1380 g
Diagonal Size (Propellers Excluded)	350 mm
Max Ascent/Descent/Horizontal Speed	6 m/s ; 4 m/s ; 20 m/s (Sport Mode)
Max Service Ceiling Above Sea Level	6000 m
Max Flight Time	Approx. 28 minutes
Operating Temperature Range	32° to 104°F (0° to 40°C)
Satellite Positioning Systems	GPS/GLONASS

Gimbal	
Stabilization	3-axis (pitch, roll, yaw)
Controllable Range	Pitch: -90° to +30°
Vision System	
Vision System	Forward Vision System Downward Vision System
Velocity Range	≤10 m/s (2 m above ground)
Altitude Range	0 - 33 feet (0 - 10 m)
Operating Range	0 - 33 feet (0 - 10 m)
Obstacle Sensory Range	2 - 49 feet (0.7 - 15 m)
Operating Environment	Surface with clear pattern and adequate lighting (lux>15)
Camera	
Sensor	1/2.3” CMOS Effective pixels:12.4 M
Lens	FOV 94° 20 mm (35 mm format equivalent) f/2.8 focus at infinite.
ISO Range	100-3200 (video), 100-1600 (photo)
Electronic Shutter Speed	8 - 1/8000 s
Image Size	4000×3000
Still Photography Modes	Single shot Burst shooting: 3/5/7 frames Auto Exposure Bracketing (AEB): 3/5 bracketed frames at 0.7 EV Bias Timelapse HDR
Video Recording Modes	UHD: 4096×2160 (4K) 24 / 25p 3840×2160 (4K) 24 / 25 / 30p 2704×1520 (2.7K) 24 / 25 / 30p FHD: 1920×1080 24 / 25 / 30 / 48 / 50 / 60 / 120p HD: 1280×720 24 / 25 / 30 / 48 / 50 / 60p
Picture Formats	JPEG, DNG (RAW)
Video	MP4, MOV (MPEG-4 AVC/H.264)
Supported SD Cards	Micro SD Max capacity: 64 GB Class 10 or UHS-1 rating required
Operating Temperature Range	32° to 104°F (0° to 40°C)
Remote Controller	
Operating Frequency	2.400 - 2.483 GHz
Max Transmission Distance	FCC Compliant: 3.1 mi (5 km) CE Compliant: 2.2 mi (3.5 km) (Unobstructed, free of interference)
Operating Temperature Range	32° to 104°F (0° to 40°C)
Battery	6000 mAh LiPo 2S
Transmitter Power (EIRP)	FCC: 23 dBm CE: 17 dBm
Operating Current/Voltage	1.2 A@7.4 V
Video Output Port	USB
Mobile Device Holder	Tablets and smart phones

Intelligent Flight Battery	
Capacity	5350 mAh
Voltage	15.2 V
Battery Type	LiPo 4S
Energy	81.3 Wh
Net Weight	462 g
Charging Temperature Range	41° to 104°F (5° to 40°C)
Max Charging Power	100 W

2.1.2 DJI Assistant y simulador

Para poder realizar una conexión con el dron y realizar un mantenimiento de su estado y versiones del firmware, Dji facilita un programa denominado DJI Assistant, disponible para sistemas operativos Windows y macOS. A continuación, se muestran las principales utilidades de este software, siendo las más importantes en este caso las actualizaciones del firmware y el simulador de vuelo.

Al iniciarse el programa, el software reconoce los dispositivos de Dji que se encuentren conectados al equipo y nos permite elegir uno de ellos (Figura 4). En el caso del Phantom 4, lo primero que nos facilita el programa es un listado con todos los registros de vuelo; cada vez que el dron se pone en funcionamiento, guarda un registro con la información de los vuelos que se sucedan, con información como el tiempo de vuelo o la fecha y hora a la que sucedió.

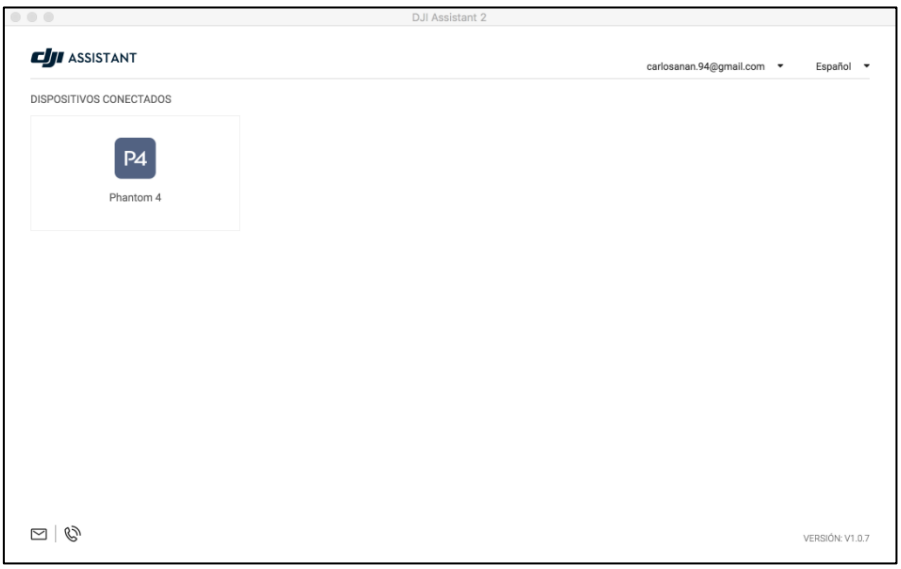


Figura 4. Pantalla de inicio de DJIAssistant

El siguiente punto a tener en cuenta es el apartado del firmware. El programa detecta automáticamente la versión del firmware instalada y comprueba la existencia de alguna versión más reciente del mismo. Al funcionar con baterías inteligentes, estas también necesitan comprobar la existencia de actualizaciones para su firmware, siendo el software el encargado de descargar e instalar las últimas versiones disponibles.

Existe también una parte del programa dedicada a la calibración del dron. En el dron se pueden calibrar dos elementos: los sensores de detección de obstáculos y la brújula (para la localización). En esta parte del programa se calibran solo los sensores, ya que para calibrar la brújula es necesario hacer uso de la aplicación DJIGo.

Por último, llegamos a la parte más importante de DJI Assistant, el simulador. Con esta herramienta es posible manejar el dron con el mando o cargar y ejecutar un programa y observar el comportamiento y la respuesta del dron a partir de un modelo del mismo que ha sido creado por los desarrolladores de Dji y que se nos muestra en la pantalla. De esta forma se pueden realizar distintas pruebas y comprobaciones para validar el funcionamiento de las acciones que se programen, siendo posible detectar la existencia de errores sin necesidad de desplegar el dron y evitando el riesgo que conlleva realizar pruebas de campo sin ninguna certeza de la validez del trabajo realizado.

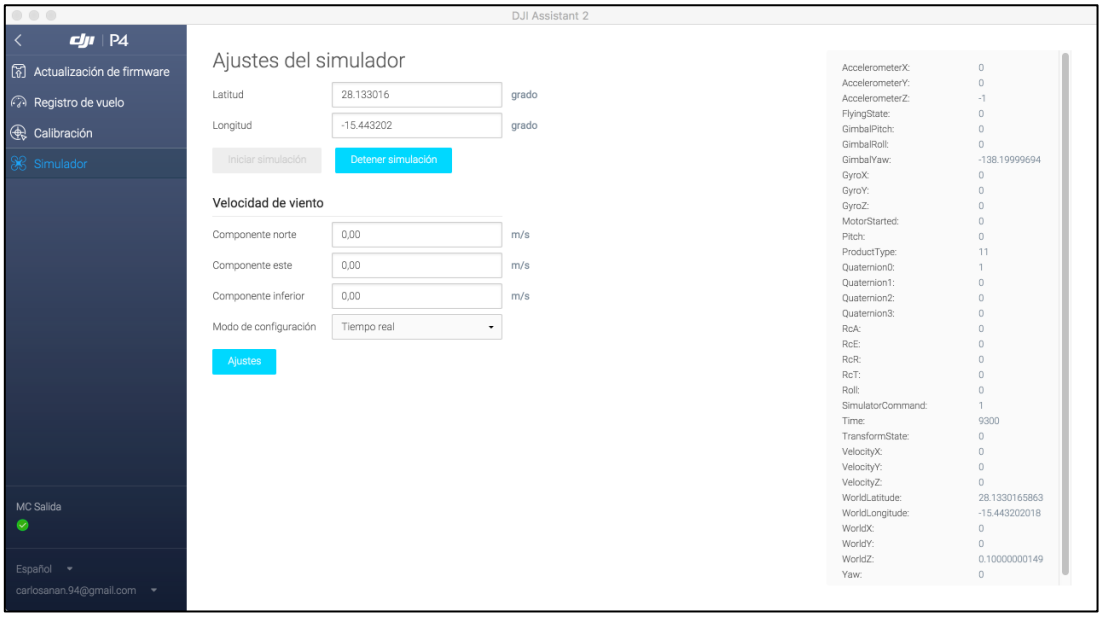


Figura 5. Configuración del simulador

En la Figura 5 se muestra la ventana de configuración inicial del simulador. Es necesario rellenar los campos de latitud y longitud del punto de inicio, ya que es en este punto en el que se situará virtualmente el dron y desde el que se iniciará el vuelo. También existe la posibilidad de simular condiciones meteorológicas adversas, siendo posible definir una corriente de viento con una dirección y velocidad determinadas. A la derecha de la imagen puede apreciarse una sección que contiene varias etiquetas y variables relacionadas con el estado del dron, encontrándose entre ellas la velocidad en cada uno de los ejes, la posición con respecto al punto de partida, el tiempo de simulación, etc. Al iniciarse la simulación, todos estos valores se van actualizando de forma constante, permitiendo así monitorizar el comportamiento del dron y detectar errores o anomalías que no correspondan con las acciones programadas.

Al iniciar la simulación se abre una nueva ventana, como se puede observar en la Figura 6, donde se muestra la interfaz gráfica del simulador. El modelo del dron que incluye se mueve tal y como lo haría el dron en la realidad, actualizando su posición a la vez que se actualizan los datos. Esta ventana también incluye un listado más reducido de parámetros que permiten corroborar los datos de posición y velocidad de forma rápida y directa.

Es aquí donde mejor se visualiza el resultado de la simulación, ya que lo que se observa son directamente los movimientos que se obtienen en todo momento. También es posible configurar el simulador para que dibuje la trayectoria que se ha seguido durante el recorrido, una opción bastante útil a la hora de comparar el resultado simulado con el esperado y detectar posibles errores.

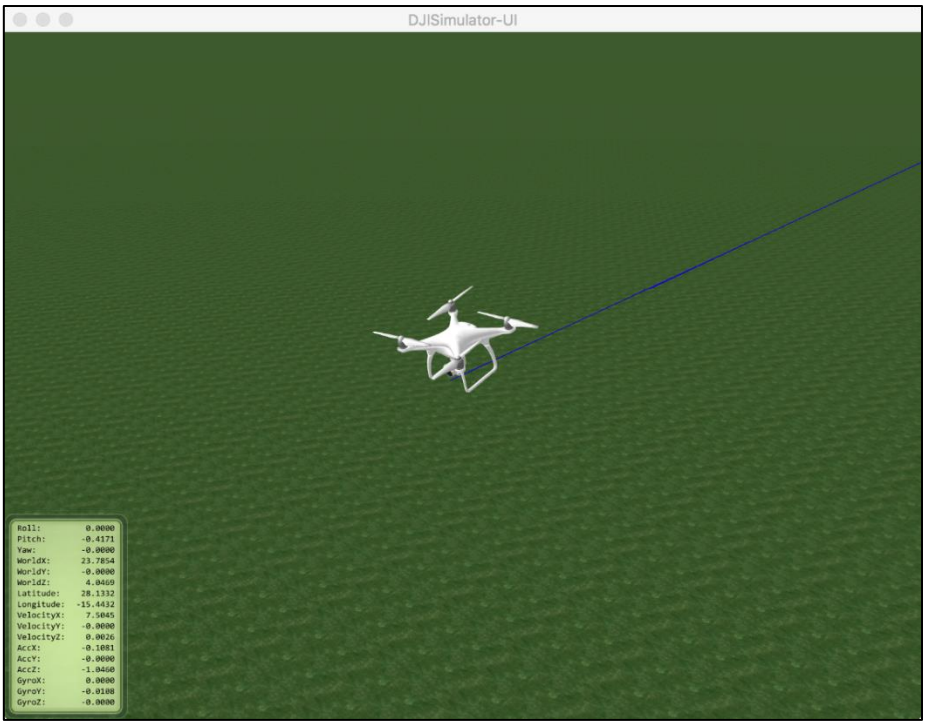


Figura 6. Interfaz del simulador en funcionamiento

2.1.3 Esquemas de funcionamiento y conexiones

Para poder entender cómo se ha realizado el programa y el por qué se emplea un lenguaje de programación y un entorno de desarrollo determinados, es necesario conocer qué elementos son necesarios para programar el dron y cómo se conectan entre ellos; de la misma forma, es fundamental conocer qué elementos nos permiten realizar las simulaciones del programa y qué conexiones deben realizarse.

Como ya se ha explicado anteriormente, el dron se maneja a través de un control remoto propio que cuenta con un soporte y conexiones USB para portar y conectarse con un dispositivo móvil externo, que permitirá visualizar en tiempo real la información captada por la cámara y configurar múltiples parámetros del dron a través de DJIGo. Es en este dispositivo móvil, Smartphone o Tablet, donde se carga y ejecuta toda la programación, siendo el control remoto el que actúa de intermediario entre el dron y el dispositivo móvil. En la Figura 7 se muestra un esquema con todos los elementos implicados en este proceso y sus conexiones.

A la hora de trabajar con el simulador, las conexiones necesarias son las mismas que se acaban de mencionar, mostradas en la Figura 7, teniendo la aplicación cargada en el dispositivo móvil y transmitiéndose a través del control remoto, con la única diferencia de que, en este caso, el dron también debe estar conectado vía USB al equipo donde se esté ejecutando el simulador.

Figura 7.Esquema de conexiones del Phantom 4

2.1.4 Software Development Kit (SDK)

Ya se ha aclarado que el Phantom 4 está preparado para funcionar con programas desarrollados de forma independiente y personalizada; de hecho, es precisamente esta característica sobre la que se fundamenta y desarrolla este trabajo. Para poder llevar a cabo esta labor, Dji cuenta con varios kits de desarrollo que pueden ser utilizados por cualquier programador. Estos kits son:

- **Mobile SDK**, que permite crear una aplicación para dispositivos móviles.
- **Onboard SDK**, más complejo y orientado a desarrollar el control del dispositivo desde un sistema dedicado e integrado en el propio dron.
- **Guidance SDK**, enfocado en el desarrollo de aplicaciones basadas directamente en la visión.

En este trabajo solo haremos uso del Mobile SDK, porque es el que se ajusta a las necesidades del proyecto y además es compatible con varios dispositivos (como el Matrice 600 o el Phantom 4). Para poder trabajar con el Mobile SDK, en la web de Dji Developer [9] se encuentra la documentación necesaria para entender su funcionamiento y se facilitan algunos ejemplos de su implementación, todo ello disponible tanto para Android como para iOS. Cabe destacar que para poder hacer uso del SDK es necesario crearse una cuenta de usuario de Dji, que es totalmente gratuita y complementa al propio SDK.

El SDK permite a los desarrolladores acceder y controlar distintas características y capacidades de los productos de Dji, encargándose de controlar de forma independiente todas las funciones de bajo nivel como la estabilización del vuelo, la gestión de la batería o las comunicaciones. De este modo, la tarea del desarrollador se centra en automatizar el vuelo y

tomar el control de la cámara y el gimbal, así como de todos los archivos multimedia que se generen.

Existen tres formas de controlar el vuelo empleando el SDK. El primero es de forma manual, manejando el dispositivo con el control remoto mientras el SDK simplemente se encarga de habilitar la visualización del vídeo y de adquirir los datos de los sensores. El siguiente se basa en el uso de sticks virtuales aprovechando los comandos que permiten simular los controles del control remoto. El último método, en el que se basará el programa desarrollado, es mediante el uso de acciones de alto nivel denominadas misiones. Una misión es un conjunto de acciones sencillas que permiten realizar una tarea más compleja.

Antes de pasar a explicar las misiones existentes y cuál de ellas ha servido de base para el programa desarrollado, se va explicar la arquitectura del SDK y cómo acceden a él las aplicaciones. En la Figura 8 se muestran las clases principales del SDK de las que hace uso una aplicación.

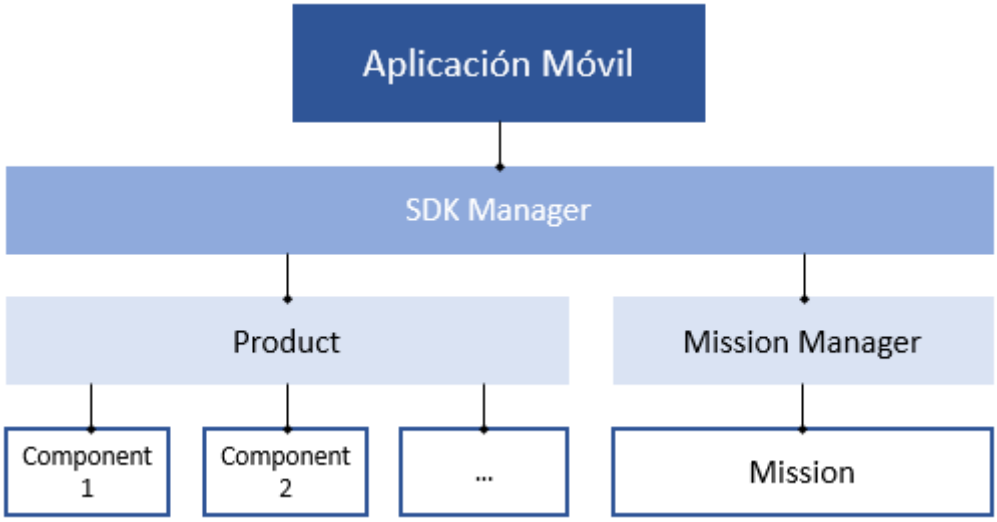


Figura 8. Clases principales del SDK

SDK Manager es la clase que se encarga de gestionar todo el proceso de registro del SDK y la conexión del producto con la aplicación. Para que una aplicación pueda hacer uso del SDK, primero debe obtener permiso para ello, permiso que solo puede adquirirse durante el registro si se cuenta con una clave de acceso. Esta clave de acceso se genera a través de la web de Dji Developer, en el espacio personal de la cuenta de usuario de Dji, y es única para cada aplicación. La clase provee además una instancia del producto una vez se haya realizado la conexión, permitiendo tomar el control del mismo. Por último, el SDK Manager también da la posibilidad de añadir un modo de depuración haciendo uso de una aplicación especial, que se describirá con más detalle a lo largo de este documento.

La clase Product, por su parte, contiene todas las propiedades básicas del producto conectado, que puede ser un dron como el Phantom 4 o un dispositivo de mano como un estabilizador para cámaras, así como los componentes de dicho producto. Estos componentes son las distintas clases que describen elementos como el gimbal, el flight controller o la cámara y que hacen posible tomar el control de cada uno de ellos y conocer su estado.

En la parte relativa a las misiones, en primer lugar se encuentra la clase *Mission Mananager*, que se emplea para tomar el control de las misiones disponibles, gestionando la preparación, ejecución, finalización o pausa de cada una de ellas, además de dar acceso al estado

actual de la misión en ejecución. La clase *Mission*, por su parte, describe cada una de las misiones disponibles y se encarga de mantener los valores de todos aquellos parámetros que la definen

Para entender un poco más en qué consisten las misiones, a continuación se resume brevemente cada una de las que están disponibles, indicando qué acciones permite implementar cada una de ellas.

- **Waypoit Mission**: Esta es la misión principal que se ha usado para automatizar el vuelo del dron. El funcionamiento de esta misión se basa en definir una serie de puntos en el mapa de forma ordenada, puntos hacia los que el dron volará de uno en uno, a una determinada altura que puede ser independiente para cada punto. La misión tiene un máximo de 99 puntos, y para cada uno de ellos da la posibilidad de realizar una serie de acciones, como capturar una imagen.
- **Hot Point Mission**: tras definir un único punto denominado como "Hot Point", el dron volará haciendo círculos de radio constante a una altura determinada, siendo posible definir la altura, velocidad y el punto de "Hot Point". Otros parámetros que pueden definirse son la orientación o la dirección del vuelo.
- **Follow Me Mission**: en una misión de *Follow Me*, el dron seguirá una serie de coordenadas GPS que se le enviarán cada cierto tiempo, manteniendo una determinada separación y una altura constante. Si no recibe una coordenada durante más de 6 segundos, el dron se mantendrá estático en su sitio. En la Figura 9 pueden observarse los conceptos de cada una de estas misiones.

Figura 9. Waypoint Mission (centro), Hot Point Mission (derecha), Follow me Mission (izquierda)

- **Active Track Mission:** permite que el dron siga a un objetivo sin utilizar ningún tipo de posicionamiento GPS. Para ello, se define un rectángulo en la pantalla que contenga el elemento a seguir; el dron procesa estas imágenes e identifica el elemento a seguir, pidiendo una confirmación por pantalla. Una vez confirmado dicho

elemento, el dron se encargará de procesar las imágenes que obtiene para seguir su trayectoria.

- **TapFly Mission:** en esta misión se tiene el dron con la cámara apuntando hacia el lugar de interés y en la pantalla se toca con el dedo sobre un punto que se encuentre dentro del rango de visión de la cámara. Una vez hecho esto, el dron calcula la dirección del punto que se ha tocado y vuela hacia el con la detección de obstáculos activada. En la Figura 10 se muestra el concepto de funcionamiento de estas dos misiones.
- **Custom Mission:** se trata de una misión que ejecuta de forma serializada una serie de misiones y pasos de misión de forma encadenada. No puede ejecutar misiones del tipo TapFly o Active Track, pero sí que permite encadenar el resto de misiones con pasos sencillos como despegar, dirigirse a un waypoint, capturar una imagen o un vídeo, ejecutar un Follow Me…

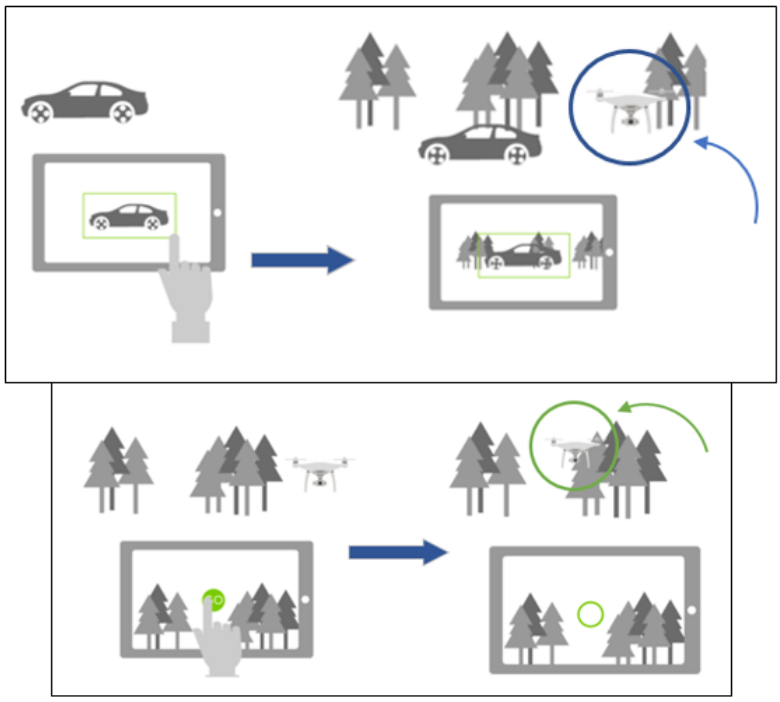


Figura 10. Active Track (superior) y TapFly (inferior) Mission

2.1.5 La aplicación base: GSDemo

Dentro de toda la documentación disponible del Mobile SDK, se encuentra una aplicación de ejemplo que se basa en una vista de mapa y la implementación de *Waypoint Mission*. Este código de ejemplo, está creado con el nombre de **GSDemo**, y ha servido de base para empezar a desarrollar el código e ir añadiendo y modificando las funciones y utilidades necesarias. Para poder distinguir las modificaciones y añadidos de las partes originales de esta aplicación de ejemplo, se detalla a continuación el funcionamiento del programa inicial y las partes que lo componen, explicando cómo se maneja la aplicación y qué resultados se obtienen.

En la Figura 11 se muestra un diagrama con el funcionamiento del programa. Al cargarse la aplicación, lo primero que hace es registrarla con SDK Manager y obtener permiso para hacer uso del SDK. Si no ha habido ningún error, el programa mostrará la vista de la Figura 12, donde se podrá manejar el mapa y añadir Waypoints (Figura 13).

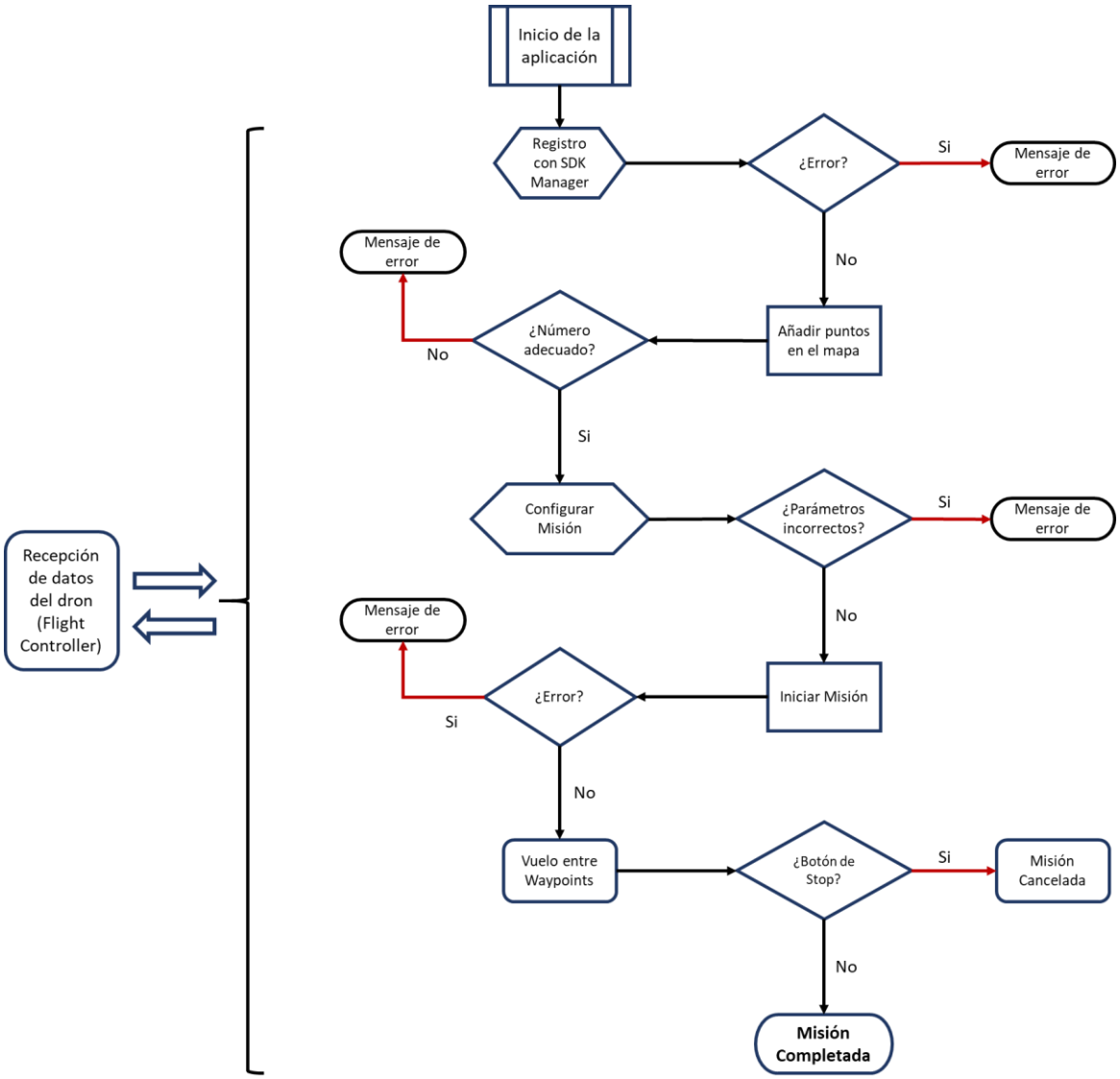


Figura 11. Diagrama de funcionamiento de GSDemo

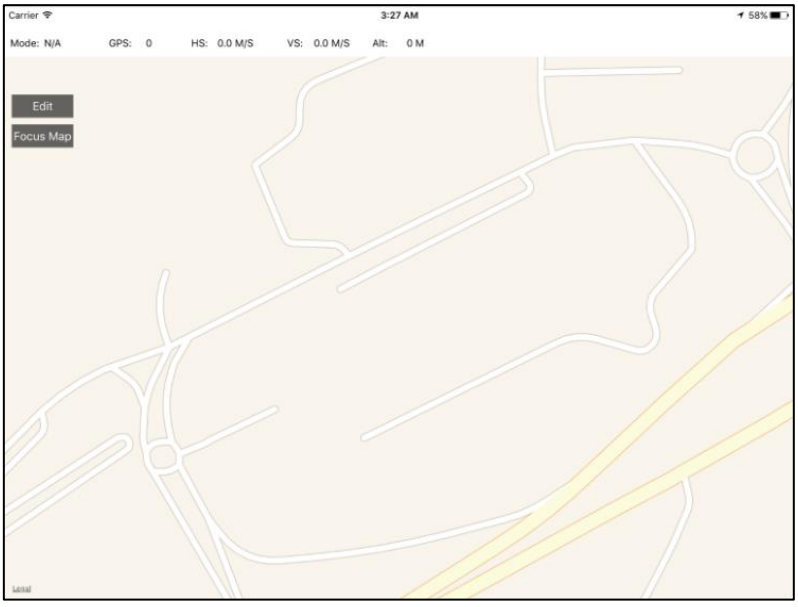


Figura 12. GSDemo: Pantalla del programa

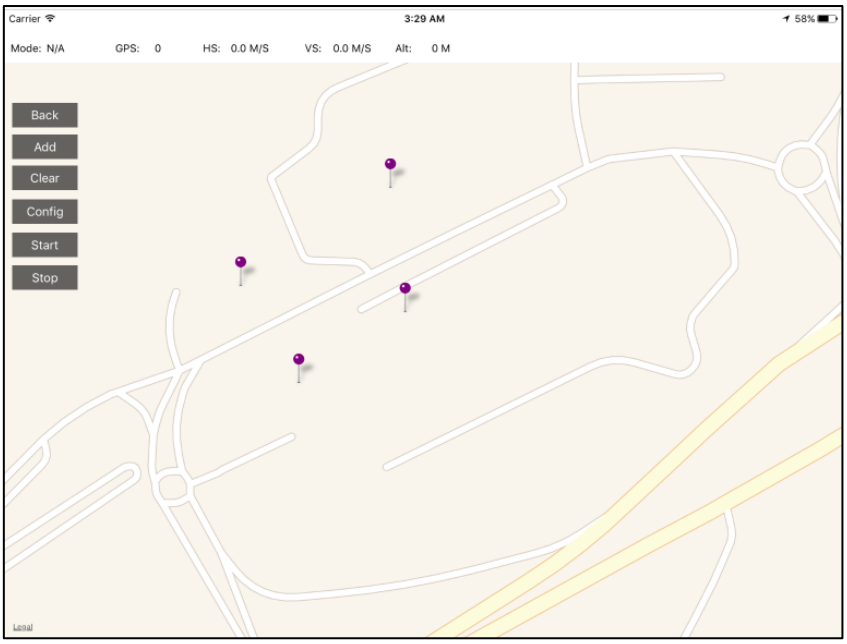


Figura 13. GSDemo: Introducción de Waypoints

Para añadir Waypoints, basta con situar el mapa en la zona deseada, habilitar la introducción de puntos con el botón "Add" y tocar en las zonas del mapa en las que se desea fijar un waypoint. A medida que se van añadiendo puntos, la aplicación los va almacenando en código dentro de un array para su posterior uso.

Una vez añadidos los puntos que marcan el recorrido, se podrá pasar a la configuración de los parámetros de la misión (Figura 14), siempre y cuando el número de puntos añadidos sea apropiado. Aquí puede fijarse la altura de los waypoints, la velocidad máxima admitida y la velocidad automática del vuelo, además de especificar qué acción se debe llevar a cabo una vez finalizada la misión. También se puede elegir qué orientación debe tener el dron durante la misión. Al finalizar la configuración, la aplicación interpreta los datos introducidos y prepara la misión con estos datos, procesando el conjunto de puntos introducidos y adaptándolos a la misión.

A lo largo de todo este proceso, desde que se realiza la conexión entre la aplicación y el dron, se lleva a cabo un intercambio constante y periódico de información entre ambos. Este intercambio se realiza para recibir los parámetros que se muestran en la parte superior de la pantalla, permitiendo conocer en todo momento la altura y velocidad del dron, así como el modo de vuelo actual y el número de satélites de GPS visibles.

El código de la aplicación está formado por un conjunto de clases, compuestas por archivos de cabecera y de implementación, que se reparten las tareas a realizar. Por ejemplo, la aplicación incluye un icono con forma de aeronave para representar la posición del dron en el mapa, y la gestión de la posición y orientación de ese icono se llevan a cabo en dos de las clases mencionadas anteriormente.

Aunque la mayoría de estos archivos sufrirán modificaciones durante el desarrollo de la aplicación, existen algunas excepciones, como las clases que controlan el icono del dron mencionadas anteriormente, en las que los archivos se mantendrán tal y como están, al no ser necesaria ninguna modificación adicional.

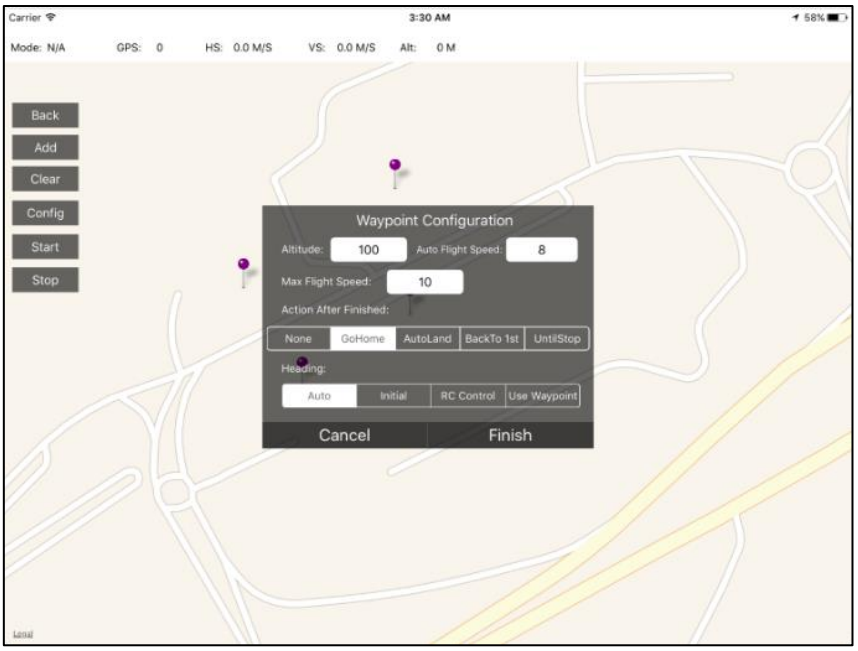


Figura 14. GSDemo: Configuración de la misión

Finalmente, habiendo introducido una configuración válida para estos parámetros, se podrá iniciar la misión al pulsar el botón de *Start*. El dron volará hacia el primer punto a la altura que se le haya especificado, y de este seguirá hacia los demás respetando el orden en el que fueron añadidos. Durante la ejecución de la misión, puede pulsarse el botón de *Stop* con el fin de detener la aplicación de inmediato y tomar los mandos del dron en caso de que fuera necesario. Si el dron llega hasta el último punto sin ningún inconveniente, la misión habrá finalizado y se ejecutará la acción especificada previamente.

2.2 Herramientas de desarrollo del software

2.2.1 Xcode

Xcode [10] es el entorno de desarrollo integrado (IDE) proporcionado por Apple para el desarrollo de software en sus plataformas. Este entorno funciona sobre macOS, y contiene las herramientas necesarias para desarrollar software para macOS, iOs, watchOS y tvOS. La versión que se ha empleado a lo largo de este trabajo es la número 8. Xcode es capaz de compilar código en varios lenguajes, pero solo dos de ellos son dos los que permiten desarrollar programas y aplicaciones nativas para los distintos dispositivos de Apple: Objective-C y Swift (aunque es posible integrar código en algunos otros lenguajes). El resto de lenguajes que es capaz de compilar Xcode lo forman C, C++, Java y AppleScript.

Para las aplicaciones de iOS, se emplea un framework llamado “Cocoa Touch”, que incluye el reconocimiento gestual de acciones sobre la pantalla, una librería para la interfaz de usuario adaptada a los dispositivos móviles y diferentes animaciones orientadas a los mismos.

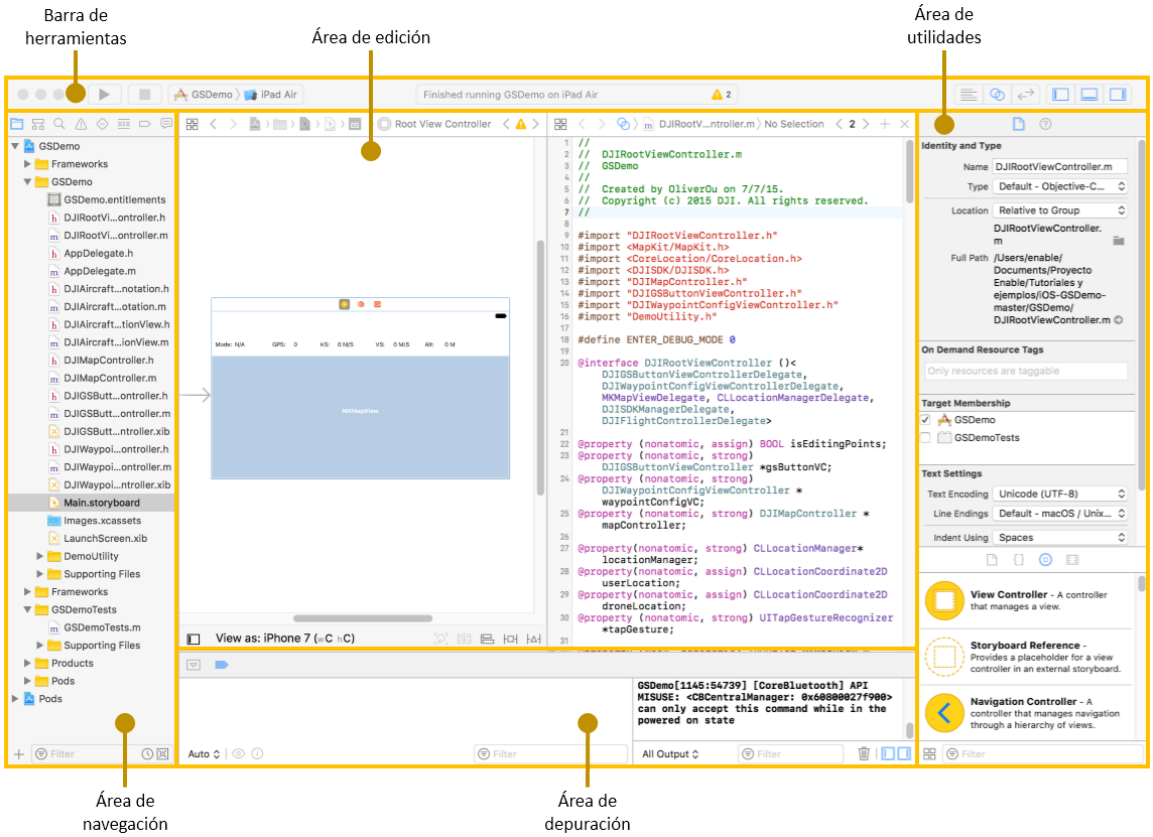


Figura 15. Interfaz de Xcode y áreas de trabajo

En la Figura 15 se muestra el aspecto que presenta Xcode cuando se está trabajando en el desarrollo de un proyecto, en este caso para iOS. Como se puede ver, el programa cuenta con varias áreas de trabajo bien diferenciadas entre las que se reparten las distintas utilidades y herramientas que ofrece el entorno.

- **Barra de herramientas**: situada en la parte superior de la ventana, contiene algunos controles que permiten ejecutar la aplicación, organizar la posición de las principales áreas de trabajo y mostrar el estado en el que se encuentra la aplicación.
- **Área de navegación:** como su propio nombre indica, es el área que permite explorar los contenidos del programa, cambiando el tipo de navegación según la pestaña seleccionada. Algunos de estos contenidos son los archivos del proyecto, la jerarquía de clases, un navegador de búsqueda, el listado de advertencias y errores o el navegador de depuración.
- **Área de edición**: es la parte central del programa. En ella se edita el código fuente de la aplicación, en ese caso en Objective-C, y se desarrolla y diseña el entorno gráfico de la interfaz de la aplicación.
- **Área de utilidades**: es la zona que agrupa herramientas como un inspector de ayuda rápida, gestión de los meta-datos del fichero o la configuración de atributos de los objetos de la interfaz. Contiene además un selector de librerías, que permite añadir objetos y trozos de código al programa.
- **Área de depuración**: formada por el visor de variables y la consola. Permite controlar la depuración del código, mostrando información de la ejecución en la consola y el estado de las distintas variables empleadas en el código.

2.2.2 Objective-C

Como ya se ha mencionado, toda la programación de esta aplicación se ha desarrollado en Objective-C. Objective-C es un lenguaje de programación orientado a objetos, empleado por Apple como lenguaje principal para sus sistemas operativos y el desarrollo de aplicaciones para los mismos, coexistiendo desde 2014 con Swift. Fue desarrollado a principios de los 80 y popularizado por la compañía NeXT al usarlo en su sistema operativo NeXTSTEP, para finalmente ser usado por Apple tras adquirir NeXT.

Este lenguaje se desarrolló tomando como referencias C y Smalltalk. Consiste en una capa situada por encima de C, y permite compilar un programa en C desde cualquier compilador propio al tratarse de un superconjunto de C, además de admitir código en C dentro de un objeto de Objective-C por este mismo motivo. De SmallTalk, por su parte, adopta la sintaxis relativa a la programación orientada a objetos, siendo el resultado final similar a la mensajería de SmallTalk.

A la hora de trabajar con Objective-C, el código de una misma clase de cocoa touch se divide en dos archivos, un archivo de cabecera con extensión .h y un archivo de implementación con extensión .m. En los archivos de cabecera se declaran los distintos métodos que se van a emplear, y en el archivo de implementación es donde realmente se escribe el código de cada método. Para poder emplear el código de una clase en otra, bastará con añadir la cabecera de la clase que se quiere emplear al inicio del archivo mediante un #import.

Por último, para facilitar la comprensión de este documento, hay que aclarar que en Objective-C las funciones se denominan “métodos”. Por lo tanto, cuando se hable de métodos, se estará haciendo referencia a las funciones que se emplean en el código del programa.

2.2.3 Depuración de aplicaciones: BridgeApp

Cuando se pretende aplicar la depuración incluida en Xcode a aplicaciones de Dji, surge un problema de conexiones que dificulta el proceso. Xcode es capaz de depurar código para iOS ejecutándose de dos formas: o bien en el simulador integrado por el entorno, o en un dispositivo con iOS conectado al equipo con Xcode. Si recordamos el esquema de funcionamiento y conexiones del Phantom 4 mostrado anteriormente, el dispositivo que ejecute el código necesita estar conectado al control remoto para poder intercambiar información con el dron.

El problema surge, por lo tanto, debido a la necesidad de conectar el dispositivo con iOS al control remoto y al equipo a la vez, ya que de otro modo las secciones de código que requieren conexión directa con el dron no pueden ser depuradas. La solución a este problema la proporciona Dji mediante una aplicación intermedia llamada BridgeApp, siguiendo el esquema de conexiones de la Figura 16.

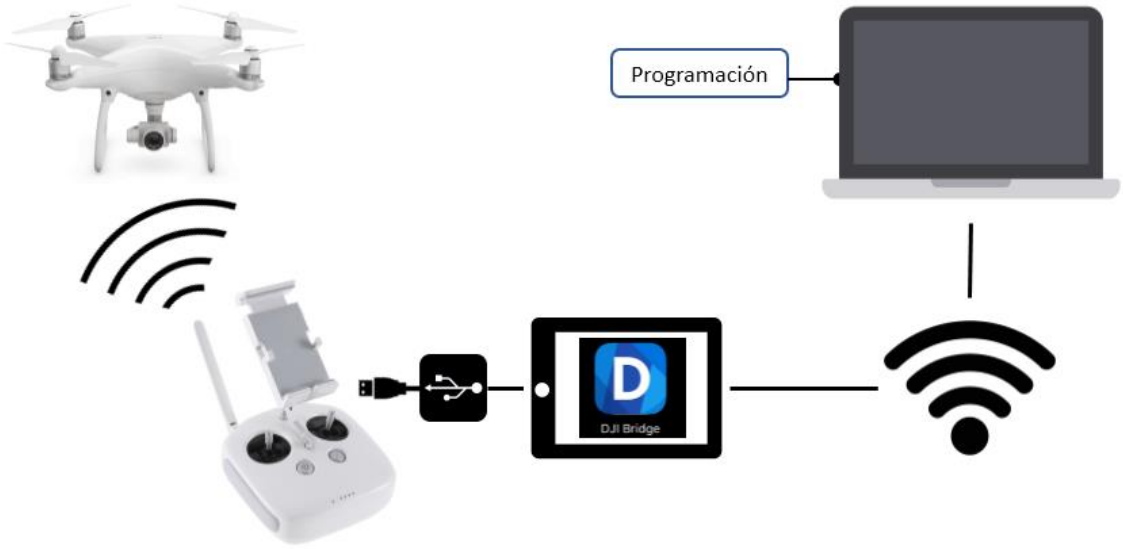


Figura 16. Esquema de conexiones con BridgeApp

El funcionamiento es el siguiente: la aplicación a depurar se ejecuta en el equipo con el simulador de Xcode, mientras que en el dispositivo conectado al control remoto se tiene instalada la BridgeApp. Mediante una serie de métodos del SDK, se establece una conexión entre ambas aplicaciones a través de internet, gracias a un identificador dado por la aplicación intermedia. De este modo, esta última aplicación actúa como puente entre el dron y la aplicación desarrollada, transmitiendo la información de un extremo a otro como si de una conexión normal se tratase, pero permitiendo depurar el código a la vez que se mantiene esta conexión.

2.3 Software de orto rectificación: OpenDroneMap

Cuando el dron finalice el barrido del campo agrícola, el resultado será un conjunto de imágenes del mismo capturadas de forma consecutiva y con un determinado nivel de superposición entre sí. Para poder analizar el terreno, es necesario componer todas las imágenes en una única imagen, para lo que será necesario algún tipo de algoritmo o software de composición.

Aunque no es el objetivo de este TFM, se hace uso de un software de orto rectificación, denominado OpenDroneMap [11] para realizar la composición de algunas imágenes y validar los resultados obtenidos.

OpenDroneMap es una herramienta de posprocesado para imágenes aéreas de drones, que extrae datos geográficos de las mismas como ortofotos, nubes de puntos y mallados de texturas. Está formada por una serie de ficheros de código con los algoritmos necesarios para estas tareas, escritos en su mayoría en Python, que se ejecutan en el terminal sobre un sistema operativo de Linux.

Capítulo 3. Desarrollo de la aplicación

A lo largo de este tercer capítulo se describirán en detalle las soluciones que se han adoptado para desarrollar la aplicación y cumplir los objetivos. En las páginas siguientes se describe la estructura y funcionamiento de la aplicación desarrollada, detallando las distintas utilidades desarrolladas y las principales clases del código.

Posteriormente, se explica con más profundidad el proceso de optimización del vuelo que se ha diseñado para esta aplicación, indicando qué datos se necesitan de entrada, qué resultados se obtienen a la salida y cómo se trabaja con ellos para configurar el vuelo del dron.

Para finalizar, se añade una guía de uso de la aplicación, donde se explican las distintas partes de la interfaz de usuario, la utilidad de cada una de ellas y los pasos a seguir para ejecutar la aplicación de forma adecuada.

3.1 Estructura general de la aplicación

A la hora de empezar con el desarrollo del software, la primera decisión tomada fue emplear la aplicación de ejemplo *GSDemo* como base sobre la que iniciar el desarrollo, y no partir desde cero. Uno de los motivos para tomar esta decisión fue que este ejemplo tenía como objetivo demostrar cómo utilizar la misión "WaypointMission", misión que se adapta a las necesidades del trabajo y que se ha explicado brevemente en páginas anteriores. La principal ventaja que proporciona esta aplicación de ejemplo es que no solo incluye el código básico necesario para llevar a cabo la misión, sino que además proporciona una interfaz de usuario preparada para ello. El hecho de partir de una aplicación como base supone un ahorro de tiempo y recursos que pueden dedicarse al desarrollo de las utilidades necesarias para la automatización y optimización del vuelo del dron.

En la siguiente figura (Figura 17) se muestra un esquema general del funcionamiento de la aplicación desarrollada. Como se puede comprobar si recordamos el esquema general del ejemplo *GSDemo* (Figura 11), las estructuras de ambas aplicaciones presentan ciertas similitudes, principalmente en su estructura. Esta similitud resulta obvia no solo porque esta sea una versión ampliada y adaptada de *GSDemo*, sino que, como ya se ha explicado, los pasos que requiere la misión para ejecutarse serán siempre los mismos, cambiando entre una aplicación y otra las acciones que se llevan a cabo en cada uno de estos pasos.

De la misma forma que se observa la similitud entre ambas, también pueden verse las diferencias existentes. De forma general, las diferencias principales son:

- El número de puntos que se deben añadir pasa de ser libre a tener un valor fijo de 4 puntos, los necesarios para definir el área a analizar.

- Los parámetros de la misión ya no se configuran directamente, sino que se ejecuta la optimización del vuelo para determinar de forma automática el recorrido y las condiciones de vuelo para el área especificada.
- Durante el barrido del área se capturan imágenes del terreno de forma ordenada, manteniendo el nivel de solape entre las mismas.
- Se añaden métodos que permiten comunicación paralela con otras características del dron, añadiendo la lectura del nivel de batería y el control del estado de la misión. El código relativo a este último método puede observarse en el apartado 3.6 de este mismo capítulo.

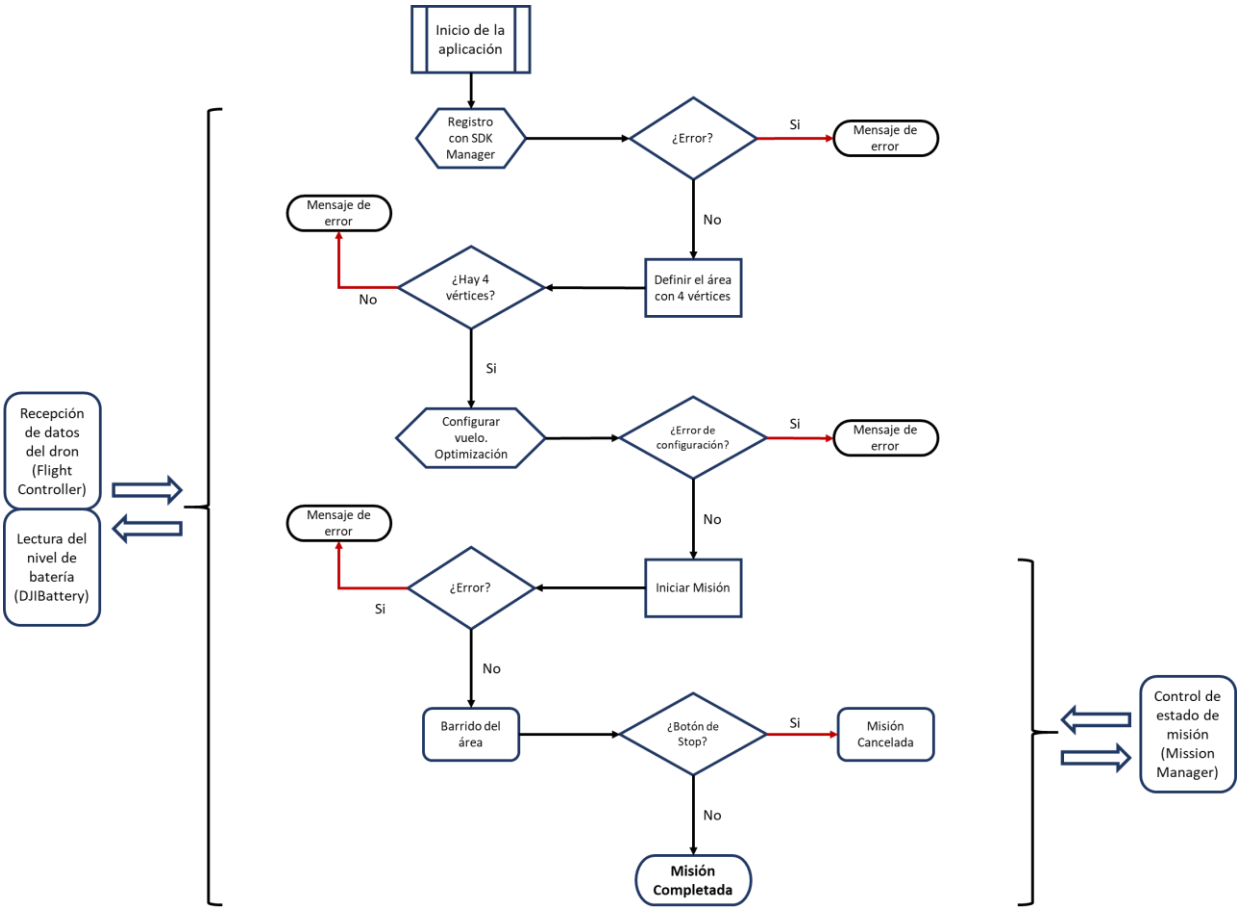


Figura 17. Estructura general de la aplicación desarrollada

Siguiendo el esquema (Figura 17), el punto de partida es el inicio de la aplicación. Lo primero que se hace de forma automática cuando se ejecuta la aplicación es inicializar todo lo relacionado con la interfaz de usuario: los valores de las etiquetas, el estado de los botones, el mapa…, así como configurar e instanciar todos los datos y variables que tengan un valor inicial o se empleen en el proceso de inicialización. Finalizado este proceso, se pasa a realizar el registro de la aplicación y la conexión con el dron. Para ello se hace uso de una clase que se presentaba anteriormente dentro de las clases principales del SDK, la clase *SDKManager*. Como ya se ha comentado anteriormente, este registro hace uso de una clave propia de la aplicación que debe incluirse en el código; esta clave se ha obtenido previamente en el espacio personal de Dji

developer (es necesario estar registrado) empleando algunas características de la aplicación como el bundle (nombre de la aplicación).

A partir de este punto, el SDKManager facilita dos formas diferentes de conectarse al producto en función de lo que vayamos a realizar, siendo necesario establecer en el código cuál de ellas se seguirá (la aplicación está preparada para elegir entre una u otra modificando un único valor en el código). Estas opciones son:

- Iniciar la conexión con el producto: la aplicación buscará el dispositivo disponible, que en este caso será el dron, y realiza el enlace entre ambos, obteniendo así una instancia del producto que permitirá acceder a él y a los elementos que lo componen.
- Entrar en modo depuración: se ha aclarado que Dji cuenta con una aplicación que soluciona la problemática de la depuración de aplicaciones, actuando como aplicación intermedia (BridgeApp). En este caso, para poder realizar la conexión con el producto, se ha introducido en el método correspondiente una identificación dada por la propia BridgeApp, estableciéndose en este caso la conexión entre la aplicación y BridgeApp, que a su vez se encuentra conectada al dron a través del control remoto.

En ambos casos, si se produce algún fallo durante el proceso, este se notificará por pantalla al usuario.

Si no ha habido ningún error en el registro y conexión, la aplicación ya se habrá iniciado y las conexiones necesarias se habrán realizado con éxito, quedando el programa listo y preparado para añadir puntos al mapa. Los puntos que se añadan deben delimitar el área rectangular sobre la que se quiere hacer un reconocimiento. Cada vez que se añada un punto, se añadirá también una etiqueta asociada a este que indica sus coordenadas, su número y su distancia respecto al último punto.

Si se han añadido los 4 puntos necesarios, se podrá acceder a la configuración del vuelo y a la optimización. Entre las configuraciones que se encuentran disponibles están los límites mínimo y máximo de altura entre los que se quiere buscar una solución o la velocidad máxima que no se debe superar. Serán todos estos requisitos y limitaciones los que se envíen a la optimización de vuelo y determinen cómo será el resultado final, una vez comprobada su validez.

En el apartado de configuración y optimización del vuelo se procesan de forma combinada las características del terreno definido a partir de los cuatro vértices y los requisitos que haya introducido el usuario previamente durante la configuración del vuelo. Con toda esta información, la aplicación devuelve los puntos que marcan el recorrido a seguir por el dron y los parámetros que determinarán las condiciones de altura y velocidad del vuelo, así como la configuración que se aplicará a la cámara.

Teniéndolo todo preparado para el vuelo, la aplicación queda a la espera de alguna orden por parte del usuario. A menos que se quiera cambiar el recorrido o las limitaciones, la orden esperada será simplemente la de iniciar la misión. En este punto el dron despegará en el caso de no encontrarse ya en el aire y seguirá el recorrido establecido. Ese recorrido, obtenido tras el proceso de optimización, barre toda el área de interés a la vez que se realizan capturas de la misma hasta llegar al punto final, momento en el que se deja de capturar imágenes del terreno y se ejecuta la acción especificada en la configuración. Mientras se realiza el barrido, existe la opción de cancelar la misión, en cuyo caso el dron se detendrá y cesará la captura de imágenes en ejecución.

Durante todo este proceso y desde la conexión con el producto (siempre que se cuente con un producto válido) el programa recibe información procedente del dron de forma constante a través de un método del FlightController, información que se emplea tanto para mostrar información por pantalla como para procesarla y realizar cálculos y acciones a partir de la misma. También de forma periódica, pero a partir del inicio de la misión, se obtiene información del estado actual de la misma, empleada sobre todo para detectar cuando se ha llegado al primer y último punto e iniciar o detener la captura de imágenes del terreno.

3.2 Clases principales de la aplicación

A continuación, se introducen brevemente las principales clases que conforman la aplicación con las características que contienen.

- **DJIRootViewController:** es el controlador principal de la vista de la aplicación. Aquí se localizan el registro de la aplicación y su conexión con el dron, la configuración de la misión, la respuesta de los distintos botones de la aplicación, el control del estado de la misión con el delegado de *MissionManager* y la adquisición de la información del *FlightController* entre otros.
- **DJIMapController:** controla las funciones relacionadas con la con la adición de anotaciones y etiquetas en el mapa (siendo esta última parte el principal añadido).
- **AppDelegate:** se crea por defecto. Provee métodos para configurar la aplicación y adaptarla a cambios en el sistema. No se ha modificado.
- **DJIGSButtonViewController y DJIWaypointViewController:** gestionan la ventana de configuración y los botones básicos de la aplicación, aunque se delegan las acciones de ambos en el controlador principal. Se han añadido las etiquetas y cuadros de texto necesarios para introducir todos los parámetros de configuración necesarios en la optimización.
- **DJIAircraftAnnotation y DJIAircraftAnnotationView:** gestionan la localización y orientación del dron en el mapa. No se han modificado.
- **EnableUtilities:** clase propia, creada para proporcionar los métodos relativos a la optimización del vuelo y evitar sobrecargar innecesariamente el controlador principal. En la Figura 18 puede observarse el código correspondiente a la cabecera de la clase (EnableUtilities.h). Los métodos que la forman son:
    - *directMetersFromCoordinate: toCoordinate:* permite medir la distancia entre dos puntos a partir de sus coordenadas, basándose en otro método existente que calcula la distancia entre dos puntos (*CGPoint*).
    - *checkIntroduced…*: comprueba la validez de los valores introducidos por la ventana de configuración y advierte de la presencia de errores.
    - *optimizateFlightOverRectangularField:…*: recibe los valores límite y el área a analizar, y la ajusta a un rectángulo en caso de ser necesario. Llama a otro método que recibe las dimensiones del área ajustada y devuelve los parámetros optimizados, para finalmente tomar esos mismos parámetros y enviarlos como resultado.

- *getOptimizedParametersFrom…*: recibe las dimensiones del área y los valores límite para la optimización. Inicia un bucle para buscar los parámetros óptimos en esas condiciones y devuelve los parámetros optimizados, o un indicador de la incapacidad de llegar a una solución si se ha superado la altura máxima.
- *getOptimizedCoordinatesToFlyFrom: withDivisions:* recibe los vértices del área y el número de divisiones necesarias. Procesa esta información para determinar el recorrido a seguir y obtener las coordenadas de los puntos necesarios para realizar dicho recorrido. Devuelve los waypoints del recorrido de forma ordenada.

```
@interface EnableUtilities : NSObject


// Método para medir distancias entre dos puntos

-(CGFloat)directMetersFromCoordinate:(CLLocationCoordinate2D)from toCoordinate:(CLLocationCoordinate2D)to;


// Método para comprobar que los valores introducidos por pantalla sean los adecuados

-(NSString*)checkIntroducedMaxAltit:(float)maxAltitude inAltit:(float)minAltit maxSpeed:(float)maxSpeed
    imgOverlap:(float)overlap timeLimit:(float)timeLimit withBatteryLimitTime:(float)batteryLimit
    withGoHomeAltit:(float)goHomeAlt;


// MÉTODOS DE LA OPTIMIZACIÓN DEL VUELO

// Método principal que recibe el array con los puntos del área a analizar y devuelve un array con todos los
    parámetros optimizados

-(NSArray*)optimizateFlightOverRectangularField:(NSMutableArray*)squareFieldCoordinates withLimitSpeed:(float)
    limitSpeed withLimitTime:(float)limitTime withMinimumHeight:(float)minimumHeight withMaximumHeight:(float)
    maximumHeight withOverlap:(float)imageOverlap;


//Método que determina y devuelve la velocidad, la altura, el intervalo de tiempo entre fotos y el número de
    divisiones que habrá que hacer en los extremos (para interpolar los waypoints) (EN ESTE ORDEN), a partir
    de las dimensiones del rectángulo y los valores límite de velocidad, tiempo, altura y solape

-(NSArray*)getOptimizedParametersFromLongLateral:(float)longLateral shortLateral:(float)shortLateral
    withLimitSpeed:(float)limitSpeed withLimitTime:(float)limitTime withMinimumHeight:(float)minimumHeight
    withMaximumHeight:(float)maximumHeight withOverlap:(float)imageOverlap;


// Método que recibe los vértices del área y el número de divisiones necesarias, y devuelve los waypoints
    ordenados para el vuelo.

-(NSArray*)getOptimizedCoordinatesToFlyFrom:(NSMutableArray*)squareCoordinates withDivisions:(int)
    numOfDivisions;

@end
```

Figura 18. Cabecera de la clase EnableUtilities (EnableUtilities.h)

3.3 Barrido de un área rectangular

En este apartado se va a detallar con más profundidad cómo se define sobre el mapa el área sobre la que se desea efectuar el barrido y obtener información. Para ello se han llevado a cabo algunas modificaciones en la interfaz que permiten introducir los puntos de varias formas diferentes.

En este trabajo, sólo se ha contemplado la introducción de áreas con formas rectangulares definidas por cuatro puntos, aunque también se ha preparado el programa para tener en cuenta el caso en el que los puntos introducidos no formen un rectángulo, sino un cuadrilátero, pero reduciéndose la calidad de la optimización a medida que aumente la deformidad del área introducida. Sin embargo, el hecho de que el área introducida para realizar el barrido sea rectangular no implica que el terreno de cultivo también deba serlo, ya que, independientemente de la forma del mismo, el barrido será igualmente válido siempre que el

área definida abarque toda el área de cultivo. Un aspecto a tener en cuenta es que la batería de los drones tiene una duración limitada, por lo que se debe ajustar todo lo posible el tamaño del área introducida.

En la Figura 19 se muestran cuatro puntos que definen un rectángulo determinado. Como se puede observar, cada uno de estos puntos está numerado, representando el orden en el que deben introducirse. No puede ignorarse dicho orden e introducir los puntos de forma arbitraria, ya que el código se ha preparado para tener en cuenta este orden y determinar el vuelo a seguir en función del mismo.

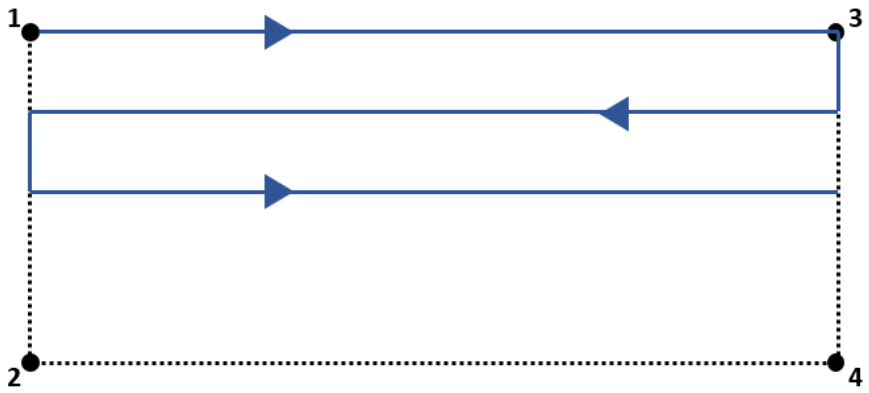


Figura 19. Orden de introducción de los vértices

El orden de introducción de puntos es importante ya que, de este modo, los dos primeros puntos forman uno de los lados de menor tamaño, mientras que los dos restantes forman el lado opuesto, de tal forma que el primer y el último punto se encuentren en los vértices más alejados entre sí. El barrido sobre el área comenzará siempre en el primer punto introducido y en dirección al tercer punto, continuando de la forma que se muestra en la figura. Cuál será el último punto vendrá determinado por el número de divisiones intermedias que sean necesarias para cubrir toda el área durante el barrido: con un número par, como 4 divisiones (3 tramos intermedios), el barrido acabaría en el último punto, mientras que con un número impar, por ejemplo 3 divisiones (2 tramos intermedios), el barrido finalizaría en el punto 3. Este hecho puede observarse en la Figura 20.

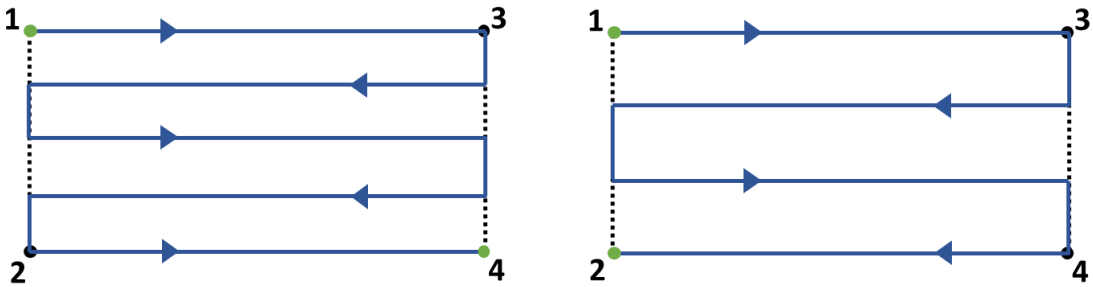


Figura 20. Diferencias entre número par e impar de divisiones

Una opción para formar el rectángulo es añadir directamente los cuatro puntos siguiendo el orden indicado. Tiene el inconveniente de que resulta complicado conseguir formar un rectángulo añadiendo 4 puntos de forma manual, pero se ha preparado la aplicación para aceptar cuadriláteros, no solo rectácgulos, ya que puede resultar interesante emplear este método en algunos casos. Un ejemplo podría ser un terreno situado en el cauce de un barranco

que se estreche en uno de sus extremos: si la diferencia en la anchura de los extremos es elevada y el barranco es profundo, cuando se realice el barrido definiendo un rectángulo puede darse el caso de que el dron colisione con las paredes del barranco, quedando seriamente dañado.

La segunda opción es definir un rectángulo a partir de únicamente dos vértices que formen parte de la diagonal del rectángulo, asegurando así que el área resultante tenga una forma completamente rectangular, como se aprecia en la Figura 21. Para ello se ha añadido un botón a la aplicación, asociado a la función que crea el rectángulo. Esta función parte de los dos vértices mencionados anteriormente, que dentro del código vienen definidos por sus coordenadas, y se aprovecha de un método propio del framework con el que trabaja el mapa. Este método es capaz de obtener, a partir de la coordenada de un punto, la posición en x e y que ocupa en la pantalla del dispositivo, y viceversa. De este modo, con las coordenadas en x e y de los vértices, resulta sencillo determinar los dos vértices restantes y transformar sus posiciones (x,y) en coordenadas geográficas. Aunque con dos puntos pueden obtenerse múltiples rectángulos con diferentes inclinaciones, el que devolverá esta función será aquel cuyos lados sean paralelos a los ejes, o lo que es lo mismo, a los bordes de la pantalla del dispositivo. Hay que recordar que el primer vértice introducido será el punto de inicio del barrido. Si por algún motivo se intenta formar un rectángulo a partir de un número de vértices diferente a 2, se advierte de este hecho con un mensaje por pantalla.

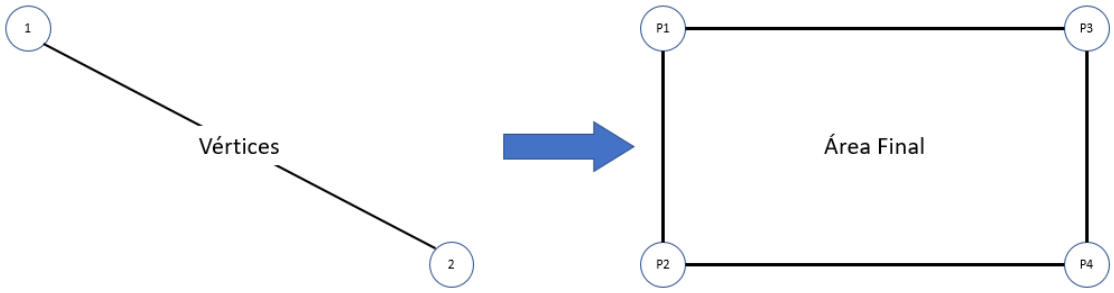


Figura 21. Definición del área a partir de dos vértices

Para añadir puntos al mapa, el programa original cuenta con un único método, tocar sobre el mapa en lugar que querías añadir como punto. Como este método, aunque rápido y eficaz, resulta impreciso, se ha añadido la opción de añadir los puntos introduciendo su latitud y su longitud. Para esto no sólo se han añadido las etiquetas y cuadros de texto correspondientes, sino que también ha sido necesario cambiar algunas de las funciones existentes. El método original empleado para añadir puntos detecta la posición de la pantalla donde se ha tocado y la transforma en sus coordenadas geográficas correspondientes, encargándose con todo esto de crear la anotación y almacenar el punto, además de integrar la modificación realizada para mostrar una etiqueta con su información. Como lo que se busca es añadir directamente el valor de las coordenadas, el método se duplica y modifica para realizar las mismas funciones, pero recibiendo directamente las coordenadas (Figura 22).

```objc
- (void)addPoint:(CGPoint)point withMapView:(MKMapView *)mapView
{

    CLLocationCoordinate2D coordinate = [mapView convertPoint:point toCoordinateFromView:mapView];
    CLLocation *location = [[CLLocation alloc] initWithLatitude:coordinate.latitude longitude:coordinate.longitude];
```

```objc
- (void)addPointFromLabels:(CLLocationCoordinate2D)coordinate withMapView:(MKMapView *)mapView
{

    CGPoint point = [mapView convertCoordinate:coordinate toPointToView:mapView];
    CLLocation *location = [[CLLocation alloc] initWithLatitude:coordinate.latitude longitude:coordinate.longitude];
```

Figura 22. Añadir punto: diferencia entre el método original (arriba) y el modificado (abajo)

Las últimas modificaciones a destacar en este punto son la posibilidad de cambiar el tipo de mapa, ya que para introducir puntos con la pantalla táctil es mucho más sencillo emplear la vista de satélite, y la opción de mostrar o esconder las etiquetas con la información de cada punto, porque en determinadas situaciones dificultan el manejo de la aplicación, y cuando hay muchos puntos cercanos no es posible distinguirlas. En la Figura 23 se muestran estos añadidos en la interfaz de la aplicación. La opción de añadir las coordenadas sólo se encuentra disponible cuando se está en modo de edición (es decir, se ha pulsado sobre el botón *Add* para habilitar la introducción de puntos), por lo que no está disponible al iniciar la aplicación.

En la parte inferior de la pantalla se encuentran las opciones relativas a la presentación de etiquetas con la información de los puntos (izquierda) y al cambio de vistas del mapa (derecha). En la parte derecha de la pantalla se encuentran, además, el botón que compone todos los vértices del rectángulo a partir de dos puntos y la sección que permite añadir los puntos directamente con sus coordenadas.

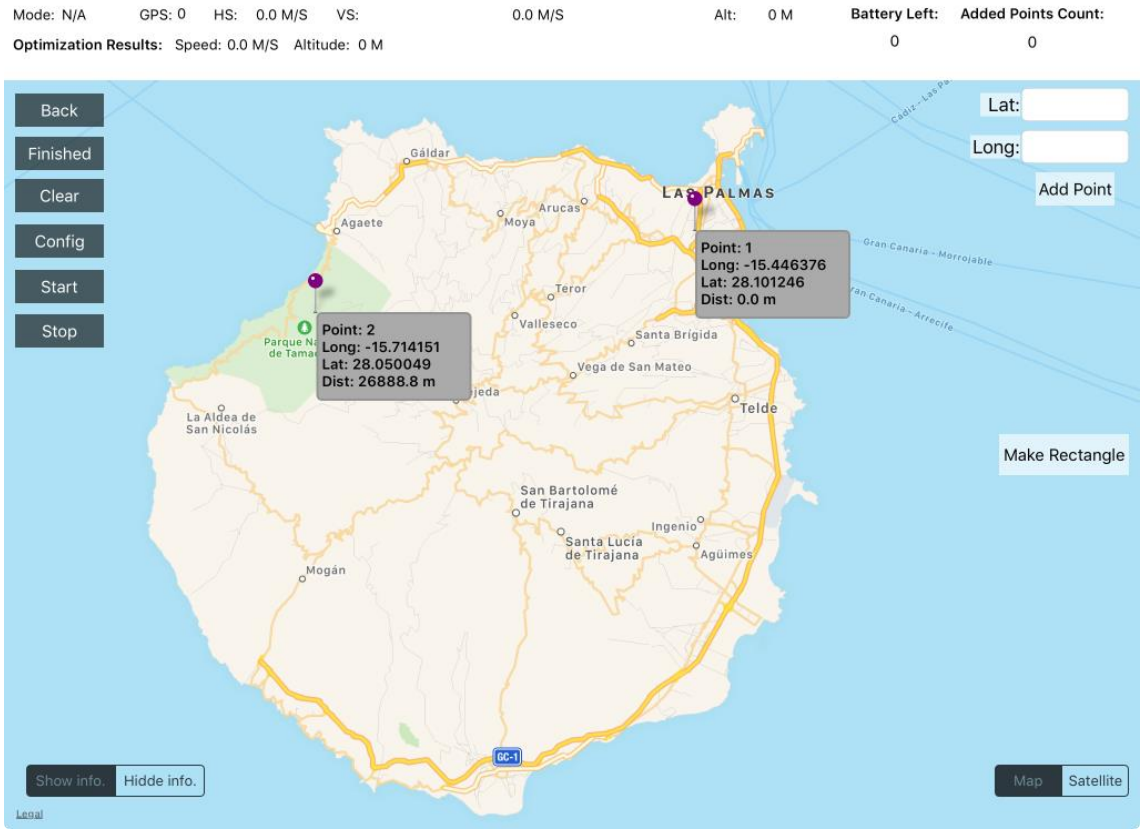


Figura 23. Modificaciones para mejorar la adicción de puntos de la aplicación

3.4 Parámetros de configuración

Ahora vamos a mostrar con más detalle las opciones de configuración previas al proceso de optimización de vuelo. En la aplicación original, estas opciones se basaban en introducir directamente los parámetros de la misión: altura de vuelo, velocidad, etc. Sin embargo, el objetivo que se persigue es que sean precisamente estos parámetros los que se determinen de forma automática, por lo que resulta imprescindible realizar un cambio en la mayoría de las opciones de configuración disponibles, adaptándolas a los nuevos requisitos.

Para este trabajo, las opciones de configuración deben ser una serie de valores que limiten el rango de búsqueda de los valores óptimos, ya que si se tomase un rango muy elevado podría tardarse demasiado en extraer un resultado o directamente obtener un resultado que no es válido. En la Figura 24 se muestra la ventana de configuración adaptada a estas necesidades.

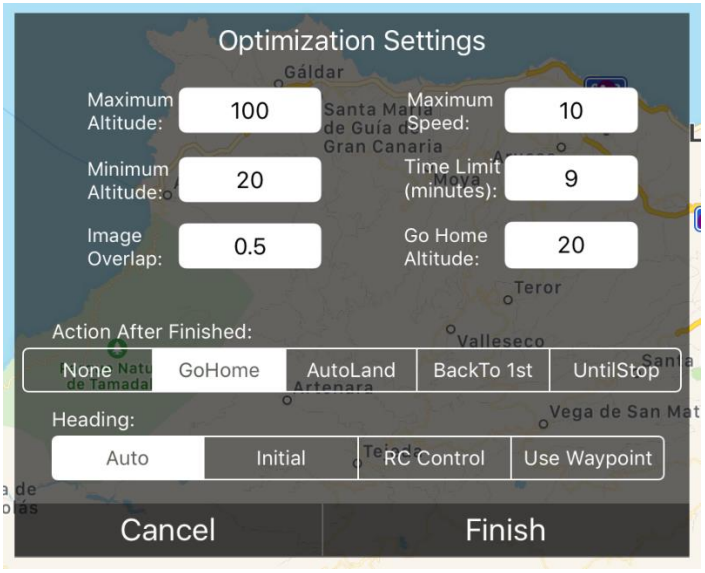


Figura 24. Opciones de configuración para la optimización

Uno de estos valores es el solape entre imágenes, representado en la Figura 25. Este parámetro resulta especialmente importante durante la optimización, ya que variará en gran medida el área abarcada por cada imagen y será necesario un barrido más largo. Aunque este valor se puede modificar, es necesario mantenerlo siempre por encima del mínimo necesario, ya que una diferencia por exceso mejorará el resultado, a costa de un mayor tiempo de computo. Este mínimo variará dependiendo de las características del terreno: entre más homogéneo sea, el software necesitará un solape mayor para diferenciar correctamente las distintas zonas del terreno. En las especificaciones de algunos softwares específicos [12][13]se fijan estos valores mínimos de solape entre un 70% y un 85% para terrenos agrícolas homogéneos. Durante este trabajo se ha probado un solape del 70% en OpenDroneMap, obteniéndose resultados satisfactorios.

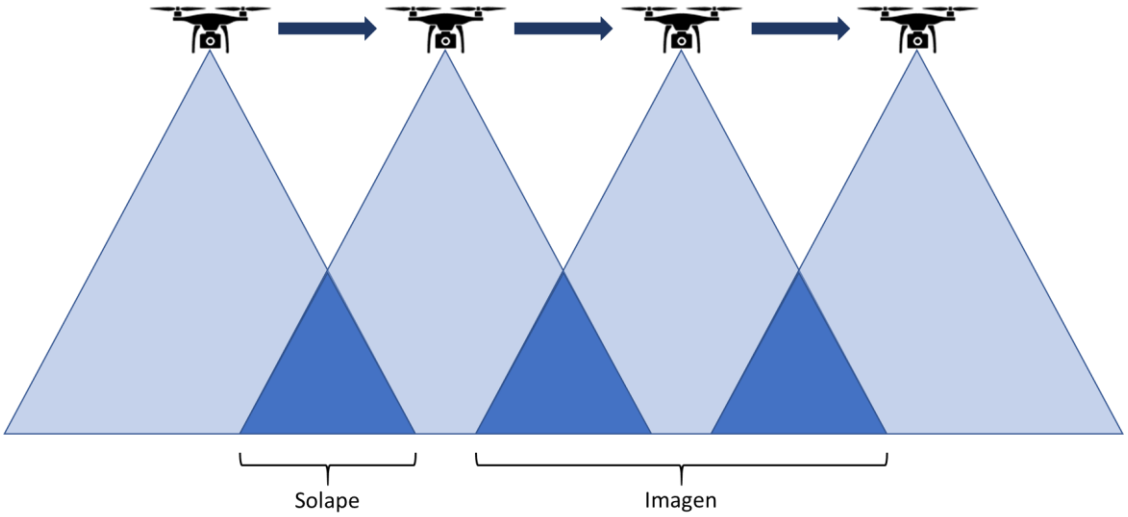


Figura 25. Solape entre imágenes aéreas

Aparte de estos valores límite, también se incluye la opción de cambiar la altitud de vuelta a casa. Cuando el dron está volando o realizado una misión a una altura determinada y se ejecuta el comando de vuelta a casa, es decir, se le ordena volver al punto donde despegó, este tiene configurada una altura determinada para ejecutar ese camino de vuelta, independientemente de cuál sea la altura a la que se encontraba previamente.

El último valor a tener en cuenta, probablemente el más importante de todos ellos, sea el tiempo de vuelo. El tiempo va a limitar todas las capacidades del dron, ya que con niveles bajos de batería no dará tiempo de recorrer toda la zona y mucho menos de hacerlo con una resolución de imagen adecuada. Para poder determinar el tiempo de vuelo restante en función de la batería, se necesita conocer qué porcentaje se tiene en el momento de realizar la misión, valor que es posible obtener mediante las herramientas del SDK. En los apartados posteriores se explicará con más detalle este valor y cómo se define en la interfaz de configuración.

Como el dato importante es el tiempo de vuelo restante y resulta complicado estimar con exactitud el valor exacto, se ha decidido determinar el valor siguiendo una sencilla regla de 3 a partir del tiempo de vuelo con la batería completa y el porcentaje restante. El fabricante indica que el tiempo de vuelo puede ser de hasta 28 minutos en condiciones óptimas, pero en la práctica se ha comprobado que, haciendo uso regular de las características del dron como la cámara y el posicionamiento por GPS, el tiempo de vuelo ronda los 20 minutos.

Partiendo de este dato, se ha decidido añadir un margen de seguridad del 25%, es decir, 5 minutos menos de tiempo de vuelo, ya que ha podido observarse que, ante situaciones climatológicas adversas, la duración de la batería se reduce debido al gasto energético necesario para estabilizar el vuelo. De esta forma, el tiempo de vuelo disponible se considera finalmente como **15 minutos** con la batería completa, valor con el que se estimará el tiempo de vuelo restante. Este cálculo se realiza automáticamente y el resultado aparece en la ventana de configuración de forma directa, pudiendo reducirse en función de las necesidades actuales.

Para poder ejercer un mayor control sobre cada uno de estos parámetros, se fijan en el código una serie de límites para los mismos, todo ello para garantizar un margen de seguridad en caso de sucederse algún tipo de error. Actualmente, dichos límites son:

- **Altura máxima:** debe estar comprendida entre la altura mínima y un máximo de 120m.
- **Altura mínima:** debe ser mayor de 20m.

- **Solape entre imágenes:** debe estar comprendido entre 0 y 1 (superposición entre 0% y 100%).
- **Velocidad máxima:** inferior a 15 m/s y por encima de 1 m/s
- **Tiempo límite:** situado entre 0 y el tiempo restante calculado con el nivel de batería actual.
- **Altura de vuelta a casa:** situada entre 20 y 120 m.

Todos estos límites pueden ser modificados en el código, teniendo en cuenta que el dron tiene unos límites físicos de velocidad y altura que puede alcanzar y que la altura de vuelta a casa está limitada al rango de 20-500 m de fábrica. Al finalizar la configuración, se ejecuta una función de comprobación previa que revisa la validez de todos estos valores antes de pasar a la optimización del vuelo. Si alguno de los parámetros no se corresponde con el rango de valores asignado, se muestra un mensaje que advierte de los errores encontrados y recuerda cuáles son los límites de los mismos, como se puede ver en la Figura 26.

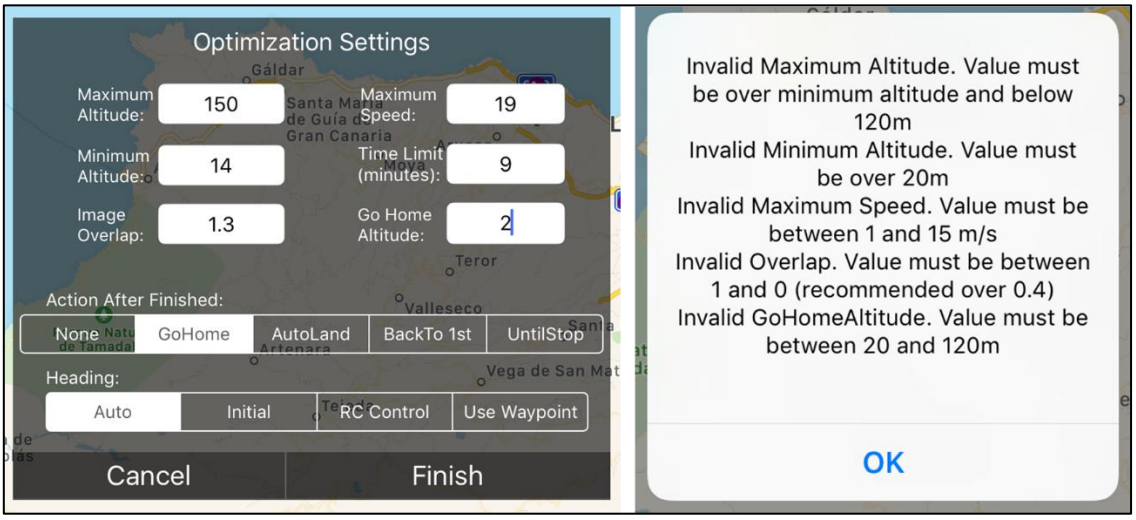


Figura 26. Ejemplo de configuración errónea

3.5 Optimización del vuelo y obtención de parámetros finales.

Una vez definidas las primeras partes del proceso, es el momento de entrar en la parte más importante del programa: la optimización. Todos los procedimientos y consideraciones previos se han tomado orientadas a este proceso, de tal forma que los datos y parámetros lleguen a este punto listos para entrar en un bucle de optimización que se ha divido en tres partes. Primero vamos a explicar en qué parte del programa se transforman los puntos en Waypoints, dónde se localiza el código relativo a esta parte y algunas otras aclaraciones previas antes de entrar en dicho bucle de optimización y analizar las decisiones que se han tomado.

3.5.1 Aclaraciones previas

En el apartado anterior se explicaba cómo el último paso de la configuración de los parámetros era la validación de los valores introducidos. En realidad, este último paso está

encadenado directamente con la optimización, ya que al pulsar el botón de finalizar es cuando se ejecutan de forma automática la validación ya mencionada, la optimización del vuelo y la configuración de la misión con los resultados de este último paso.

Cabe destacar que, en este punto del programa, el código ya ha crecido de forma moderada, debido a que todo el control de la interfaz se lleva en el controlador de la vista, la clase "DJIRootViewController". Como aún quedan por añadir en este mismo controlador todas las líneas de código relacionadas con la configuración del vuelo, además de algunas otras utilidades que se explicarán en apartados posteriores, surge la necesidad de liberar en la medida de lo posible la carga sobre esta clase, ya que no solo dificulta el seguimiento y mantenimiento del código, sino que además facilita la aparición de errores en el mismo. El ejemplo ya integraba una clase independiente, "DJIMapController", que se ocupa de controlar ciertos aspectos del mapa, como la adición y almacenamiento de puntos o la actualización del icono del dron; también es en esta parte de la aplicación donde se ha añadido el control de las etiquetas de cada punto, con su coordenada, número, etc. como ya se mencionó anteriormente.

Con esta premisa se ha añadido otra clase aparte denominada "EnableUtilities", que contiene los métodos necesarios para optimizar el vuelo. Es precisamente en esta clase donde se encuentra la función que valida los valores introducidos en la configuración, además de un método para calcular la distancia entre dos puntos.

Otra acción que puede pasar desapercibida pero que resulta fundamental, es la instanciación de la misión con sus respectivos Waypoints, aunque estos no sean los definitivos. Todo este proceso se lleva a cabo justo antes de que se abra la ventana de configuración de tal forma que, cuando se pase a la optimización del vuelo, la misión y los Waypoints correspondientes a los vértices se encuentren disponibles. En este punto, además, se toma el valor del tiempo de vuelo restante calculado a partir del porcentaje de batería y se añade como valor predefinido en los parámetros de configuración. En la Figura 27 podemos ver el proceso que se sigue desde que se inicia la configuración hasta obtener el vuelo optimizado para el área introducida.

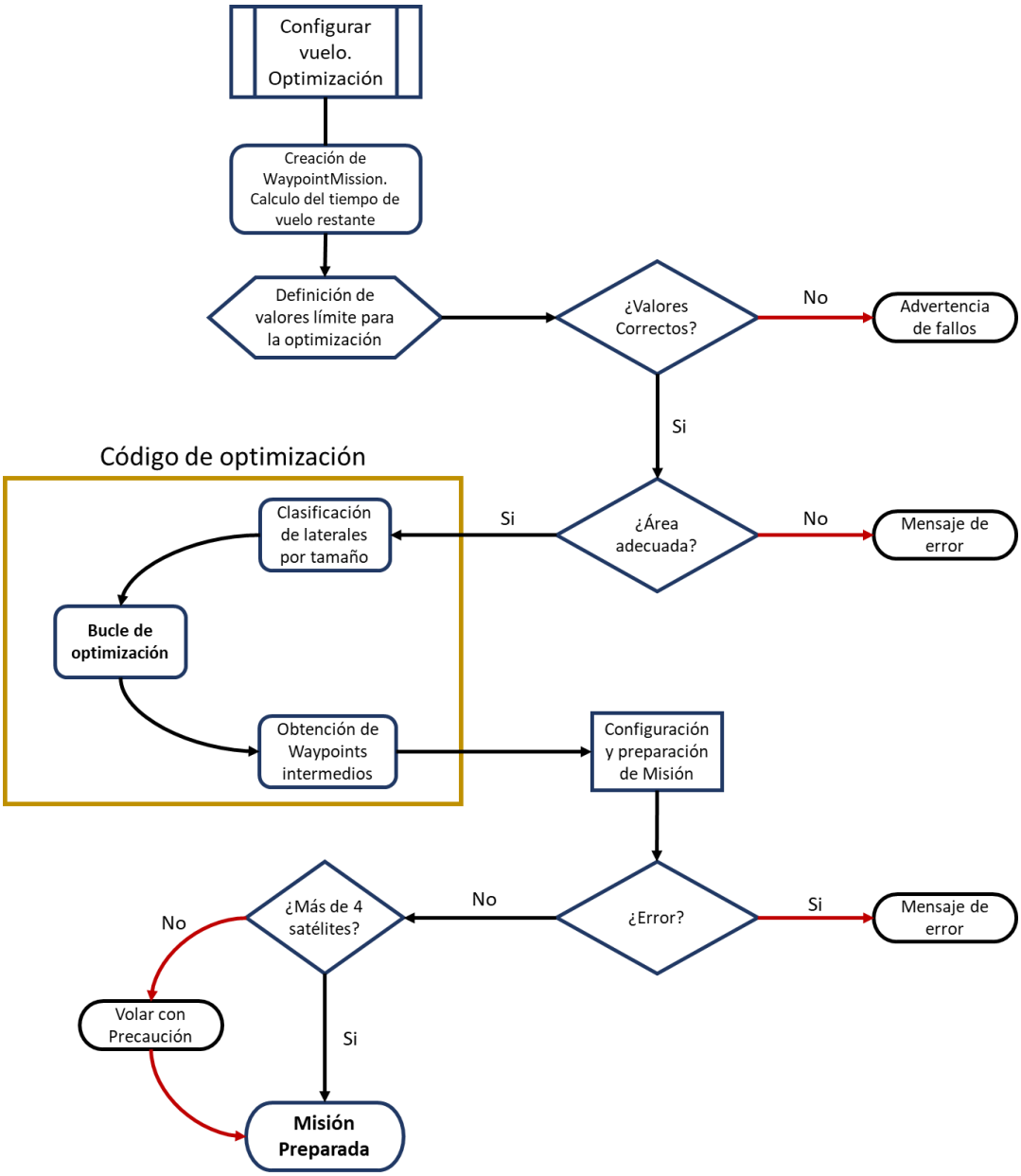


Figura 27. Proceso de configuración, optimización y preparación del vuelo y la misión

Después de esto, se lleva a cabo el proceso explicado en el apartado anterior sobre los parámetros de configuración, y si los valores son correctos además de tenerse un área definida por cuatro vértices, se pasa al proceso de optimización, para finalmente configurar y preparar la misión con los resultados obtenidos. Finalmente, siempre que no se detecten errores, se comprueba que el número de satélites sea suficiente, advirtiendo del peligro en caso negativo, y finalizando la preparación de la misión.

3.5.2 Valores de FOV y funcionamiento de la cámara

El último paso antes de entrar en la optimización es aclarar los valores del FOV de la cámara y el modo de funcionamiento de la misma. Respecto al primer punto, el FOV que da el fabricante es de 94º diagonal para el Phantom 4, pero para el cálculo de la resolución espacial de las imágenes se necesitan los valores del FOV horizontal y vertical. Afortunadamente, en una sección de la documentación dedicada a la cámara [14], se da el ejemplo de una cámara prácticamente idéntica a la incluida en el Phantom4: sensor de 4000x3000 pixeles y lente de FOV 94°, 20 mm, concluyendo que los valores del FOV horizontal y vertical son de 84° y 62° respectivamente.

En lo referente a la cámara, esta cuenta con dos modos de funcionamiento que pueden ser útiles para lograr capturar imágenes del terreno: la captura individual de imágenes y la captura por intervalos, tomando una imagen cada determinado espacio de tiempo. Utilizar el primer método requeriría generar una nube de puntos sobre el mapa, uno por cada imagen que sea necesario tomar, y contrastar la posición del dron en todo momento con la nube de puntos para capturar una imagen cada vez que se dé una coincidencia.

La complejidad de todo este proceso hace que la segunda opción sea la elegida para realizar la captura. Con el Phantom 4, el mínimo intervalo de tiempo es de 2 segundos en formato JPEG y 5 segundos en formato RAW. El intervalo debe ser un valor entero igual o mayor al mínimo del formato empleado (en este caso JPEG). Utilizando la captura por intervalos, únicamente será necesario ajustar el intervalo de tiempo a aplicar con la velocidad y la distancia a recorrer entre una foto y otra, eliminando la necesidad de las nubes de puntos y el contraste entre posiciones.

Para obtener unos resultados satisfactorios, se ha empleado la aplicación de DJIGo para fijar una configuración específica de la cámara. Tras varias pruebas, se ha determinado que la configuración que da mejores resultados en la mayoría de los casos es fijar la relación entre la sensibilidad (ISO) y la velocidad del obturador a 0, manteniendo el resto de parámetros en modo automático.

3.5.3 Clasificación de los laterales

Si bien existen dos formas de definir un rectángulo mediante la aplicación desarrollada (definiendo los cuatro vértices, o definiendo los dos vértices que conforman una diagonal), si se introducen de forma manual los cuatro vértices podría generarse un cuadrilátero cuyos ángulos no coincidieran con los de un rectángulo (todos los ángulos de 90 grados). Es por ello por lo que es necesario hacer ciertas modificaciones que nos aseguren que el área barrida se ajuste al área de interés.

Para asegurar la calidad del resultado, manteniendo siempre el mayor solape posible entre las imágenes, se realiza una comparación entre los laterales que deberían ser paralelos entre sí, tomando el mayor en cada caso y optimizando el vuelo como si se tratase del rectángulo formado por los lados mayores. En la Figura 28 se refleja mejor el significado de esta comparación con un caso práctico, donde a la izquierda se muestra el área introducida y a la derecha el rectángulo equivalente generado sobre el que se aplicará la optimización; este rectángulo no representa un área real y se emplea únicamente para optimizar el vuelo garantizando los límites especificados.

De esta forma, cuando se determine el número de tramos intermedios que debe tener el barrido, se habrán realizado los cálculos para cumplir con los requisitos del peor escenario posible, el de un rectángulo de área máxima. Este proceso de clasificación se implementa en uno de los tres métodos creados específicamente para la optimización del vuelo.

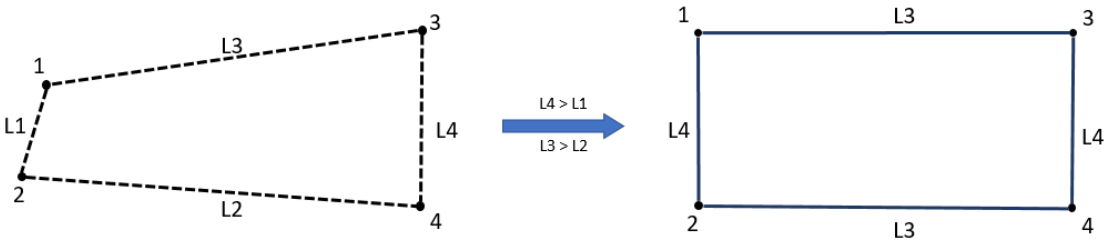


Figura 28. Comparación de los lados del polígono para asegurar solape

Llegados a este punto, aplicar ese número de divisiones calculado al área irregular solo podrá mantener o aumentar el solape entre las imágenes, pero nunca reducirlo, ya que el área a cubrir será igual o menor al área del peor caso con la que se han determinado. En la Figura 29 se muestra qué aspecto tendría el resultado final en caso de haberse obtenido, por ejemplo, un barrido con 3 divisiones (dos tramos intermedios). Se ajusta como se muestran en la figura para garantizar que sólo se sobrevuela el área especificada, evitando así colisiones con posibles obstáculos externos al área (por ejemplo, torres eléctricas, edificios, montañas…). Puede apreciarse la diferencia existente con el barrido de un área completamente rectangular, recordando el caso expuesto en la Figura 20.

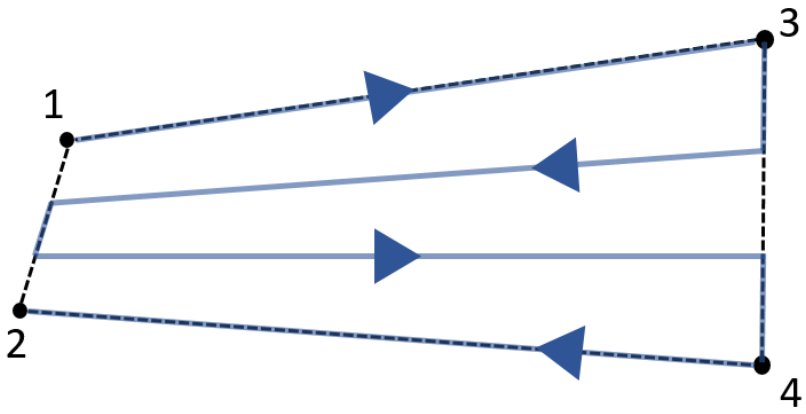


Figura 29. Resultado con área irregular y barrido de 3 divisiones

En esta última figura puede apreciarse también la finalidad de introducir los puntos en un orden determinado y distinguir entre los lados de mayor y menor dimensión del rectángulo o área definida. En este ejemplo hipotético, al tomarse los lados de menor dimensión para alternar entre cada tramo, se consigue reducir el número de recorridos que deben realizarse durante el barrido, reducir el número de divisiones.

Si se hubiesen tomado los lados mayores para realizar los giros, el resultado sería un barrido con más tramos de menor longitud. El problema de este caso es que entre mayor sea el número de tramos, mayor será el número de giros a realizar y por lo tanto se perderá más tiempo de vuelo para recorrer la misma área. Por lo tanto, la máxima a seguir en este aspecto es reducir el número de giros a realizar para aprovechar mejor la batería disponible.

3.5.4 Obtención de parámetros optimizados

En el paso anterior se obtienen las dimensiones del rectángulo sobre el que se aplicará la optimización en el caso de haber introducido los cuatro vértices del área de interés a mano; si se ha empleado el método facilitado para definir el rectángulo a partir de dos vértices, las dimensiones obtenidas serán exactamente iguales a las introducidas al tratarse de un rectángulo. Una vez determinada cual será el área definitiva para aplicar la optimización, se llama al segundo método de optimización creado, que recibe las dimensiones del área y los valores límite de la optimización introducidos en la ventana de configuración.

A continuación, se entra en el bucle de optimización (Figura 30). El primer paso es definir la altura mínima como altura inicial, ya que una de las premisas que se busca es tener la mayor resolución posible. De esta manera, primero se comprobará si es posible barrer el área con la altura mínima y solo en caso negativo se procederá a aumentar la altura para volver a realizar la comprobación, y así sucesivamente.

Definida la altura inicial, es necesario determinar qué dimensiones tendrán las imágenes captadas a esa altura para determinar el recorrido a seguir. Partiendo de la altura y de los valores del FOV horizontal y vertical, añadidos en el código como constantes con valores de 84° y 62° como se mencionó anteriormente, las dimensiones de la imagen se obtienen aplicando trigonometría, de acuerdo con los cálculos que se muestran en la Figura 31. Cuando se habla de la distancia horizontal de la imagen, se hace referencia al lado de mayor dimensión (4000 píxeles), y la distancia vertical indica el lado de menor dimensión (3000 píxeles).

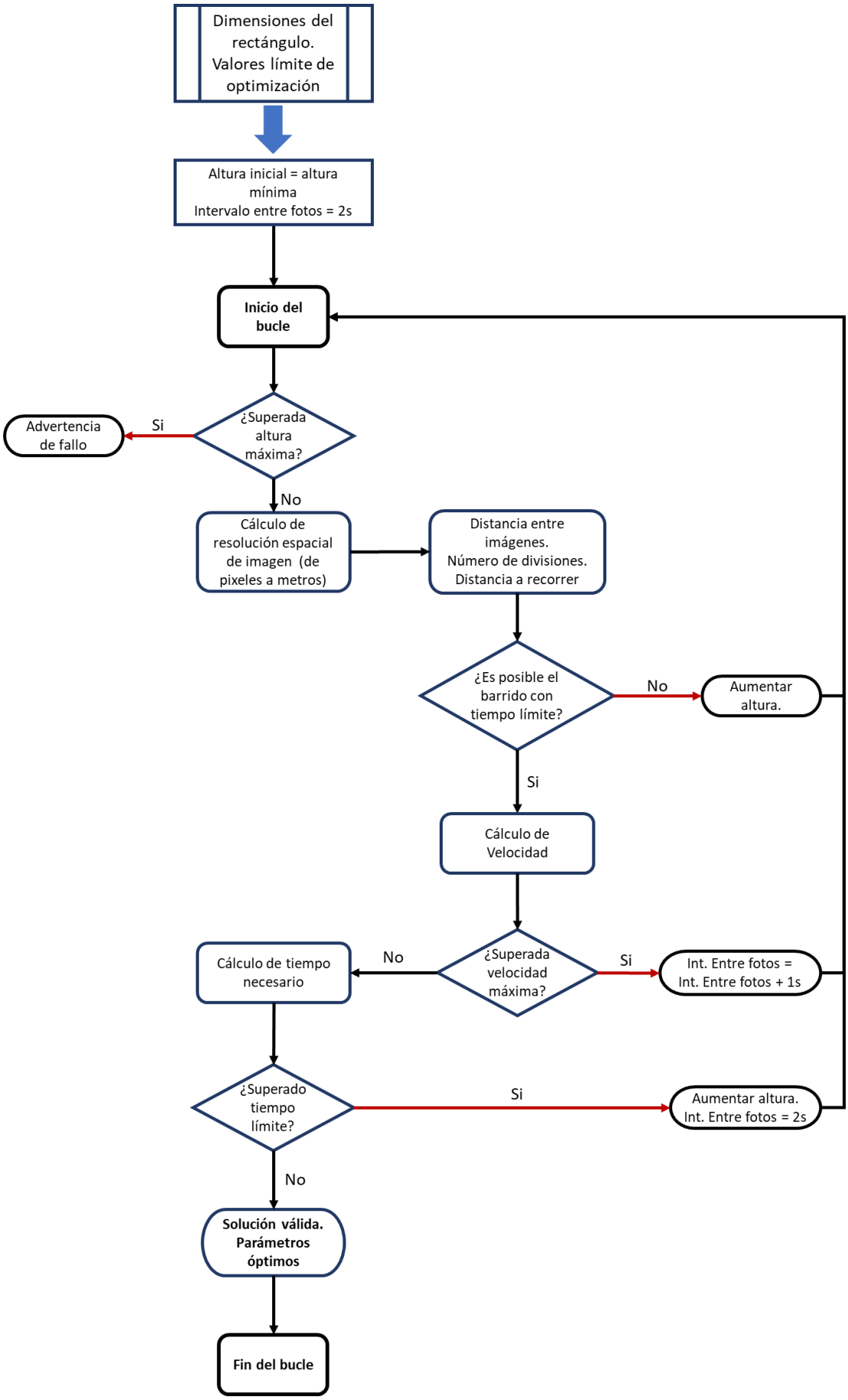
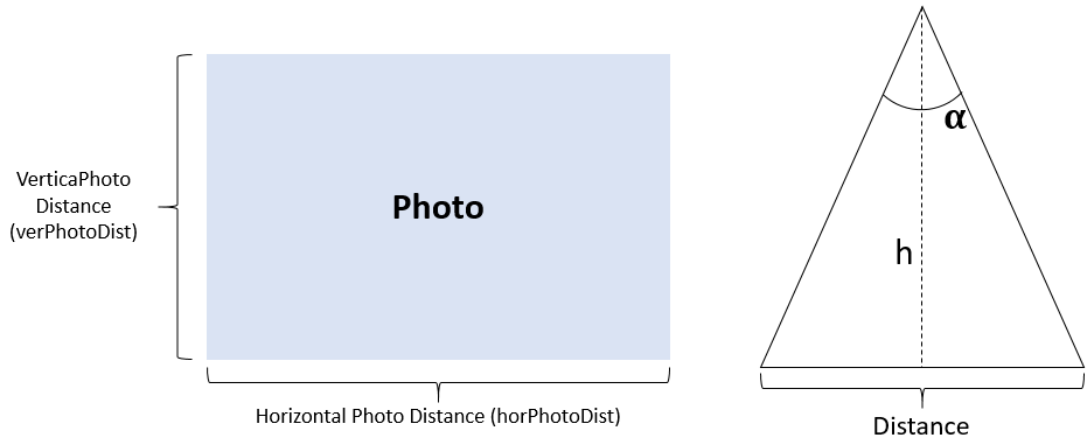


Figura 30. Esquema del bucle de optimización.



$$\text{Distance} = 2 * \text{h} * \tan\frac{\alpha}{2} \begin{cases} \text{horPhotoDist} = 2 * \text{h} * \tan\dfrac{hFOV}{2} \\ \text{verPhotoDist} = 2 * \text{h} * \tan\dfrac{vFOV}{2} \end{cases}$$

Figura 31. Cálculo de las dimensiones de una imagen en función de del FOV y la altura

Con la resolución espacial de la imagen, se procede a calcular la distancia que debe existir entre una foto y otra en función del solape especificado. Para ello se toma un único valor de solape que coincide tanto en la vertical como en la horizontal; en caso de ser necesario, podría añadirse una diferenciación entre solape horizontal y vertical mediante modificaciones sencillas en la aplicación. En la Figura 32 se muestra cómo realizar el cálculo, distinguiéndose las distancias horizontales y verticales. Las distancias horizontales entre las fotos determinarán el número de divisiones necesarias para realizar el barrido, obteniendo así el recorrido a seguir. Las verticales, por su parte, se emplearán para determinar la velocidad que se debe tomar y el intervalo que debe haber entre fotos.

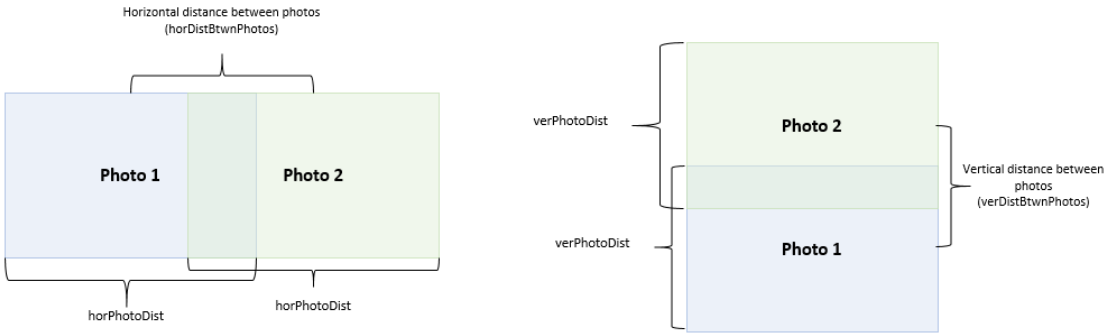


Figura 32. Cálculo de la distancia entre fotos

El problema de estas distancias es que difícilmente coinciden de forma exacta con el tamaño del área a barrer. Por ejemplo, si el área a barrer tiene unas dimensiones de 400x100m, y la distancia que se ha determinado entre fotos es de 28 m, al calcular el número de divisiones

necesarias para barrer el área se obtienen 100/28 = 3,57 divisiones. La solución es tomar el valor entero inmediatamente superior, en este caso 4 divisiones, cambiando la distancia entre fotos a 25m y quedando perfectamente definido el recorrido a seguir.

Esta solución se aplica a cualquier caso: se comprueba si el resultado obtenido es un valor entero, y en caso de no serlo, se toma el valor entero inmediatamente superior. Se aplica, por lo tanto, una aproximación por exceso, ya que en algunas situaciones un redondeo convencional llevaría a reducir las divisiones en el barrido con su consiguiente reducción en el solape entre las fotos, y se ha establecido que siempre se debe tomar la solución que aumente el solape. Para la distancia vertical se sigue el mismo procedimiento: se comprueba si es un valor entero, y en caso de no serlo, se reajusta la distancia para así tener un número exacto de fotos en cada pasada.

La distancia que será necesario recorrer durante el barrido puede obtenerse de forma casi inmediata. Cada pasada tendrá la misma distancia que la longitud del lado mayor del área definida, y el número de pasadas será igual al número de divisiones que se hayan determinado más una unidad. Por otro lado, entre una pasada y otra se recorre una distancia equivalente a la distancia del lado menor del área. En la Figura 33 se representa un ejemplo con 4 divisiones, indicando cuál sería el recorrido a seguir y cómo se calcula la distancia que se cubre con el mismo.

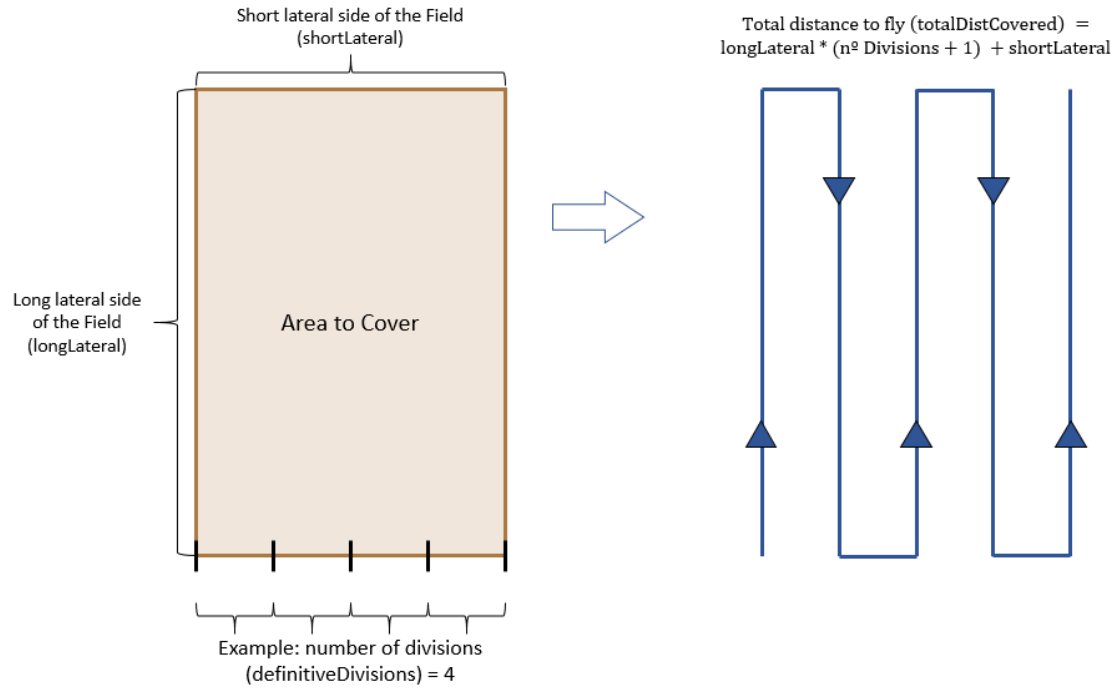


Figura 33. Ejemplo de la determinación y cálculo de la distancia a recorrer

Una vez calculada la distancia que se debe recorrer, el siguiente paso es determinar cuánto tiempo se tardaría en finalizar el barrido y si sería posible recorrerlo con los límites establecidos. Para ello, se comprueba el tiempo que sería necesario para realizar el barrido si se vuela a la distancia límite introducida. En realidad, a la hora de calcular ese tiempo, no se emplea directamente la velocidad límite, sino un 90% de la misma para tener un cierto margen de seguridad, teniendo en cuenta que hay momentos del vuelo en los que se emplea un cierto tiempo para realizar los giros.

El tiempo obtenido tras ese cálculo se contrasta con el tiempo límite, contemplándose dos resultados posibles:

- Que el tiempo calculado sea mayor o igual al límite: esto significaría que el recorrido que se ha determinado para esa altura es demasiado largo como para recorrerlo con el nivel de batería disponible.
- Que el tiempo calculado sea menor al tiempo límite. Esto significa que el recorrido obtenido es una posible solución a la optimización del vuelo.

Si se da el primer caso, se aumenta la altura con la que se determina el recorrido (la altura inicial es la mínima especificada) en 1 m y se vuelve a repetir todo el proceso, calculando la resolución de las imágenes y el recorrido con el nuevo valor de la altura. La altura se está variando en saltos de 1 metro, pero puede modificarse este valor para buscar un ajuste más o menos restrictivo. Se ha utilizado un intervalo de 1 metro, ya que, como se ha podido comprobar, con este ajuste se consiguen unos resultados adecuados para esta aplicación. Si se desea cambiar el ajuste, bastará con modificar en el código el valor del salto entre alturas de 1 metro al valor deseado.

En el segundo caso, se procede a calcular la velocidad y el intervalo de tiempo entre fotos necesario. Para ello, se fija el valor del intervalo entre fotos en el valor mínimo, esto es, 2 segundos. De este modo, ya se tiene el valor de la distancia entre una foto y otra, así como el valor del tiempo en el que se debe recorrer, y resulta sencillo calcular la velocidad necesaria. El hecho de partir de un intervalo de tiempo tan reducido provoca que la velocidad obtenida sea la máxima posible, por lo que se debe comprobar que este valor de velocidad no sea mayor que la velocidad máxima admisible (la velocidad límite). Si resulta que la velocidad calculada es mayor o igual a la velocidad límite, se aumenta el intervalo entre fotos en un segundo y se vuelve a iterar. De este modo, el valor de la velocidad calculada irá disminuyendo progresivamente hasta llegar a un punto en el que se trate de una velocidad válida.

Llegados a esta parte del programa, se vuelve a revisar que con la velocidad que se ha determinado es posible realizar el barrido dentro del tiempo disponible, ya que aunque anteriormente se diese un ligero margen de error al emplear el 90% de la velocidad límite en los cálculos, podría suceder que con un intervalo entre fotos la velocidad obtenida sea mayor que la velocidad límite, mientras que con el siguiente intervalo entre fotos disponible la velocidad se reduzca hasta el punto de ser insuficiente para realizar todo el barrido. En este caso, se aumentaría el valor de la altura en 1 m (o en el valor fijado en el código en caso de haberse modificado) y se reiniciaría el intervalo de tiempo entre fotos a 2 segundos, volviendo a repetir todo el proceso con el nuevo valor de altura.

Si se cumple el límite de tiempo, entonces la solución será válida, permitiendo realizar el barrido del área con un valor mínimo de altitud a la vez que se respetan todos los límites de velocidad, tiempo y solape. Los valores que se tienen en ese momento de altura, velocidad, intervalo de tiempo entre fotos y número de divisiones necesarias (empleado para saber los tramos necesarios del barrido e interpolar los Waypoints a utilizar) se almacenan y se devuelven como parámetros optimizados, siendo el resultado de la optimización del vuelo.

Si durante el bucle de optimización se alcanzase el límite de la máxima altura permitida, se saldría del bucle y se mostraría un mensaje informando de la incapacidad de optimizar el vuelo del área cumpliendo los límites establecidos.

3.5.5 Obtención de los Waypoints necesarios

En los pasos anteriores se han realizado todos los cálculos y estimaciones necesarios para optimizar el vuelo, obteniendo la altura, velocidad, intervalo de fotos y número de divisiones que se emplearán durante el vuelo. El último paso es tomar el barrido que se ha determinado y extraer los Waypoints que permitan definirlo en la aplicación. Para ello se hace uso del tercer y último método de optimización creado, que recibe el área definida y el número de divisiones necesarias para recorrerla.

Como el área introducida puede no ser exactamente un rectángulo, sino un cuadrilátero, lo primero que se hace es tomar los lados menores de la misma y almacenarlos para emplearlos en el cálculo (estos lados se determinan directamente sabiendo que son los correspondientes a los vértices 1 y 2 por un lado, y 3 y 4 por el otro). Como los vértices 1 y 3 siempre serán el primer y segundo puntos de la misión, se crean los Waypoints correspondientes a los mismos y se almacenan. Es fundamental para el correcto funcionamiento de la misión respetar el orden en el que deben recorrerse los Waypoints, por lo que se cuidará este aspecto a la hora de almacenarlos.

Aunque las coordenadas geográficas no son coordenadas planas, puede aplicarse la interpolación lineal para determinar las coordenadas intermedias cuando la distancias con la que se trabaja no son excesivamente largas, todo ello debido a las dimensiones de la tierra, que, con un radio medio de 6371 Km, hacen que la extensión del área que puede abarcar un dron, dada su batería y su altura de vuelo, pueda ser tratada de esta forma para el cálculo. En la Figura 34 se muestra el cálculo que se debe realizar para extraer la longitud y latitud de un punto a partir de las coordenadas de los extremos y el número de divisiones.

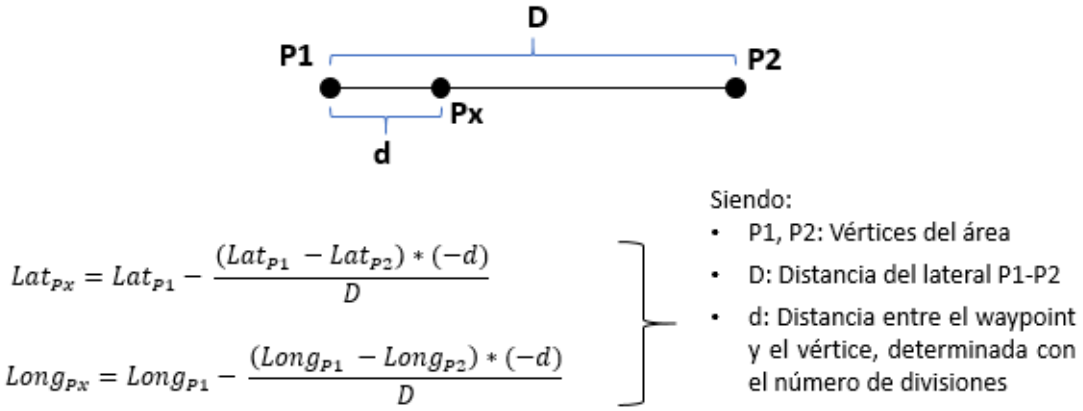


Figura 34. Interpolación de waypoints

La distancia entre cada waypoint y el vértice se determinará dependiendo del tramo del barrido que se esté interpolando; así, la distancia hasta el primer waypoint intermedio vendrá dada por $D\ (long.del\ lateral)/Nº\ de\ divisiones$, que se correspondería con la longitud de cada tramo. La distancia hasta el segundo waypoint sería el doble, hasta el tercero el triple … y así sucesivamente hasta determinar todos los waypoints necesarios.

De este modo se van añadiendo los waypoints de un extremo y otro de forma ordenada, teniendo en cuenta de si se trata de un tramo par o impar. Al llegar al último tramo, el correspondiente a los vértices 2 y 4, se comprueba también en qué orden deben añadirse, ya que el último waypoint se corresponderá con el vértice 4 si el número de divisiones es par, y con el vértice 2 en caso contrario. Una vez se tienen todos los waypoints necesarios para la misión

almacenados de forma ordenada, solo restará aplicar los resultados de la optimización para preparar la misión.

3.5.6 Configuración y preparación de la misión

Una vez reunidos todos los resultados generados en el proceso de optimización, solo resta emplearlos para configurar la misión, de tal forma que se realice el barrido de toda el área de forma óptima. En caso de no haberse podido optimizar el vuelo, sólo se mostrará el mensaje de advertencia y la configuración de la misión no se ejecutará.

Lo primero que se hace una vez finalizada la configuración es añadir a la misión los waypoints obtenidos y representarlos en el mapa, permitiendo observar de este modo el resultado obtenido. El siguiente paso es fijar la velocidad de la misión con la velocidad obtenida, así como la altura de vuelo. Una particularidad de *Waypoint Mission* es que el valor de la altura se debe fijar de forma independiente para cada waypoint, aunque este hecho no afectará al resultado ya que se fija la misma altura en todos ellos.

A la vez que se configura la altura, se aprovecha además para añadir acciones de rotación del gimbal al primer y último waypoint, con el objetivo de colocar la cámara en la posición adecuada. De este modo, se gira el gimbal para apuntar al suelo en el primer punto, mientras que, en el último punto, finalizada la misión, se devuelve a su posición por defecto, apuntando hacia el horizonte, para evitar así dañar el objetivo de la durante el aterrizaje. Como también se ha añadido la opción de modificar la altura de la acción *Go Home*, la altura de vuelta al punto de origen, antes de ejecutar la preparación de la misión se toma el valor que se haya introducido y se configura dicha altura.

En el siguiente paso, se ejecuta un método específico del SDK que prepara la misión para su ejecución, con todas las configuraciones y waypoints que se han aplicado. Si hubiese algún error, se notifica por pantalla y la misión no se podría ejecutar. Si no hay ningún fallo, se procede a configurar la cámara del dron para funcionar con un ratio de 4:3 y con el intervalo de tiempo entre fotos determinado en la optimización.

Por último, se comprueba si el número de satélites disponibles es igual o superior a 4, ya que con menos de 4 satélites la precisión en el posicionamiento se reduce de forma considerable. En ese caso, se advierte del riesgo existente, y, tanto en un caso como en otro, se finaliza todo el proceso de configuración y optimización de la misión, siendo necesario pulsar el botón “*Start*” para que esta dé comienzo.

3.6 Control con DJIFligthContrller, DJIMissionManager y DJIBattery

En esta aplicación hay tres partes del programa que se ejecutan de forma constante, tres métodos del SDK orientados a controlar el estado general del dron, de la misión y de la batería.

El primero de ellos es el proporcionado por el FlightController para recibir información de su estado 10 veces por segundo. Desde este método se puede acceder a la altura o velocidad actual del dron, así como al número de satélites disponibles o el modo de vuelo actual. Al mismo tiempo que se recibe y actualiza esta información, se muestran sus valores por pantalla en la barra superior de la aplicación, mostrada en la Figura 36. Una parte de los valores que se representan ya venían incluidos en la aplicación de ejemplo, mientras que los que se han añadido para cubrir las necesidades del proyecto son los resultados de la optimización (valores de velocidad y altura), el número de puntos añadidos (para detectar posibles puntos incluidos de forma accidental al rozar la pantalla), y el nivel de batería, que en realidad se actualiza en un punto diferente del programa.

Para controlar el estado actual de la misión, se ha añadido el delegado de DJIMissionManager, que permite hacer uso del segundo método empleado, que se puede ver en la Figura 35. En este método se controla en qué momento se ha llegado a un waypoint (*isWaypointReached*) y de cuál se trata, teniendo en cuenta una propiedad que indica a qué waypoint se está apuntando (*targetWaypointIndex*) para distinguirlos entre sí. De este modo, se ha desarrollado el código necesario para iniciar la captura de imágenes cuando se haya llegado al primer waypoint, y detener la captura en caso de haber alcanzado el último waypoint, que indica el final del barrido.

```
#pragma mark DJIMissionManagerDelegate
-(void)missionManager:(DJIMissionManager *)manager missionProgressStatus:(DJIMissionProgressStatus *)
    missionProgress
{

    // En el primer Waypoint, iniciar la captura de imágenes

    if([(DJIWaypointMissionStatus *)missionProgress isWaypointReached]==YES && [(DJIWaypointMissionStatus *)
        missionProgress targetWaypointIndex]==0)
    {
            __weak DJICamera* camera = [self fetchCamera];

            [camera startShootPhoto:DJICameraShootPhotoModeInterval withCompletion:^(NSError * _Nullable error
                ){
                if (error){
                    NSString* cameraError = [NSString stringWithFormat:@"Camera shot photo mode interval
                        failed:%@", error.description];
                    ShowMessage(@"", cameraError, nil, @"OK");
                }}];
    }

    // En el último Waypoint, detener la captura de imágenes

    if([(DJIWaypointMissionStatus *)missionProgress isWaypointReached]==YES && [(DJIWaypointMissionStatus *)
        missionProgress targetWaypointIndex]==(self.waypointMission.waypointCount-1))
    {
        __weak DJICamera* camera = [self fetchCamera];

        [camera stopShootPhotoWithCompletion:^(NSError * _Nullable error){
            if (error){
                NSString* cameraError = [NSString stringWithFormat:@"Camera Stop shot photo mode interval
                    failed:%@", error.description];
                ShowMessage(@"", cameraError, nil, @"OK");
            }}];
    }

}
```

Figura 35. Método de DJIMissionManager empleado para el control de la misión

Por último, se emplea un método relacionado con la batería que nos permite acceder de forma periódica a varias características relacionadas con su estado actual, siendo el porcentaje

de batería actual el dato de mayor interés para la aplicación. Es en este mismo método donde se actualiza el valor que se muestra por pantalla.

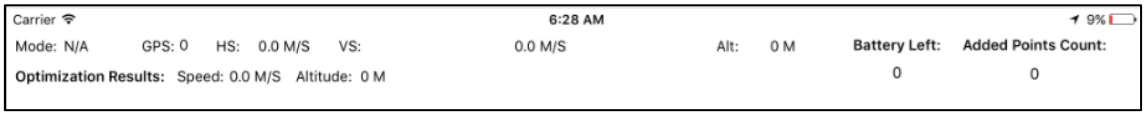


Figura 36. Barra de estado de la aplicación

3.7 Guía de uso de la aplicación

Hasta ahora se han presentado los detalles técnicos de la aplicación, los algoritmos empleados, la estructura de la programación, etc. No obstante, la aplicación será usada, en última instancia, por agricultores y personal cuyo único interés será ponerla en funcionamiento, sin necesidad de entrar en todos los detalles anteriores. Por este motivo, se presenta una guía de uso de la aplicación, de tal forma que cualquier persona sea capaz de emplearla independientemente de los conocimientos que posea.

El primer paso siempre será asegurarse que todos los dispositivos implicados, es decir, el control remoto, el dispositivo móvil y el dron cuenten con un nivel de batería apropiado, que permita poner en marcha la aplicación y ejecutarla. Una vez se encuentren todos estos elementos encendidos y correctamente conectados, se podrá ejecutar la aplicación.

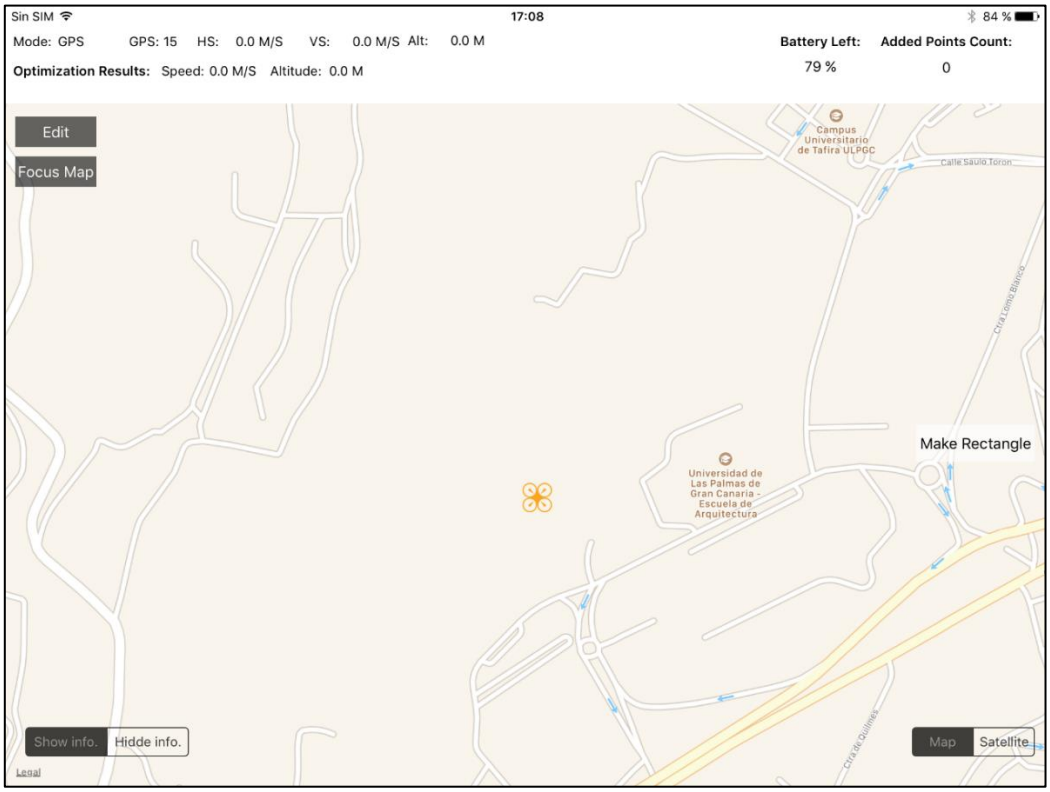


Figura 37. Pantalla de inicio

En la pantalla aparecerá un mapa del mundo, con la marca de una aeronave representando la posición del dron. En la Figura 37 se muestra el aspecto de la pantalla una vez se haya acercado a la posición actual. En la parte superior se encuentra la barra de estado con la información del dron, y en la inferior se sitúan dos botones de selección, uno para mostrar información de los puntos y otro para cambiar la vista del mapa.

A la izquierda se encuentran dos botones, *Edit* y *Focus Map*. Con el segundo botón, la vista se enfoca en la posición del dron y se centra en él, mientras que con el primer botón se realiza esta misma acción y, además, aparecen los distintos botones que permiten realizar todos los ajustes y configuraciones necesarias. En la Figura 38 se muestra el efecto de enfocar en el mapa haciendo uso del segundo botón. Además, puede verse el cambio que se produce al cambiar el tipo de mapa mediante el control situado en la esquina inferior derecha de la pantalla.

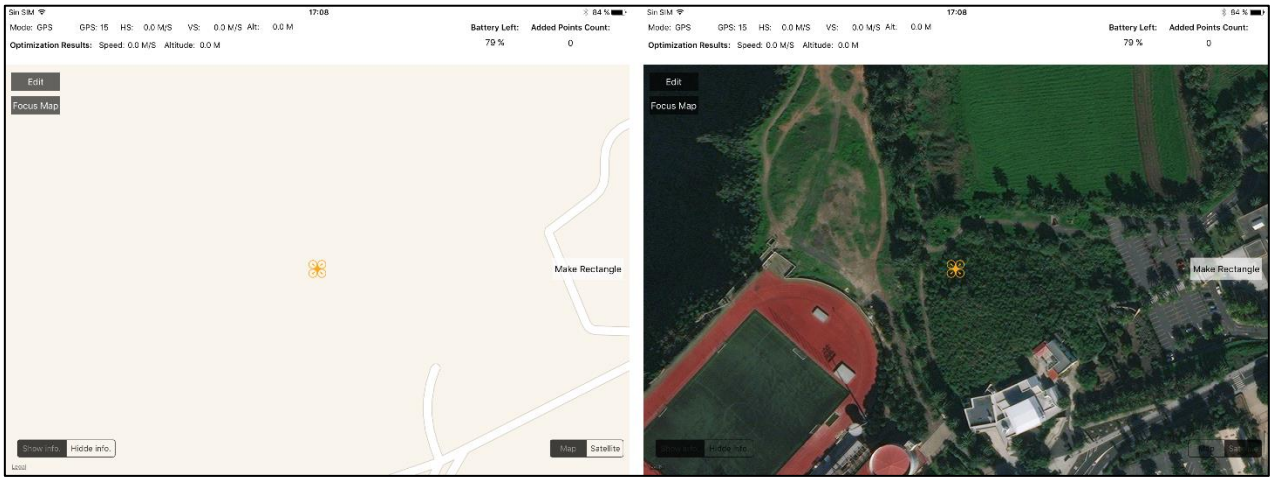


Figura 38. Vista enfocada de mapa (izquierda) y satélite (derecha)

Una vez se haya pulsado en el botón de *Edit*, los botones que se encuentran a la izquierda cambiarán. El siguiente paso será localizar el área a analizar y centrarla en el mapa, de tal forma que encaje dentro de un área rectangular. El campo de cultivo de la Figura 39 servirá de ejemplo a lo largo de esta guía para representar los pasos que se deben seguir.

Figura 39. Área agrícola ajustada en la aplicación

A la izquierda de la pantalla se aprecian los nuevos botones de la aplicación:

- **Back**: permite volver al estado anterior, ocultando los botones y volviendo a mostrar las opciones *Edit* y *Focus Map*
- **Add/Finish**: la aplicación entra/sale del modo edición, en el que se permite añadir puntos al programa.
- **Clear**: elimina todos los puntos que se hayan introducido.
- **Config**: abre la ventana de configuración
- **Start**: inicia el proceso de sobrevolar y analizar el área definida después de haber configurado la misión
- **Stop**: detiene el vuelo en curso.

Para definir el área, es necesario pulsar el botón *Add* y añadir sus vértices, ya sea tocando en el mapa o introduciendo las coordenadas con los controles que aparecen en la esquina superior derecha de la pantalla. Si se añaden únicamente dos vértices como se muestra en la parte izquierda de la Figura 40, se generarán los dos vértices restantes del rectángulo al pulsar el botón *Make Rectangle*, obteniéndose el resultado de la pantalla derecha. También pueden definirse los cuatro vértices de forma directa, pero es recomendable emplear el primer método siempre que sea posible. En cualquiera de los casos, si no se desea mostrar la información de cada punto como se ve en la figura, bastará con modificar el control situado en la esquina inferior izquierda.

Es importante tener en cuenta que el primer punto que se defina será el primer punto de todo el recorrido.

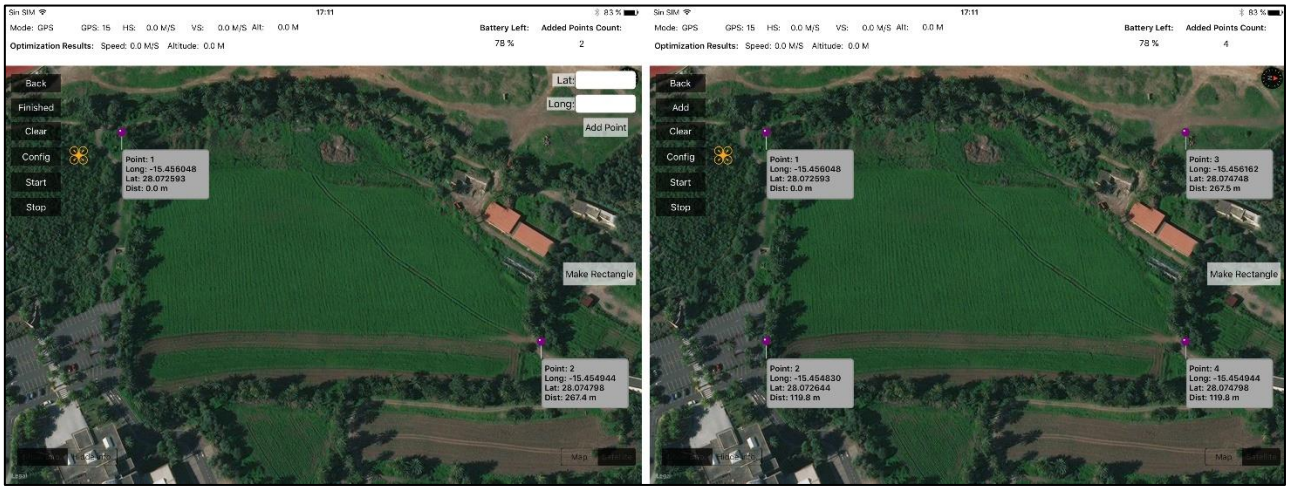


Figura 40. definición del área en la aplicación

Cuando se hayan definido los vértices, debe pulsarse en el botón *Finish* para evitar introducir cualquier punto no deseado de forma accidental. En cualquier caso, en la esquina superior derecha se muestra el número de puntos definidos (cuatro en el caso de la Figura anterior). A continuación, se accionará el botón de configuración, y aparecerá una ventana emergente (Figura 41) con las opciones de configuración. Estas opciones son:

- **Maximum/Minimum Altitude**: el intervalo de alturas entre las que se limitará el vuelo ya sea por la resolución de las imágenes o por las características del espacio aéreo (los valores admisibles son entre 20 y 120m respectivamente).
- **Image Overlap**: el nivel de solape entre imágenes, definido entre 0 y 1 (0 - 100%); obviamente, para obtener resultados válidos, no puede definirse un solape del 0% (no existiría solape) ni del 100% (la imagen sería siempre la misma), siendo los valores intermedios los que se deben emplear.
- **Máximum Speed**: velocidad límite de la aplicación.
- **Time Limit**: tiempo límite de vuelo; el valor por defecto viene dado por el nivel de batería restante, y cualquier valor introducido por encima de este no será válido.
- **Go Home Altitude**: Altura a la que el dron volverá al punto de inicio.
- **Action After Finished**: el comportamiento del dron una vez finalizada la misión. De izquierda a derecha, puede ser no hacer nada, volver al punto de origen, aterrizar, volver al primer waypoint o detenerse permitiendo volver a recorrer los waypoints.
- **Heading**: la orientación del dron, que podrá ser automática (en la dirección de vuelo), orientado al ponto inicial, al control remoto o hacia cada waypoint.

Si cualquiera de las configuraciones es incorrecta, al pulsar el botón *Finish* aparecerá un mensaje de error que advertirá de dicha situación, indicando qué parámetros son incorrectos y cuáles son los valores admitidos. Respecto a las dos últimas opciones, se recomienda mantener sus valores por defecto, es decir, volver al punto de inicio como acción final y mantener la orientación automática.

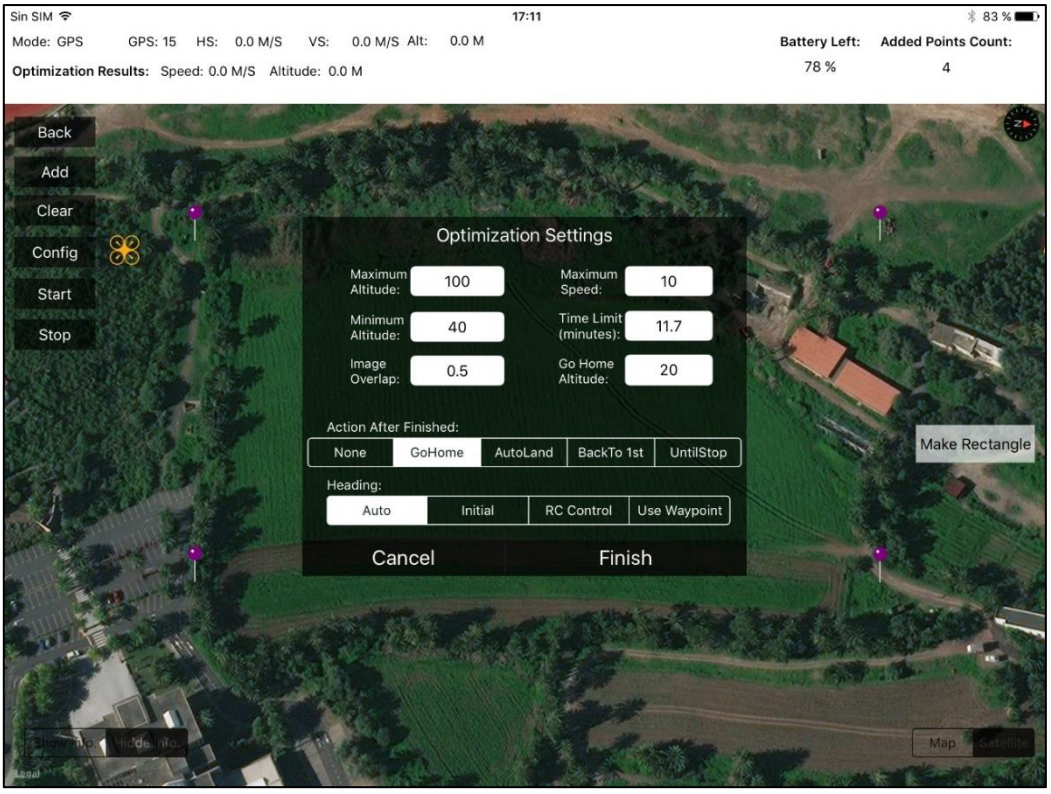


Figura 41. Opciones de configuración

Finalizada la configuración, la aplicación procesará los datos introducidos y determinará la trayectoria a seguir. Si no hay ningún error, en la pantalla aparecerán el mensaje de la Figura 42 y los waypoints intermedios que conformarán el recorrido del vuelo, actualizándose el número de waypoints añadidos a la derecha de la barra de estado. También se actualizarán los valores de la altura y velocidad a la que se llevará a cabo el vuelo en la parte izquierda de la barra de estado (en este caso, 40 metros y 8 m/s).

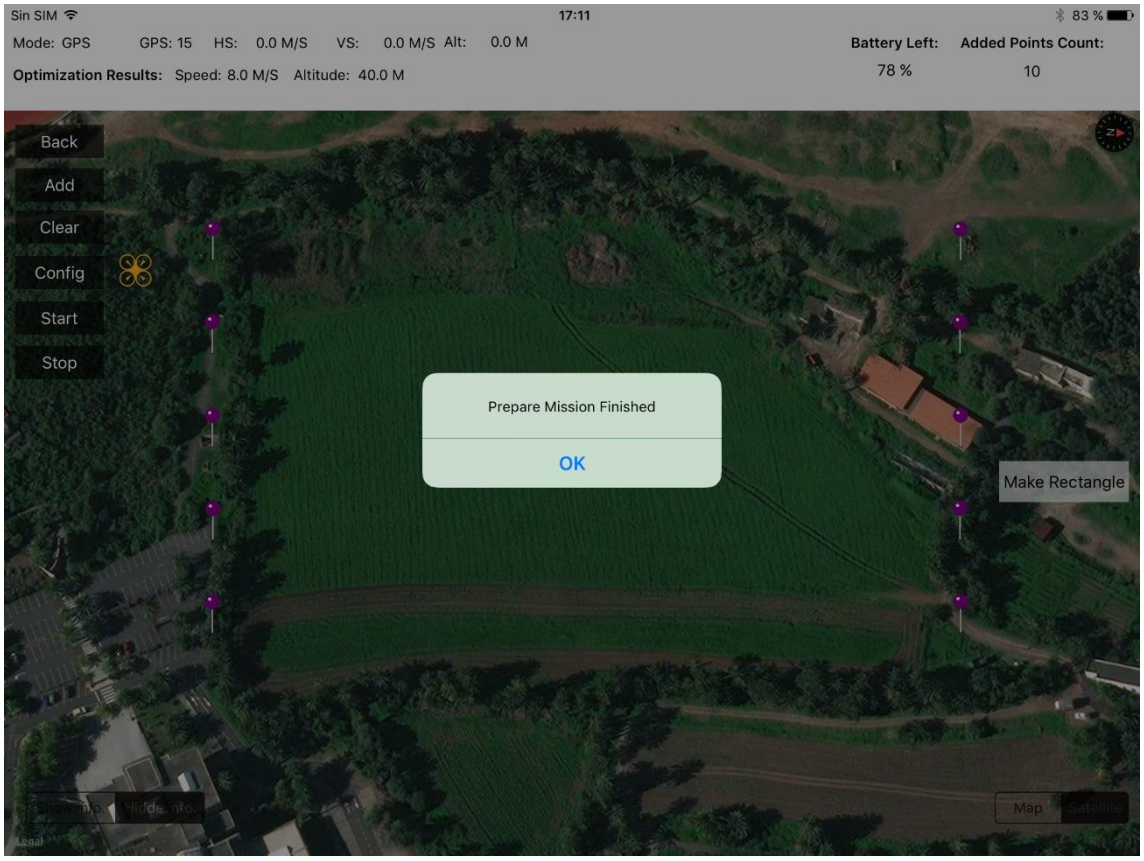


Figura 42. Configuración finalizada

Finalmente, solo restará pulsar en el botón *Start* y el dron despegará y recorrerá el área que se ha definido, volviendo al punto de inicio una vez terminado el proceso. Si en algún punto se desea detener el vuelo, bastará con pulsar el botón de *Stop* y el dron se quedará fijo en esa posición.

Capítulo 4. Resultados obtenidos y conclusiones

En este capítulo se recogen los resultados que se han obtenido tras el desarrollo de esta aplicación. Por un lado, se ha comprobado la respuesta de la optimización para distintos niveles de batería para una misma área. Por otro lado, se ha realizado una prueba con el dron en un campo de fútbol, capturando imágenes del mismo y componiéndolas con la herramienta OpenDroneMap.

Más adelante, en este mismo capítulo, se muestra el resultado de la ejecución del código en el simulador usando el Matrice 600. Finalmente, se presentan las conclusiones de este trabajo y algunas líneas futuras para el mismo.

4.1 Resultados del bucle de optimización

Para comprobar el funcionamiento del bucle de optimización, se ha definido el área que se muestra en la Figura 43, con unas dimensiones de 401.6 x 456.7 metros. Primero, se ha optimizado el vuelo sobre esta área con un ajuste de 5 m en el bucle de optimización (la altura se decide en saltos de 5 metros), empezando con la batería totalmente cargada y repitiendo el proceso de optimización con niveles de carga cada vez menores. Posteriormente, se ha modificado el código para aplicar un ajuste de 1 m y se ha repetido el proceso, volviendo a partir de la batería totalmente cargada.

Dado que los resultados solo serán concluyentes si el área definida es la misma en todos los casos, se ha modificado el código para esta prueba para definir como parámetros fijos sus coordenadas, garantizando así que la optimización se ejecuta siempre con las mismas condiciones para todos los niveles de batería.

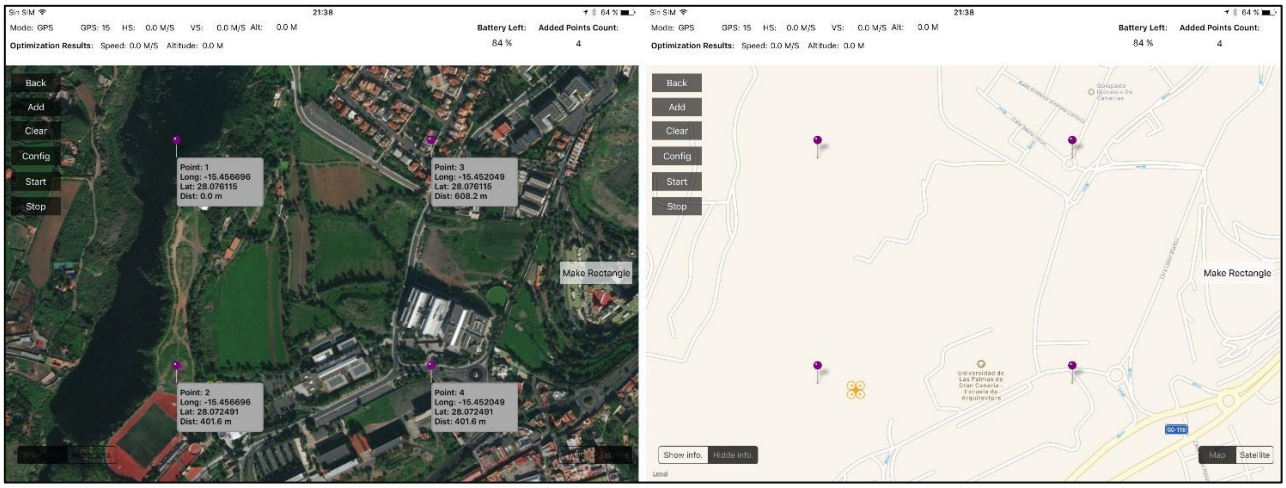


Figura 43. Área definida para la obtención de resultados

Para todos los casos se ha aplicado la misma configuración, con los límites de altura máximo y mínimo fijados en 120 y 20 metros respectivamente, un solape entre imágenes de 0.5

(50%) y una velocidad máxima de 10 m/s. En las gráficas que se muestran a continuación se representan los resultados obtenidos en ambos casos.

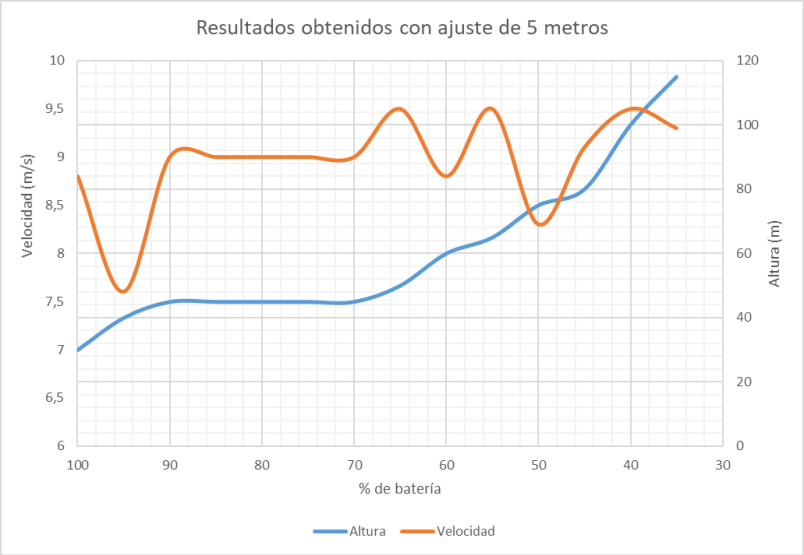


Figura 44. Resultados de optimización (ajuste de 5m)

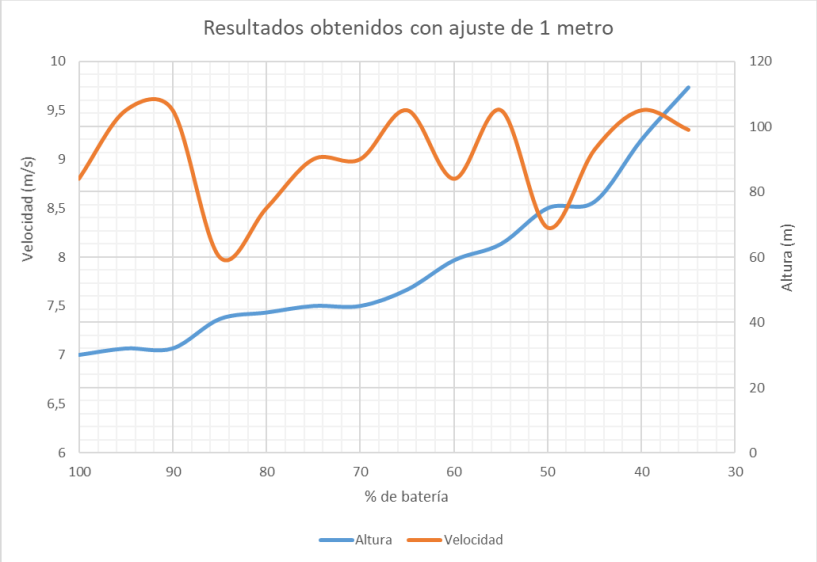


Figura 45. Resultados de optimización (ajuste de 1m)

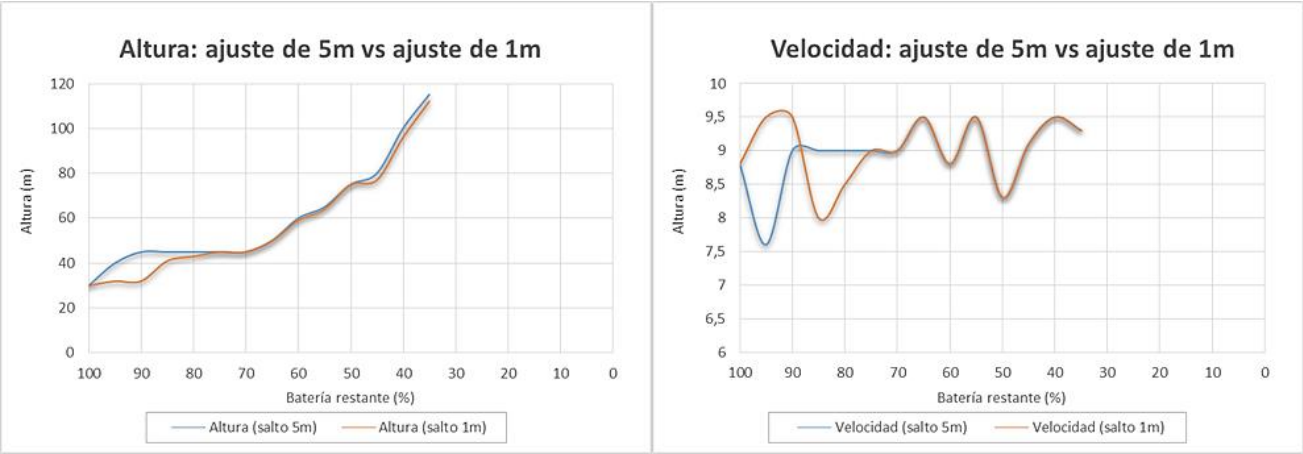


Figura 46. Comparativa de resultados de optimización

En general, puede apreciarse como la altura de vuelo que se fija para cada caso va aumentando a medida que se reduce el nivel de batería. La altura de vuelo es la mínima posible para la configuración tomada, ya que tal y como se explicó en los anteriores capítulos, se trata de obtener la mayor resolución posible (que implica volar a la mínima altura posible) al barrer un área con el dron, teniendo en cuenta las limitaciones de tiempo impuestas por el nivel de batería del mismo.

En el caso de la velocidad, se observa como esta fluctúa para los distintos niveles de batería, pero manteniéndose siempre cercana al valor máximo. Esta fluctuación se explica debido a las condiciones del problema: la distancia que debe haber entre cada foto tiene un valor fijo, ya que debe garantizarse el nivel de solape entre las mismas, y el intervalo de tiempo entre cada foto está limitado a valores enteros, siendo el mínimo de 2 segundos. Por lo tanto, la velocidad dependerá de estos valores, tal y como se muestra a continuación, y podrá ser mayor o menor dependiendo de las condiciones particulares del problema a optimizar.

$$Velocidad = \frac{Distancia\ entre\ fotos\ (valor\ fijo)}{Tiempo\ entre\ fotos\ (variable\ en\ incrementos\ d\ 1s)}$$

Si comparamos los resultados obtenidos en cada caso, puede observarse que, al disminuir el salto entre las distintas alturas en la optimización, se aplica un ajuste más fino y se obtienen mejores resultados. Esta mejora se aprecia de forma especialmente significativa en el tramo comprendido entre el 100% y el 80% de batería, donde las diferencias obtenidas son mayores.

Estas diferencias pueden observarse también en la Tabla 2, que recoge de forma más detallada los valores exactos que se han obtenido en cada caso, particularmente en los casos del 95% y 90% de batería, donde las diferencias entre ambos casos alcanzan los 13 metros. Esta diferencia sorprende porque, en primera instancia, la máxima diferencia que cabría esperar es de 4 metros al ser los ajustes de 1 y 5 metros, pero la realidad es que las diferencias pueden ser mayores, tal y como se ha visto.

Para explicar estas diferencias, se tomará como ejemplo el caso de tener un 95% de batería. Al optimizar el vuelo con un ajuste de 1 metro se obtiene una altura de 32m; la altura que se esperaría obtener con el ajuste de 5 metros es 35m, ya que es el valor mínimo a estudiar por encima de los 32m obtenidos con el primer ajuste. Sin embargo, el resultado final es de 40m, y no 35.

Estudiando los valores que se manejan durante la optimización, la distancia entre fotos a 35m de altura debe ser de 20,76 metros. Con el mínimo intervalo de tiempo (2s), la velocidad que se debe mantener es de 10,38 m/s, mayor que el límite de 10 m/s establecido, por lo que debe tomarse un intervalo mayor, es decir, 3s. Con este tiempo, la velocidad pasa a ser de 6,2m/s, una velocidad válida, pero que resulta ser insuficiente para recorrer toda el área sin exceder el límite de tiempo. Por este motivo, la altura se sigue aumentando, empeorando así la calidad del resultado para no superar el límite de velocidad. En este caso en particular, la diferencia entre la velocidad límite (10 m/s) y la primera velocidad calculada (10.38 m/s) no es excesiva, pero en cualquier caso debe evitarse superar el límite de velocidad fijado (además, en otros casos esta diferencia será mayor y no podrá ser ignorada).

De este modo, se demuestra que las diferencias obtenidas con los distintos ajustes pueden ser mayores de lo esperado, y se comprueba la eficacia de emplear un ajuste más preciso.

Con respecto a la velocidad, las fluctuaciones obtenidas se mantienen en ambos casos a pesar de variar el ajuste, tal y como se ha explicado con anterioridad

Tabla 2. Resultados obtenidos para un área de 401,6 x 456,7 metros

RESULTADOS OBTENIDOS				
% de batería	Ajuste de 5 m		Ajuste de 1m	
	Altura	Velocidad	Altura	Velocidad
100	30	8,8	30	8,8
95	40	7,6	32	9,5
90	45	9	32	9,5
85	45	9	41	8
80	45	9	43	8,5
75	45	9	45	9
70	45	9	45	9
65	50	9,5	50	9,5
60	60	8,8	59	8,8
55	65	9,5	64	9,5
50	75	8,3	75	8,3
45	80	9,1	77	9,1
40	100	9,5	96	9,5
35	115	9,3	112	9,3
30	No es posible realizar el vuelo con este nivel de batería			

4.2 Ejecución de optimización y vuelo en el simulador

Con el fin de comprobar la eficacia de la aplicación a la hora de llevar a cabo el vuelo, se realiza una prueba ejecutando la aplicación en modo simulación, con el dron conectado al equipo. En la aplicación se define un área idéntica a la empleada en el apartado anterior, con la configuración de la Figura 47, donde se pueden observar además los puntos obtenidos.

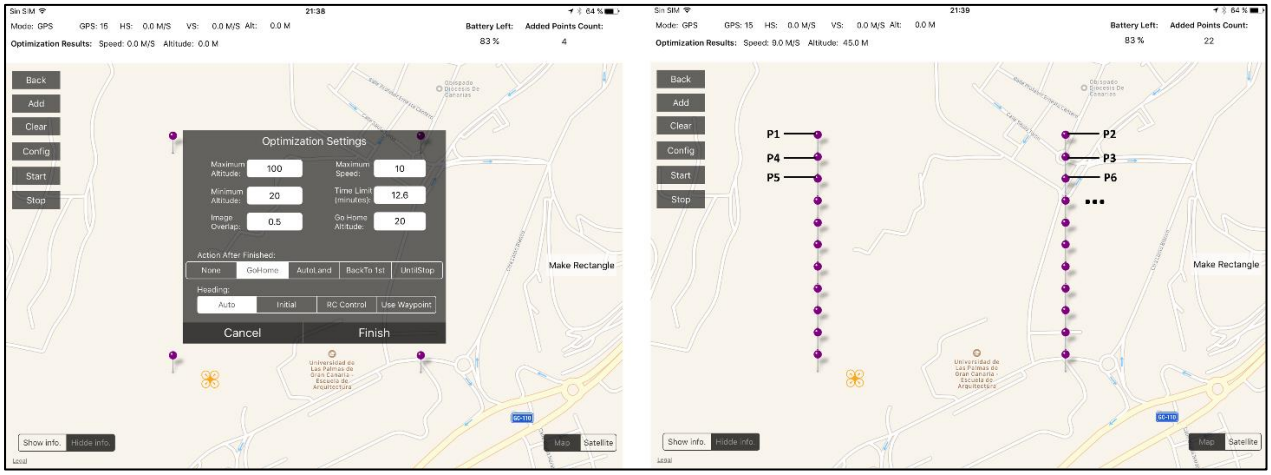


Figura 47. Vuelo para la comprobación en simulador

Una vez preparado el recorrido, obteniendo una altura de 45 metros y una velocidad de 9 m/s (que, con un 83% de batería, es acorde con los resultados del apartado anterior), se inicia el vuelo sobre el área. A medida que pasa el tiempo, se puede observar como, tanto en el simulador como en la pantalla del dispositivo, el dron sigue el recorrido programado, como se muestra en la Figura 48, demostrando así la validez de la aplicación desarrollada.

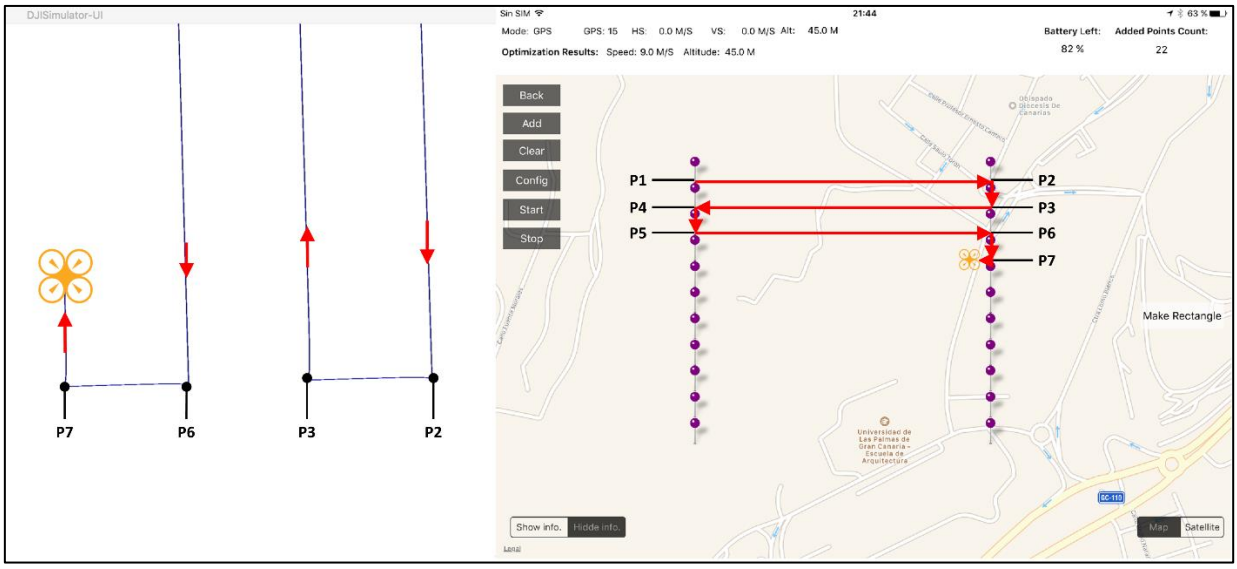


Figura 48. Ejecución final en simulador

La trayectoria del vuelo en el simulador solo muestra una sección de la trayectoria, ya que la ventana del mismo no permite abarcarla en su totalidad, dadas las dimensiones del área analizada.

4.3 Adquisición de imágenes y composición con OpenDroneMap

En esta sección se muestran los resultados de una prueba llevada a cabo al aire libre. En esta prueba, se ha llevado el dron al campo de fútbol de la Universidad de Las Palmas de Gran Canaria, tal y como se muestra en la Figura 49, y se ha preparado para analizarlo haciendo uso de la aplicación. Para ello, se define el área del campo como área a optimizar, fijando un solape del 70%. La altura mínima se ha fijado a 50m, de tal forma que, al finalizar el bucle de optimización, la altura obtenida es precisamente de 50 metros. Se ha elegido este valor de 50 metros por varios motivos: por un lado, para evitar ciertos obstáculos presentes en el campo de futbol (redes, gradas, postes de luz), y por otro lado, para reducir el número de imágenes obtenidas, ya que el software a utilizar para la composición de las imágenes, OpenDroneMap, requiere un tiempo de cómputo considerable.

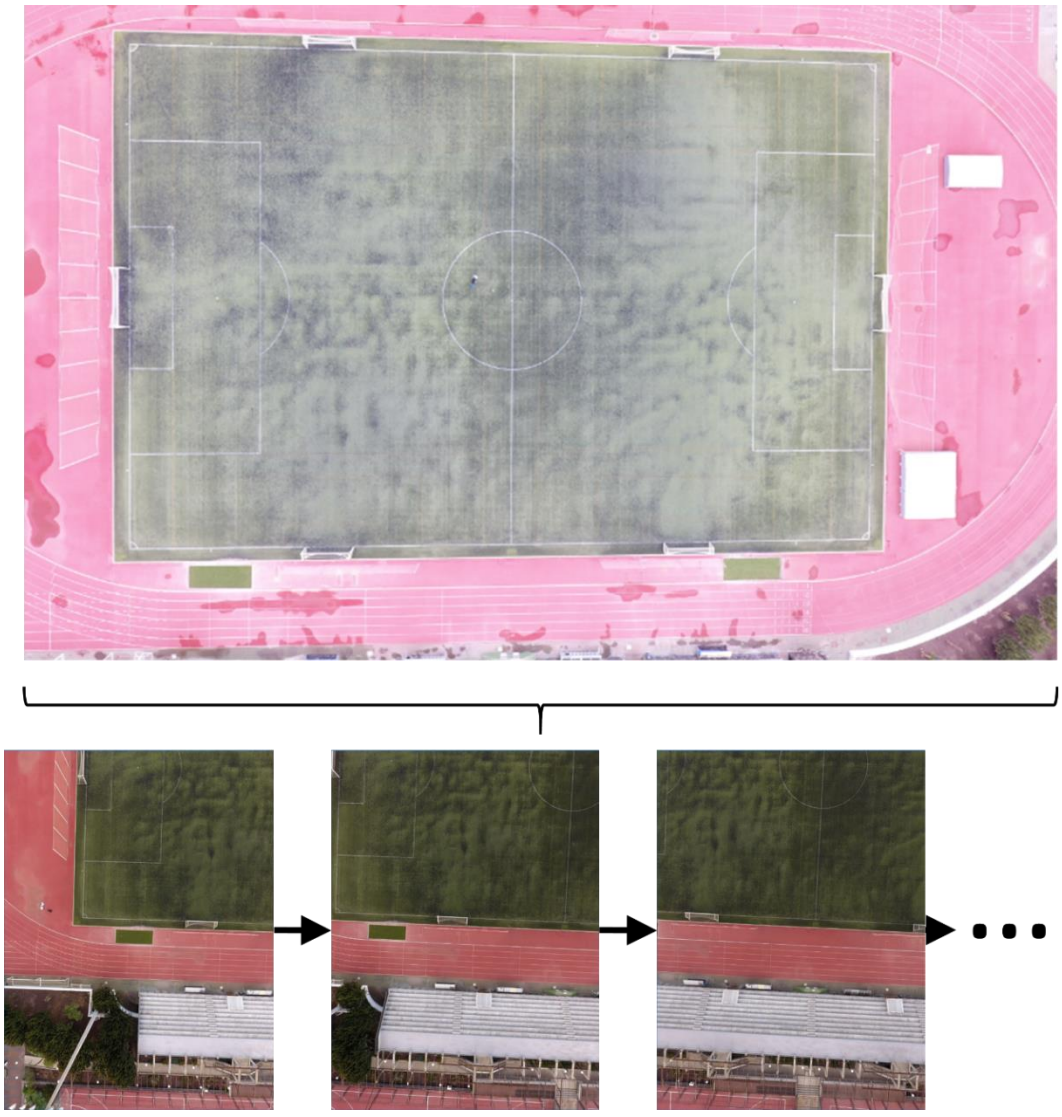


Figura 49. Campo de fútbol a analizar e imágenes capturadas

De este modo, el dron sobrevuela el campo capturando imágenes del mismo, tal y como se muestra en la figura anterior, con un solape del 70% entre ellas. Finalizado el proceso, se ha seleccionado una parte de las imágenes generadas y se han procesado con el software OpenDroneMap. El resultado final (Figura 50) es una única imagen con una gran resolución espacial que abarca toda el área procesada.

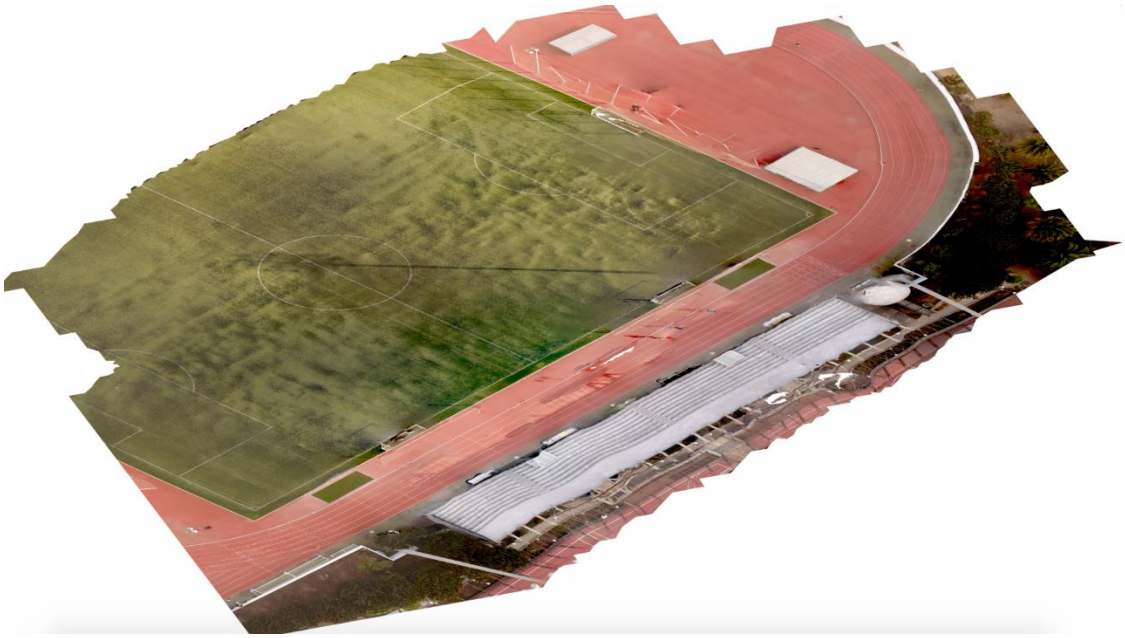


Figura 50. Composición de imágenes capturadas

Como con los distintos resultados que da el software se puede obtener la imagen compuesta, ortorectificada y georreferenciada, el último paso es emplear Google Earth para superponer el resultado obtenido sobre la vista de satélite y contrastarlas. Como se puede observar en la Figura 51, la imagen se sitúa en las coordenadas correctas, coincidiendo la línea central del campo a la perfección, confirmando así la validez del resultado.

Sin embargo, las líneas de los extremos del campo no coinciden entre sí. Esto se debe a que, al realizar la composición, el resultado en los bordes siempre presenta imperfecciones. La solución a este fallo es definir un área ligeramente mayor al área de interés, de tal forma que las imperfecciones de los bordes no afecten a la misma.

Figura 51. Resultado superpuesto sobre Google Earth

4.4 Exportación a Matrice 600

Uno de los objetivos de este TFM era conseguir que la aplicación desarrollada fuese exportable al Matrice 600 de Dji. Para comprobar que se cumple esta característica, se ha creado una nueva versión del código donde se han eliminado las partes que hacían uso de los elementos característicos del Phantom 4, como por ejemplo la cámara.

Debido a las características de la cámara hiperespectral que llevará el Matrice 600, que al ser de tipo push-broom no capturará imágenes completas, sino líneas de pixeles, el bucle de optimización funciona de forma diferente. En este caso, la velocidad será fija y no un resultado del bucle de optimización, además de no emplearse ningún tipo de intervalo entre fotos. De cualquier modo, esta nueva versión de la aplicación solo se ha creado para corroborar la compatibilidad de la aplicación con el Matrice 600, por lo que la optimización deberá mejorarse y pulirse en las futuras fases del proyecto, atendiendo a las particularidades de la tecnología hiperespectral.

En la Figura 52 se encuentra el área definida en la aplicación para ejecutar el vuelo empleando el Matrice 600 y el simulador. El dron ha respondido de forma satisfactoria, siguiendo el recorrido establecido sin ningún tipo de error. En la Figura 53 pueden observarse el montaje empleado, con el simulador y la aplicación funcionando correctamente.

Figura 52. Área definida para la prueba con el Matrice 600

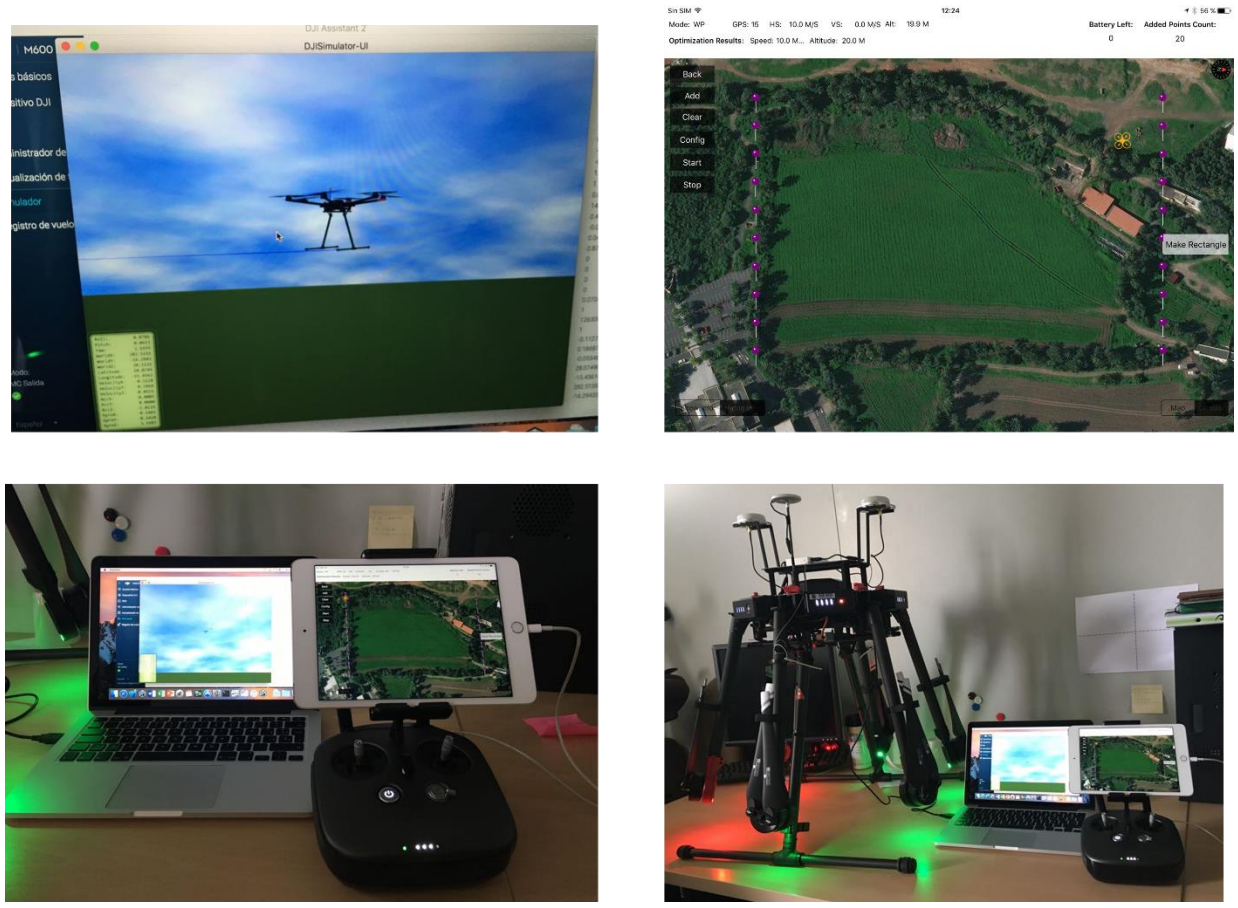


Figura 53. Aplicación funcionando con Matrice 600

4.5 Conclusiones y trabajos futuros

En este apartado se presentan las conclusiones de este Trabajo de Fin de Máster, una vez finalizado el desarrollo de la aplicación y obtenidos los resultados pertinentes. En el primer capítulo de este documento se presentaban los objetivos a alcanzar al término del presente trabajo, objetivos que se han cumplido en su totalidad, ya que:

- ✓ Se ha desarrollado una aplicación en Objective-C capaz de optimizar el vuelo de un dron sobre un área determinada, partiendo de una aplicación de ejemplo y haciendo uso del Mobile SDK de Dji. Esto se ha podido comprobar en los diferentes resultados obtenidos.
- ✓ La interfaz de usuario de la aplicación permite definir el área a explorar de forma sencilla, ya sea de forma directa al pulsar sobre el mapa o introduciendo los valores exactos de las coordenadas.
- ✓ El proceso de optimización tiene en cuenta los valores de altura, velocidad y solape introducidos, así como el FOV de la cámara y el nivel de batería restante, obteniendo resultados coherentes y satisfactorios.

- ✓ Se capturan imágenes del área a analizar de forma automática, respetando el nivel de solape especificado y obteniéndose el resultado esperado, como se ha podido comprobar gracias al software de OpenDroneMap.
- ✓ La programación se ha podido exportar al Matrice 600 y se ha corroborado su funcionamiento con el mismo.

Para seguir avanzando con los objetivos propuestos en el proyecto ENABLE-S3, a partir de este TFM se proponen una serie de líneas de trabajo para el futuro:

- Progresar con la implementación de la aplicación en el Matrice 600, adaptando la optimización al mismo, aprovechando las características de este para mejorar el rendimiento de la aplicación y teniendo en cuenta sus particularidades. Es preciso implementar la comunicación entre la aplicación desarrollada con el Mobile SDK que se ejecuta en la Tablet, y el PC on board que se montará en el dron, para así coordinar las operaciones de captura de las imágenes con el transcurso de la misión.
- Profundizar en el proceso de composición de las imágenes, buscando mejorar los resultados obtenidos y acelerar el proceso en la medida de lo posible. Estudiar la posibilidad de extrapolar este proceso a otras aplicaciones más complejas, como por ejemplo el uso de escuadras de drones que actúen de forma conjunta para abarcar un área mayor.
- Incorporar la tecnología hiperespectral en el proceso, llevando a cabo el montaje de la cámara hiperespectral en el Matrice 600, buscando las soluciones necesarias a los problemas que esto conlleva: la estabilización de la cámara, la comunicación entre cámara y dron y el control de la misma, además del hecho de ser una cámara push-broom que captura una única línea de píxeles.
- Ajustar en tiempo real los parámetros de altura, velocidad e intervalo de captura de imágenes, según la lectura del nivel de batería restante durante el vuelo, ya que esta puede se puede descargar de forma irregular debido a las condiciones meteorológicas (frio y/o viento) o a posible malfuncionamiento de la misma.

Referencias

[1] "Introduction to hyperspectral imaging", en www.microimages.com/documentation/Tutorials, última visita el 08.06.2017

[2] H. Fabelo, S. Ortega, S. Kabwama, G. M Callico, D. Bulters, A. Szolna, J. F. Pineiro, R. Sarmiento, "HELICOID Project: a new use of hyperspectral imaging foir brain cáncer detection in real-time during neurosurgical operations", SPIE Commercial+ Scientific Sensing and Imaging, Baltimore, EEUU, 17-21 abril 2016

[3] S Ortega, H Fabelo, R Camacho, ML Plaza, GM Callico, R Lazcano, D Madroñal, R Salvador, E Juárez, R Sarmiento, "Detection of human brain cáncer in pathological slides using hyperspectral images", Neuro-oncology, vol. 19, mayo 2017

[4] L. Santos, L. Berrojo, J. Moreno, J. F. López, R. Sarmiento, "Multispectral and hyperspectral lossless compressor for space applications (HyLoC): a low complexity FPGA implementation of the CCSDS 123 standard", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 9, num. 2, enero 2015

[5] A. García, L. Santos, S. López, G. M. Callicó, J. F. López, R. Sarmiento, "Efficient lossy compression implementations of hyperspectral images: tolos, hardware platforms and comparisons", SPIE Sensing Technology + Applications, Baltimorer, EEUU, 5-9 mayo 2014

[6] M. Teke, H.S. Deveci, O. Haliloglu, S.Z. Gurbuz, U. Sakarya, "A short survey of hyperspectral remote sensing applications in agriculture", 6th International Conference on Recent Advances in Space Technologies (RAST), Estambul, Turquía, 12-14 junio 2013

[7] R.N. Sahoo, S.S. Ray, K.R. Manjunath, "Hyperspectral remote sensing of agriculture", Current Science, vol. 108, num. 5, marzo 2015

[8] J. Pedraza, "La revolución que nos dará de comer (y cuidará el planeta)", Periódico El País, 21 abril, 2017

[9] https://developer.dji.com/mobile-sdk, consultado por última vez el 27 de junio de 2017

[10] https://developer.apple.com/xcode, consultado por última vez el 27 de junio de 2017

[11] http://opendronemap.org, consultado por última vez el 27 de junio de 2017

[12] https://support.pix4d.com/hc/en-us/articles/20https://support.pix4d.com/hc/en-us/articles/202557459#label32557459#label3, consultado por última vez el 3 de Julio de 2017

[13] https://dronemapper.com/software/DroneMapper-Instructions.pdf, consultado por última vez el 3 de Julio de 2017

[14] https://developer.dji.com/mobile-sdk/documentation/introduction/camera_concepts.html, consultado por última vez el 27 de junio de 2017