



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

Desarrollo de una Aplicación de Análisis y
Visualización de Vuelos de Drones

Autor: Álvaro Lozano Gil
Tutor(a): Vicente Martínez Orga

Madrid, enero 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Desarrollo de una Aplicación de Análisis y Visualización de Vuelos de Drones

Enero 2024

Autor: Álvaro Lozano Gil

Tutor:

Vicente Martínez Orga
Departamento de Ingeniería Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Dedicatoria

Quiero dedicar este trabajo a mi familia y amigos, por su apoyo incondicional en las malas, las peores y las terribles, y en las buenas, las mejores y las superiores.

Resumen

Hoy en día existe una gran oferta de aplicaciones para la planificación de vuelos de drones, pero hay una carencia de herramientas que permitan evaluar y analizar los datos de esos vuelos una vez realizados.

Situaciones comunes como desconexiones, fuertes vientos, falta de almacenamiento o bugs en el software, pueden comprometer una sesión de trabajo si los pilotos no tienen la capacidad de analizar las misiones voladas hasta después de volver de la operación en campo. En muchos casos, esto se traduce en la necesidad de repetir misiones de vuelo, implicando un coste adicional de tiempo y recursos.

Este trabajo de fin de grado trata de abordar este problema, proponiendo el diseño y desarrollo de una aplicación de visualización y análisis de vuelos de drones, con el principal objetivo de facilitar la inspección de los datos de vuelo, mediante un enfoque visual que permita una compresión rápida e intuitiva de los mismos y que permita identificar posibles problemas o anomalías y optimizar futuros vuelos.

Para ello, usando Python y el framework Streamlit, además de diversas librerías externas, se ha desarrollado una aplicación web especialmente diseñada para trabajar con el software de planificación de vuelos PIX4Dcapture Pro. La aplicación extrae y procesa datos de vuelo como la velocidad, posición, orientación o batería del dron, tanto de imágenes capturadas como de logs, para ofrecer diversos análisis numéricos y visualizaciones tridimensionales interactivas.

Debido a la posible complejidad de estos análisis, la aplicación busca tener una interfaz gráfica simple que prioriza la eficiencia y la usabilidad, procurando que la mayor parte de las funciones estén disponibles a tan solo unos pocos clics.

Abstract

Nowadays, there is a wide range of applications for drone flight planning available, but there is a lack of tools for evaluating and analysing the data from these flights once they have been performed.

Common situations such as disconnections, strong winds, lack of storage or software bugs can compromise a work session if pilots do not have the ability to analyse the missions flown until after returning from the field operation. In many cases, this results in the need to repeat flights, incurring additional time and resource costs.

This final degree project tries to address this problem, by proposing the design and development of an application for the visualization and analysis of drone flights, with the main objective of facilitating the inspection of flight data, through a visual approach that gives a quick and intuitive understanding of the data and that allows to identify possible problems or anomalies and optimize future flights.

To achieve this, using Python and the Streamlit framework, as well as several external libraries, a web application especially designed to work with the PIX4Dcapture Pro flight planning software has been developed. The application extracts and processes flight data such as speed, position, orientation or battery of the drone, both from captured images and logs, to provide various numerical analyses and interactive three-dimensional visualizations.

Due to the possible complexity of these analyses, the application seeks to have a simple graphical interface that prioritizes efficiency and usability, ensuring that most functions are available within just a few clicks.

Tabla de contenidos

1	Introducción	1
2	Trabajos Previos y Estado del Arte.....	3
3	Tecnologías empleadas	6
3.1	Lenguaje de programación: Python	6
3.2	Librerías externas	6
3.3	Framework: Streamlit	9
3.4	Otros	9
4	Desarrollo.....	11
4.1	Identidad de la aplicación	11
4.2	Estructura del proyecto	13
4.3	Extracción de datos	14
4.3.1	Imágenes	15
4.3.2	Logs.....	22
4.4	Selección de vuelo.....	27
4.4.1	Imágenes	29
4.4.2	Logs.....	35
4.5	Análisis básico.....	37
4.6	Visor interactivo.....	38
4.6.1	Imágenes	40
4.6.2	Logs.....	54
4.6.3	Ejemplos de uso	65
4.7	Gráficos	70
4.7.1	Imágenes	71
4.7.2	Logs.....	76
4.7.3	Personalizados.....	79
4.8	Análisis estadístico	80
4.8.1	Básico	80
4.8.2	Avanzado.....	81
4.9	Instalación.....	85
5	Resultados y conclusiones.....	90
6	Análisis de impacto	92
7	Bibliografía.....	94

1 Introducción

La industria de los drones ha experimentado un crecimiento exponencial en los últimos años, emergiendo como una herramienta vital en una variedad de sectores [1].

En la agricultura, los drones están revolucionando las prácticas agrícolas a través de la cartografía de precisión, inspección por infrarrojos, fumigación y gestión de cultivos, lo que permite una agricultura más eficiente y sostenible. En 2022, se estimó el valor global de la industria de drones dedicada a la agricultura en 3.69 miles de millones de dólares [2]. En el ámbito forestal, facilitan la vigilancia y el manejo de los recursos naturales, contribuyendo a la conservación y al combate de incendios forestales. En la topografía, ofrecen una forma rápida y precisa de recabar datos del terreno, lo que es esencial para la planificación y ejecución de proyectos a gran escala.

La inspección de infraestructuras se ha transformado con el uso de drones, proporcionando una manera económica (por ejemplo, el envío de un helicóptero o un barco para inspeccionar un dique antihielo cuesta más de 10 veces lo que enviar un dron [2]) y segura (ya que los pilotos pueden hacer la inspección desde una zona alejada de las estructuras, lo que puede salvar vidas [3]) y económica de evaluar puentes, carreteras, edificios. En el sector de los servicios públicos, los drones son fundamentales para el mantenimiento y la inspección de líneas eléctricas, torres de comunicación, tanques de almacenamiento y otras infraestructuras críticas. En la construcción, ofrecen la posibilidad de cuantificar y comparar el progreso de las obras con objetivos y planos, y ayudan en la logística y la seguridad del sitio, reduciendo hasta en un 91% los accidentes [4]. La minería y las construcciones subterráneas también se benefician del uso de drones para la exploración de minerales y la gestión de minas, gracias a la tecnología LiDAR que puede funcionar en condiciones de poca luz.

Además, los drones están transformando el transporte de suministros esenciales a áreas remotas y en emergencias, superando obstáculos geográficos para entregar ayuda y medicamentos de forma rápida. En desastres humanitarios, son clave para la evaluación de daños, la búsqueda de sobrevivientes y la planificación de rescates, ahorrando tiempo y salvando vidas.

La versatilidad y la capacidad de innovación de los drones en todo tipo de industrias continúan abriendo nuevos caminos, lo que indica que su impacto en estas industrias seguirá creciendo en los años venideros, potenciado además localmente por la financiación estatal y europea [5]. Según la comisión europea se prevé que para 2030 alcance un valor de 14.5 miles de millones de euros, con un crecimiento del 12.3% anual y emplee a 145,000 ciudadanos de la Unión Europea [6]. Para la próxima década, se estima que el valor total de la industria completa de los drones se sitúe en 260.5 miles de millones de dólares [7], con algunos estudios estimando cifras tan altas como 699.8 miles de millones de dólares para 2032 [8].

En este contexto y ante el inminente volumen de operaciones con drones, se requiere la creación de herramientas avanzadas y versátiles para la revisión y análisis post-vuelo. Actualmente, la industria de los drones dispone de numerosas soluciones para la planificación y automatización de vuelos, pero adolece de un déficit en aplicaciones dedicadas a la evaluación detallada de los vuelos ya completados. Es esencial, especialmente en sectores como la fotogrametría y otros campos de aplicación, contar con un software capaz de analizar exhaustivamente

cada misión para evitar la repetición de vuelos por errores no detectados a tiempo, como pérdida de conexión, condiciones meteorológicas adversas, problemas de almacenamiento o fallos de software.

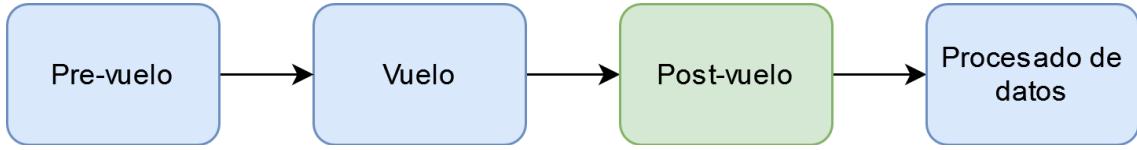


Figura 1.1: Posición en el flujo de trabajo de la futura aplicación. Si el vuelo va mal el procesado de datos sería inútil de no hacerse un análisis post-vuelo.

El proyecto que se propone aspira a llenar este vacío con el desarrollo de una aplicación para analizar y visualizar vuelos de drones automatizados y llevados a cabo principalmente con la aplicación de Android e iOS PIX4Dcapture Pro [9], pero también dando soporte a otras aplicaciones aunque a un nivel más limitado, y dando soporte a drones de DJI y Parrot, dos de los fabricantes más importantes del mundo [10] con la meta de brindar un entorno de análisis con visualizaciones interactivas y una interfaz intuitiva que simplifique la identificación de anomalías y valide la conformidad del vuelo con los parámetros indicados, logrando así una mayor eficiencia y efectividad en las operaciones con drones y optimizando el flujo de trabajo de los pilotos.

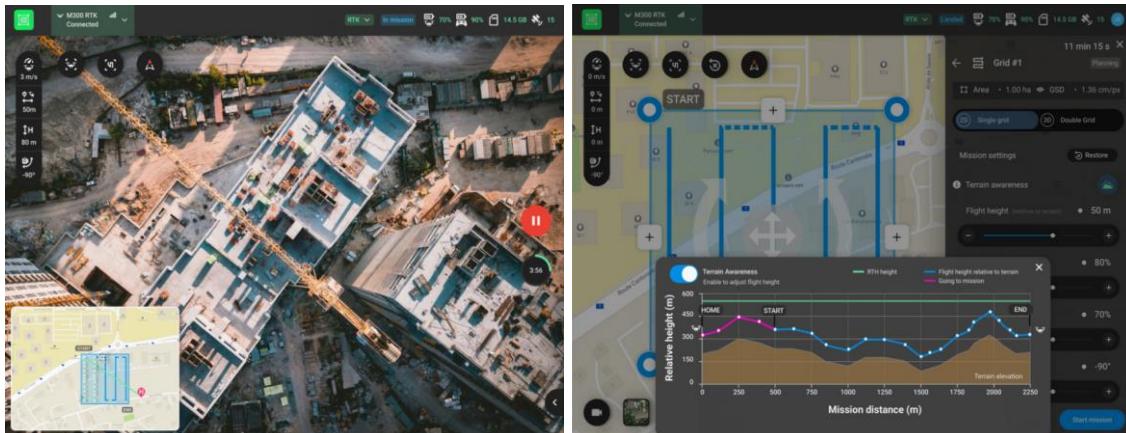


Figura 1.2: Capturas de pantalla de la versión de iPad de PIX4Dcapture Pro

2 Trabajos Previos y Estado del Arte

En esta sección se van a mencionar investigaciones académicas cuyo objetivo tiene relación con el mío con la intención de identificar puntos de mejora y de aprendizaje:

En primer lugar, se ha encontrado un trabajo [11] que, si bien es muy simple, tiene relación con la extracción de metadatos de las imágenes tomadas por el dron y puede servir como referencia muy básica del proceso a seguir. En este caso se utiliza una herramienta llamada ExiftoolGUI para extraer algunos metadatos básicos de las imágenes obtenidas con dos drones. Sin embargo, este proceso es extremadamente manual y más allá de demostrar la posibilidad de obtener las coordenadas geográficas, no tiene mucho más valor.

En segundo lugar, encontramos un estudio, esta vez centrado en la extracción de información sobre el vuelo de logs proporcionados por los fabricantes DJI, Parrot y Yuneec [11]. Estos logs son procesados y convertidos a archivos .csv, que permiten la creación de gráficos sobre el vuelo, con la ventaja de la mayor densidad de información que proporcionan los logs respecto a las imágenes. Si bien es un estudio cercano a mi objetivo, tan solo se está optimizando el proceso de lectura de los logs mediante un parseo, y luego se procede a obtener los gráficos de manera manual.

Se ha encontrado un estudio enfocado en el diseño y creación de un dron con guardado de telemetría [12]. Si bien no se ahonda en el procesado de esta telemetría, se ofrece información sobre cómo aplicar una matriz de rotación a los datos del gimbal: el yaw, el pitch y el roll, o las rotaciones del dron en los ejes x, y, z, respectivamente. Esto puede ser de gran utilidad para representar la orientación del dron o la cámara en los distintos visores de la aplicación.

También se ha encontrado un estudio sobre la superposición o solapado de imágenes ortogonales de drones [14]. Esta superposición de imágenes permite después que los modelos tridimensionales obtenidos por fotogrametría tengan mucha mayor calidad ya que un solapado de los datos facilita al algoritmo de fusión unir todas las imágenes en un solo modelo. Además de proponer un modelo de cálculo rápido de este solapamiento, también se realiza sobre un DEM (modelo de elevación digital), o sea, sobre un terreno irregular, lo que afecta a la proyección de la imagen. Finalmente se muestran visualizaciones en las que el color de la proyección varía dependiendo de la cantidad de solapado.

Como conclusión de estas propuestas e investigaciones académicas he obtenido que, si bien se han tratado diversos análisis de gran utilidad, se ha hecho de forma separada y sin prestar atención a las posibles automatizaciones que pueden convertir a este conjunto de herramientas en una sola aplicación muy versátil.

A continuación, analizaré productos comerciales que también tienen relación con mis objetivos:

En primer lugar, encontramos Drone Harmony [15], que ofrece el flujo de trabajo completo, desde planificación hasta inspección pasando por el propio vuelo. Tiene un visor 3D de los vuelos, pero es meramente visual y no ofrece información sobre los datos o análisis estadísticos de ningún tipo. Sólo soporta drones de la marca DJI.

Otro de los productos comerciales es Droneviewer [16], que se centra en la inspección visual de imágenes y videos. Ofrece gráficos 2D de algunos datos básicos respecto al tiempo y un visor 2D del vuelo superpuesto sobre un mapa. No ofrece información sobre los datos o análisis estadísticos de ningún tipo.

Otros dos productos proporcionados por Phantom Help, o sea, el equipo de asistencia técnica de DJI son DJI Flight Log Viewer [17] y DJI Flight Reader [18], aplicaciones web muy similares que actúan como parsers de los archivos log de DJI y los convierten a un formato .csv interactivo. Además, proporcionan herramientas para hacer una visualización en 2D del vuelo superpuesto sobre un mapa bastante básica. Al ser productos de DJI, sólo soportan los drones de su compañía.

Finalmente, Airdata [19] es un software centrado en el mantenimiento de drones, por lo que trata de llevar registros de toda la información posible como son la batería, el recorrido total, las zonas de mayor velocidad y aceleración, y ofrece un visor 2D superpuesto sobre un mapa que ofrece información sobre los distintos puntos del vuelo. Soporta logs e imágenes de multitud de fabricantes de drones y es online, por lo que se deben subir los archivos a su sistema antes de poder analizarlos, lo que significa que en caso de estar el piloto en campo y querer analizar un vuelo, no puede hacerlo si se trata de una zona remota o con mala conexión.

Como conclusión, de las opciones comerciales, Droneviewer y Airdata son los más similares a la aplicación objetivo de este trabajo de fin de grado, sin embargo, ninguno de los dos ofrece un visor 3D y aunque ofrecen gráficas sobre los datos y los parámetros de los drones a lo largo del vuelo, son demasiado superficiales y no son suficiente para hacer un análisis en profundidad de requerirlo el piloto. Además, el que la aplicación se pueda ejecutar localmente sin depender de conexiones a internet es un requisito indispensable que no cumple Airdata.



Figura 2.1: Captura de pantalla de la aplicación Droneviewer

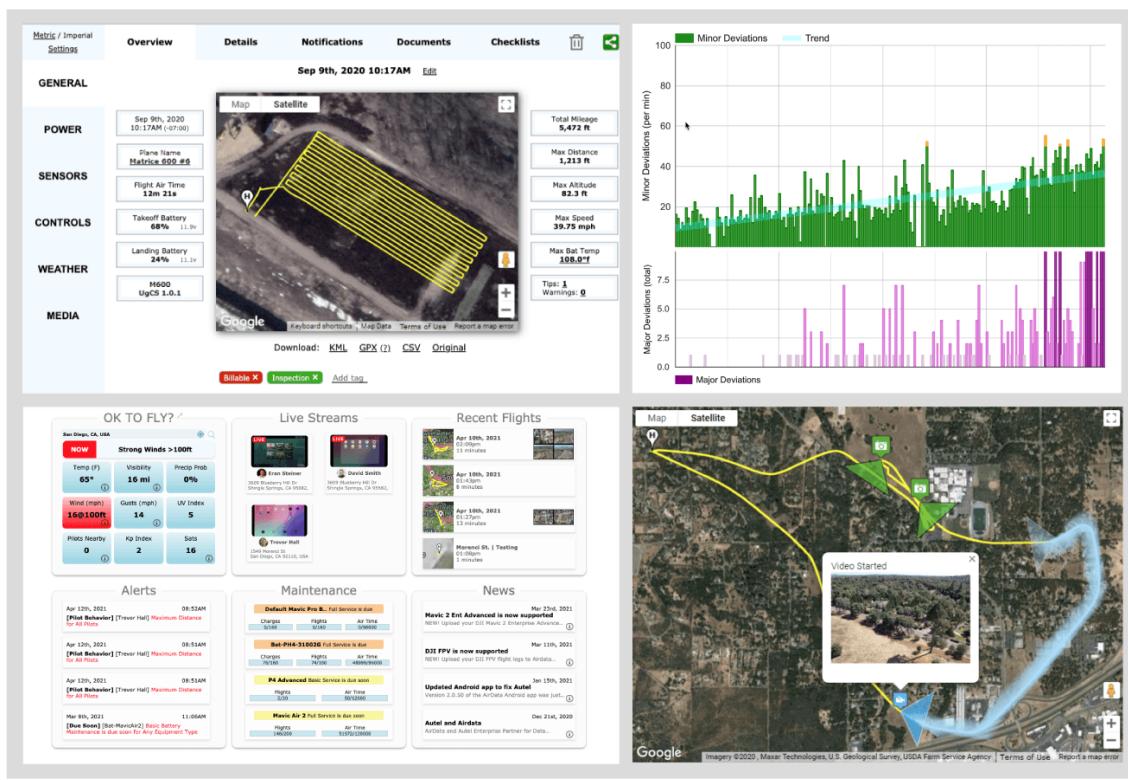


Figura 2.2: Montaje de cuatro visualizaciones distintas disponibles en la aplicación Airdata

3 Tecnologías empleadas

3.1 Lenguaje de programación: Python

Desde el inicio del proceso de conceptualización y diseño de la aplicación, estuve claro que el lenguaje de programación con el que se llevaría a cabo el desarrollo de la aplicación debía ser Python [20]. Esta decisión se basó en una serie de consideraciones estratégicas detalladas a continuación.

En primer lugar, hay que destacar la velocidad de desarrollo que Python ofrece en comparación con otros lenguajes. Aunque es cierto que su tiempo de ejecución es considerablemente más lento si lo comparamos con lenguajes compilados como C++ o Java, la agilidad con la que se pueden implementar y probar nuevas ideas en Python compensa con creces esta desventaja, permitiendo una iteración rápida que considero fundamental en un proyecto con un tiempo de desarrollo tan limitado como este.

Otro factor determinante es la amplia variedad de librerías externas disponibles, que han supuesto un ahorro significativo de tiempo y esfuerzo, ya que ha eliminado la necesidad de desarrollar una cantidad considerable de código desde cero. En otros lenguajes, muchas de las funcionalidades que se han utilizado en este proyecto habrían requerido una implementación manual, lo cual es de nuevo, inviable en un proyecto con un tiempo de desarrollo tan limitado.

Además, Python destaca por ser uno de los lenguajes de propósito general más eficaces en el ámbito de las ciencias de datos. Esto es un aspecto crítico para la aplicación, ya que su funcionalidad principal es el análisis de los registros de vuelo de drones. La capacidad de Python para manejar y procesar grandes conjuntos de datos de manera eficiente es por tanto esencial.

Finalmente, hemos elegido Python no sólo porque se alinea con los requerimientos técnicos del proyecto, sino también porque es el lenguaje que me resulta más familiar y con el que tengo más experiencia previa. Al haber sido el lenguaje que más he usado tanto a lo largo de mi trayectoria académica y profesional como en mis proyectos personales, he adquirido una comprensión avanzada de sus particularidades y sus mejores prácticas, lo que me ha permitido abordar los desafíos del proyecto con confianza y eficacia, y traducir mis ideas en soluciones prácticas con una rapidez que otros con otros lenguajes simplemente no habría sido posible.

3.2 Librerías externas

A continuación, se detallan las librerías externas utilizadas en el proyecto, junto con una breve descripción de su funcionalidad y la justificación de su elección.

1. Matplotlib:

Matplotlib [21] es una de las librerías más populares para la visualización de datos en Python. Es una librería muy completa y versátil que ofrece una gran variedad de gráficos y opciones de personalización. Se ha utilizado para todas las visualizaciones estáticas o no interactivas de la aplicación, ya que ofrece un control total sobre el diseño y la estética de los gráficos.

2. Numpy:

NumPy [22] es una librería de Python diseñada para trabajar con matrices y vectores de forma eficiente, además de ofrecer una gran variedad de funciones matemáticas y estadísticas que expanden las capacidades de Python en estos ámbitos.

Se ha utilizado tanto para el procesamiento de los datos de los diferentes tipos de registros de vuelo, como para funciones intermedias que requerían de implementaciones de cálculos como transformaciones geométricas, conversiones de unidades, operaciones de álgebra lineal, entre otros.

Por ejemplo, la implementación de NumPy del “array”, al ofrecer una estructura de datos más eficiente que las listas de Python, se ha utilizado para almacenar los datos de los registros de vuelo, ya que permite un acceso más rápido a los elementos de la matriz y facilita las operaciones con sus elementos. Gracias a esto, funciones trigonométricas como “sin”, “cos” o “arctan” o conversiones de unidades como “radians” o “deg2rad”, esenciales para el trabajo con diversos sistemas de coordenadas y orientaciones espaciales y por tanto usadas constantemente en el proyecto, se han podido aplicar a todos los elementos de un array de manera vectorizada con una sola llamada a la función, en lugar de tener que iterar sobre todos los elementos de una lista y aplicar la función a cada uno de ellos de forma individual, optimizando así el rendimiento de la aplicación.

Se han usado también funciones estadísticas y matemáticas proporcionadas por NumPy como “mean”, “min”, “max” o “sqrt” para cálculos de distancias y análisis estadísticos básicos, al ser la implementación de NumPy más eficiente que la de Python.

Las funciones de álgebra lineal como “dot”, “cross” o “meshgrid” han sido fundamentales a la hora de hacer cálculos de intersección con mapas de elevación, permitiendo hacer cálculos vectoriales de forma increíblemente eficiente.

Además, otras funciones como “empty”, “zeros_like”, “concatenate”, “interp” o “stack” se han podido usar para inicializar, combinar, estructurar y manipular los datos de los registros de vuelo con una facilidad que las estructuras de datos de Python no posibilitan.

3. Pandas:

Pandas [23] es una librería de Python diseñada para el análisis de datos. Ofrece una gran variedad de estructuras de datos y funciones para el procesamiento de datos. Junto a NumPy se ha utilizado para el procesamiento de los datos de los registros de vuelo, ya que ofrece una gran variedad de funciones para el análisis de datos que han facilitado enormemente el trabajo con los registros de vuelo.

La posibilidad de crear estructuras de datos llamadas “dataframes” que permiten almacenar múltiples tipos de datos de forma tabular, con filas y columnas, y acceder a ellos de forma eficiente, ha sido básica para el procesamiento de los registros de vuelo, ya que ha permitido almacenar sus datos de forma estructurada, y acceder y operar con ellos fácilmente.

Se ha utilizado también para resolver el problema que supone trabajar con registros de vuelos, donde los datos pueden no ser continuos, al posibilitar la interpolación de estos datos, rellenando los huecos con valores interpolados de forma muy sencilla.

4. Pandas_profiling y streamlit_pandas_profiling:

La librería pandas_profiling facilita enormemente la realización de análisis exploratorios de datos, facilitando la creación a partir de un “dataframe” de Pandas de un informe detallado sobre el conjunto de datos en cuestión, incluyendo estadísticas descriptivas, distribuciones de variables, correlaciones, y hasta gráficas y visualizaciones, que permiten obtener una visión general de los datos de un registro de vuelo, y así poder identificar rápidamente posibles problemas o anomalías en los datos.

Por su parte, la librería streamlit_pandas_profiling permite integrar en el framework el informe generado, facilitando su visualización en la aplicación web.

5. Pillow:

La librería Pillow [24] es una librería de Python diseñada para el procesamiento de imágenes. En este proyecto se ha utilizado tanto para implementar el visor de imágenes cuando se utilizan imágenes y no logs como fuente de datos del vuelo, así como para la lectura de los metadatos de las imágenes en cuestión, ya que ofrece una serie de funciones que permiten acceder a la sección EXIF y XMP de las imágenes, donde se almacenan los datos espaciales, temporales, de orientación del dron y su cámara, entre otros, en el momento de la captura de la imagen, entre otros.

6. Plotly y streamlit_plotly_events:

Plotly [25] es una librería de Python diseñada para la visualización de datos. Es similar a Matplotlib, pero ofrece una serie de ventajas que la hacen más adecuada para visualizaciones de datos interactivas como las que usa el visor 3D en torno al que gira la aplicación, como un mayor control sobre la cámara de la visualización, mayor rendimiento al renderizar representaciones en 3D, la integración con mapas geográficos bidimensionales, así como una excelente integración con el framework gracias a la librería complementaria streamlit_plotly_events, que facilita las interacciones con el gráfico al simplificar los eventos de interacción con el usuario.

7. SciPy:

SciPy [26] es una librería de Python diseñada principalmente para la realización de cálculos científicos. En este proyecto se usa exclusivamente por su la implementación del algoritmo de triangulación de Delaunay, que permite la creación de una malla de triángulos a partir de un conjunto de puntos y es necesario para visualizaciones de mapas de elevación en 3D.

8. Trimesh:

Trimesh [27] es una librería de Python diseñada para el procesamiento de mallas (*meshes*) tridimensionales. Al igual que SciPy, se ha utilizado exclusivamente para la visualización de mapas de elevación en 3D, ya que permite crear una malla a partir de datos geoespaciales.

9. Watchdog:

Watchdog [28] es una librería de Python diseñada para facilitar la monitorización de cambios en el sistema de archivos, y si bien no es necesaria para la ejecución del framework en Windows, se recomienda para su uso en

sistemas Unix, ya que permite la actualización automática de la aplicación web cuando se detectan cambios en el código, lo cual ha resultado muy útil durante el desarrollo de la aplicación porque no es necesario realizar un reinicio del servidor local para implementar cambios, evitando romper el flujo de trabajo.

3.3 Framework: Streamlit

Tras deliberar entre las muchas opciones de frameworks disponibles para Python, con el objetivo de simplificar y agilizar el desarrollo de la aplicación para poder centrarme en la implementación de la lógica de la aplicación y no en la implementación de la interfaz de usuario y poder cumplir con los plazos de entrega, se ha elegido Streamlit [29] como framework para el desarrollo de la aplicación.

Streamlit es un framework para Python que tiene como foco la creación de aplicaciones web interactivas, generalmente orientadas a data science y machine learning, pero que se puede enfocar a cualquier otro tipo de aplicación web, como la que se ha desarrollado, más orientada al análisis y visualización interactiva de datos. Destaca por su sencilla sintaxis, su facilidad de uso y su excelente integración con Python y algunas de sus librerías de visualización de datos más populares, incluyendo Matplotlib y Plotly, lo que asegura una representación de los datos precisa y correcta.

Es gracias a esta profunda integración con el lenguaje, que permite que los elementos de la interfaz de usuario sean objetos de Python a los que se puede acceder, consultar y modificar desde el propio código sin necesidad de un lenguaje como HTML o CSS, lo que simplifica enormemente el desarrollo de la aplicación al no resultar necesario aprender un nuevo lenguaje para crear el frontend. La sintaxis y la lógica son muy similares a las de Python, por lo que la curva de aprendizaje es muy suave y se puede empezar a crear aplicaciones web con solo unas pocas líneas de código. Esto podría resultar una limitación en aplicaciones más complejas, pero para el alcance de este proyecto, que requiere una interfaz funcional antes que una estética elaborada, es una opción ideal.

Además, aprovecha el hecho de que Python es un lenguaje interpretado, permitiendo modificaciones del código en tiempo real, sin necesidad de recompilaciones o redesplicajes del servidor local. Esta característica agiliza enormemente el desarrollo de las aplicaciones, ya que los cambios en el código se reflejan instantáneamente en la aplicación web, lo que permite iterar rápidamente sobre el diseño y la funcionalidad de la aplicación, y facilita la depuración de errores.

3.4 Otros

1. API Mapbox:

La API de Mapbox [30] es una API muy versátil que ofrece una amplia gama de herramientas de mapeo y geolocalización, algunas de las cuales se han integrado en las visualizaciones de la aplicación, incluyendo la posibilidad de obtener datos de elevación de puntos específicos, lo cual es esencial para la generación de mapas de elevación en 3D, y también proporciona mapas en 2D que pueden destacar diversos tipos de datos como el relieve, imágenes satelitales, tipo de terreno, edificaciones y carreteras, entre otros, lo que resulta muy útil en casos en los que el vuelo se tiene que ceñir a un área, relieve o estructura específica, como en el caso de los vuelos de inspección de infraestructuras.

Desde una perspectiva de costo, la API de Mapbox es ideal para la aplicación, ya que debido a su modelo de precios, que solo inicia el cobro una vez superado un umbral de llamadas muy superior al que puede realizar un solo individuo o una pequeña empresa, su uso es esencialmente gratuito y sin limitaciones.

Finalmente, el hecho de usar una API no implica que la aplicación dependa de ella, ya que sólo se llamará si el usuario activa la opción. En caso de no disponer de conexión a internet, la aplicación seguirá funcionando con normalidad, aunque sin las funcionalidades que dependen de la API.

2. Adobe Illustrator:

Adobe Illustrator [31] es un programa de diseño gráfico vectorial que se ha utilizado para la creación del ícono de la aplicación. Concretamente, se ha utilizado la versión del programa Adobe Illustrator CC 2021.

3. Git:

Durante el desarrollo de la aplicación se ha utilizado Git como sistema de control de versiones, y GitHub como repositorio remoto. Esto ha permitido mantener un registro de todos los cambios realizados en el código, así como la posibilidad de volver a versiones anteriores en caso de ser necesario.

4 Desarrollo

4.1 Identidad de la aplicación

Desde el inicio del desarrollo de la aplicación, se decidió hacer énfasis en crear una identidad propia para la aplicación, con el objetivo de hacer del proyecto uno más profesional, y no simplemente una aplicación de prueba.

Con este objetivo en mente, se decidió crear un nombre para la aplicación: SkyStats, un nombre corto pero muy descriptivo, que hace referencia a la utilidad de la aplicación, que es el análisis de registros de vuelo del cual se obtienen estadísticas (stats), y también al entorno en el que se desarrollan estos vuelos, que es el cielo (sky).

Además del nombre, también se decidió crear un logotipo y una paleta de colores muy sencilla.

El logotipo, diseñado con Adobe Illustrator CC 2021 y después renderizado como archivo “png” o “ico” para adaptarse a distintas circunstancias, consiste en un águila con las alas extendidas y con dos representaciones de ondas saliendo de su cabeza, haciendo referencia a la naturaleza de la aplicación, gracias a la cual recibimos información del vuelo de los drones que hay en el aire, todo ello rodeado por un engranaje que representa su carácter como herramienta. El engranaje se ha modificado para no tener sus dientes espaciados equitativamente por razones estéticas. Se ha elegido una estética minimalista y estilizada, ya que transmite correctamente el mensaje de que la aplicación es una herramienta profesional, y da una sensación de profesionalidad y modernidad.

Como paleta de colores, se ha elegido una combinación de morado y tonos blancos y claros, tanto por preferencia personal como porque son colores que destacan muy bien entre sí y dan una idea clara de los ajustes actuales de la aplicación. El morado se usa principalmente para los elementos interactivos y el logo, mientras que los tonos blancos y claros se usan para el fondo y los elementos no interactivos o menos importantes.



Figura 4.1.1: Logotipo de SkyStats



Figura 4.1.2: Logotipo junto al texto de SkyStats dentro de la aplicación



Figura 4.1.3 y 4.1.4: Pestaña de la aplicación web en el navegador Mozilla Firefox, con el logo como favicon. Tema claro y tema oscuro.



Figura 4.1.5: Icono de la aplicación en Windows

Como podemos observar en las figuras anteriores, el logotipo destaca y es fácilmente distinguible en diversas circunstancias, además de contrastar correctamente con distintos tonos de fondo.

Ya que el framework Streamlit es excelente a la hora de respetar las normas de diseño de Material Design [32] de Google, se ha decidido seguir estos estándares para la estética de la aplicación, al tratarse de un estilo muy sólido y que resulta muy familiar para los usuarios, lo que facilita la interacción con la aplicación.

Finalmente, ya que parte del desarrollo de esta aplicación se realizó durante la estancia en la empresa Pix4D, donde las comunicaciones se realizan en inglés, la aplicación comenzó a desarrollarse en inglés, y se ha decidido mantenerla así con el objetivo de mantener una coherencia en el código, y por conveniencia, ya que es el idioma en el que estoy más acostumbrado a programar y uno que hace la aplicación mucho más accesible a un público internacional.

4.2 Estructura del proyecto

A continuación, se muestra un diagrama de la estructura del proyecto, que se ha diseñado con el objetivo de mantener una organización lógica y coherente que facilite la navegación por el código y el resto de archivos, y la comprensión de su funcionamiento, a la vez que respeta los requisitos de organización que exige Streamlit para su correcto funcionamiento:

```
sky_stats/
├── streamlit/
│   └── config.toml
├── assets/
│   ├── logo_256.ico
│   └── logo.png
├── scripts/
│   ├── unix-setup.sh
│   └── windows-setup.ps1
└── src/
    ├── terrain_processing/
    │   ├── mapbox_elevation.py
    │   └── surface_intersections.py
    ├── camera.py
    ├── extractor.py
    ├── flight_logs_plots.py
    ├── flight_logs.py
    ├── gpscoords.py
    ├── image_plots_ui.py
    ├── image_plots.py
    ├── mission.py
    ├── shared_plots.py
    └── workspace.py
    ├── app.py
    ├── README.md
    ├── requirements.txt
    ├── run.ps1
    └── WINDOWS_SETUP.bat
```

Dentro de la carpeta “sky_stats/” tenemos las siguientes subcarpetas y archivos:

- “streamlit/”, que contiene el archivo de configuración “config.toml” de la aplicación, que detalla la paleta de colores.
- “assets/” que incluye los logos de la aplicación.
- “scripts/”, que junto a los archivos “app.py”, “requirements.txt”, “run.ps1” y “WINDOWS_SETUP.bat” permiten la instalación automática de la aplicación con un sólo clic.
- “src/” que contiene todo código de la aplicación. Siendo “image_plots_ui.py” el frontend y el resto de los archivos el backend. Estos archivos se van a explicar en mucho más detalle en secciones posteriores del documento.
- “README.md” contiene una descripción general de la aplicación junto a las instrucciones de instalación.

También hemos incluido en el directorio del proyecto dos carpetas adicionales.

La primera de ellas contiene una sencilla documentación que explica el funcionamiento y las características de la aplicación sin entrar en detalles técnicos sobre la implementación:

```
└── doc/
    └── images/
        └── doc.md
```

La segunda contiene algunos logs de prueba en formato “txt”, que al ser ligeros se pueden adjuntar con el resto del proyecto sin ocupar demasiado espacio y permiten a cualquiera probar la aplicación sin necesidad de obtener algún tipo de registro de vuelo por su cuenta:

```
└── test_data/
    ├── flight_log_grid_anafi.txt
    ├── flight_log_helix.txt
    ├── flight_log_line_of_sight.txt
    ├── flight_log_ta_long.txt
    └── flight_log_ta_short.txt
```

4.3 Extracción de datos

La aplicación SkyStats soporta dos tipos de datos como fuente de datos de vuelo: imágenes y logs.

Las imágenes son archivos de imagen en formato “jpg” o “jpeg”, que se obtienen al sacar fotografías con el dron durante el vuelo y son, generalmente, guardadas en una tarjeta SD que el usuario puede extraer y conectar al ordenador para transferirlas. Contienen metadatos en formato EXIF [33] y XMP [34] [35], que se pueden extraer con la librería Pillow. Estos metadatos contienen multitud de información sobre el vuelo, que podemos extraer y utilizar para realizar un análisis del vuelo.

Por su parte, los logs son archivos de texto en formato “txt” que contienen información muy detallada sobre el vuelo. Los logs pueden o bien guardarse en la tarjeta SD de la misma manera que las imágenes, o bien obtenerse a través de la aplicación PIX4Dcapture Pro. Estos logs usan un estándar propio de Pix4D, por lo que la implementación está específicamente pensada para funcionar con estos logs, aunque podría adaptarse para funcionar con otros logs de otros fabricantes. Cada línea del log proporciona información sobre un evento del vuelo y lo ubica en el tiempo, por lo que podemos utilizar esta información para realizar un análisis del vuelo.

Estos métodos de obtención de datos son independientes entre sí, por lo que sólo se puede usar uno de ellos a la vez. Esto se debe a que, después de realizar muchas pruebas con vuelos de los que se disponía tanto de imágenes como de logs, se ha llegado a la conclusión de que los logs y las imágenes no siempre coinciden.

Por ejemplo, puede darse el caso de que el log haya registrado un evento de captura de imagen pero que esa imagen no se haya llegado a sacar porque el dron tenía que esperar a que el buffer de la cámara se vaciara. Si bien este es un ejemplo bastante común que se daba especialmente en drones viejos durante los vuelos

de prueba y también dependía de factores externos como la velocidad de escritura de la tarjeta SD, no es raro encontrar una falta de correspondencia entre los logs y las imágenes, por lo que se ha decidido que la aplicación funcione con uno de los dos tipos de datos a la vez, y no con ambos.

Esto puede parecer una limitación, ya que es cierto que el análisis más completo se obtiene al utilizar ambos tipos de datos, pero introduce una serie de ventajas que hacen que esta decisión sea la más adecuada.

En primer lugar, se gana flexibilidad, ya que no es necesario tener tanto imágenes como logs para poder analizar un vuelo, sino que se puede analizar con uno de los dos, lo cual es especialmente útil en caso de que no se disponga de ambos tipos de datos.

Además, podemos sacar partido de las ventajas que ofrece cada tipo de datos. Por ejemplo, en caso de querer realizar una inspección meramente visual, algo muy común en el sector de la construcción o del mantenimiento de infraestructuras, las imágenes dan toda la información necesaria, por lo que el requerimiento de tener logs se convertiría en una limitación innecesaria.

Por otro lado, si se quiere realizar un análisis más detallado del propio vuelo y no tanto de aquello que haya fotografiado el dron, los logs son la mejor opción, ya que contienen mucha más información sobre el vuelo que las imágenes. Los logs tienen además la ventaja de que son mucho más ligeros que las imágenes, por lo que se pueden guardar en caso de querer realizar una inspección más detallada en el futuro sin tener que ocupar demasiado espacio, o enviarlos por correo o entre personas con facilidad.

A continuación, explicaremos el proceso de extracción de datos de las imágenes y los logs, así como la estructura de los datos extraídos.

4.3.1 Imágenes

La implementación de la extracción de datos de las imágenes se ha realizado en el archivo “extractor.py”, que contiene una serie de funciones que hacen uso de la librería Pillow y se encargan de extraer los datos de las imágenes y devolverlos en un formato adecuado para su uso en la aplicación.

Sin embargo, antes de explicar el funcionamiento de estas funciones, es necesario explicar la estructura de los metadatos de las imágenes. En primer lugar, explicaremos los dos estándares de metadatos que se utilizan en las imágenes: EXIF y XMP.

El formato EXIF (EXchangeable Image File format) es un estándar de metadatos creado por la JEIDA (Japan Electronic Industry Development Association) con el objetivo de estandarizar la información que se almacena en las imágenes digitales. Se trata de uno de los estándares de metadatos más populares y extendidos y destaca por su simplicidad y facilidad de implementación. Esto se debe a que el formato EXIF es un estándar muy conciso y específico, que define una serie de etiquetas o tags de metadatos que se pueden almacenar en las imágenes, y cada etiqueta tiene un formato específico que define el tipo de datos que se puede almacenar en ella. Es gracias a esta falta de flexibilidad que se trata de un formato tan sencillo y robusto, en el que no hay lugar para la ambigüedad, y es muy fácil

implementar un lector de metadatos que sepa exactamente qué datos va a encontrar en cada etiqueta y cómo interpretarlos.

Sin embargo, esta simplicidad también supone una desventaja, ya que el formato EXIF no es extensible, es decir, no se pueden añadir etiquetas nuevas, por lo que no es posible almacenar metadatos que no estén definidos en el estándar. Es por ello que los metadatos específicos de los drones se almacenan en el formato XMP.

El formato XMP (Extensible Metadata Platform) es un modelo de metadatos creado por Adobe que resulta particularmente útil en el ámbito de la fotografía digital con drones, al permitir que cada usuario u organización pueda definir sus propios metadatos que no se han contemplado en el estándar EXIF.

De nuevo, esto supone una desventaja, ya que al no existir un estándar formal, cada fabricante puede implementar su propio formato de almacenamiento de datos usando XMP, lo que supone que cada fabricante tenga su propio estándar de metadatos, con sus propias etiquetas y particularidades.

En nuestro caso, los drones de DJI y Parrot usan tanto el formato EXIF como su propia versión del formato XMP, por lo que la implementación de la extracción de metadatos de las imágenes se ha realizado teniendo en cuenta estos dos estándares.

Para ello, disponemos de una función “get_dji_metadata_v2” encargada de extraer los metadatos del formato XMP de DJI, a la que se le introduce un archivo de imagen pasándole su ruta, que abre en modo “rb” (read binary) con el objetivo de encontrar las etiquetas de delimitación de los metadatos XMP “<x:xmpmeta”. Una vez ha hecho eso, convierte los bytes delimitados a un string y lo parsea obteniendo un diccionario con los nombres de las etiquetas y sus valores. Este diccionario se devuelve como resultado de la función.

De la misma manera, tenemos una función “get_parrot_metadata” encargada de hacer lo mismo, pero teniendo en cuenta las particularidades del formato XMP de Parrot.

Ya que nos interesa tener un formato unificado de metadatos XMP, después de inspeccionar la documentación de los metadatos de cada uno de los fabricantes [36] [37], nos hemos decantado por usar como estándar el formato de DJI, ya que está mejor estructurado y es más consistente, además de ser el principal fabricante de drones a nivel mundial. Con ello en mente, hemos creado una función “transform_parrot_metadata_to_dji” que transforma los metadatos XMP de Parrot al formato de DJI.

Por ejemplo, en el caso de la etiqueta “RelativeAltitude”, DJI la almacena como un entero, mientras que Parrot la almacena como una fracción en formato string, por lo que la función transforma el valor de la etiqueta de Parrot a un entero.

En el caso de las orientaciones de la cámara y el dron, DJI las almacena ambas, mientras que Parrot sólo almacena la orientación de la cámara, por lo que la función inicializa las tres orientaciones a 0.

Finalmente, se uniformizan algunas etiquetas distintas que contienen la misma información, como es el caso de “GimbalRollDegree” o “SelfData” en DJI, que se almacenan como “CameraRollDegree” y “CustomId” en Parrot, respectivamente.

Por lo tanto, este formato no altera los metadatos de Parrot, sino que los adapta y los uniformiza al formato de DJI, para que todos los metadatos XMP tengan el mismo formato y sea más fácil trabajar con ellos.

A continuación, tenemos una función “`get_exif_metadata`” encargada de extraer los metadatos EXIF de una imagen. En este caso ya no es necesaria una implementación específica para cada fabricante, gracias a la uniformidad del estándar EXIF. Esta función también toma como argumento la ruta del archivo de imagen, pero aprovecha la función “`getexif`” de la librería Pillow para extraer los metadatos, que se filtran para eliminar aquellos que no nos interesan y devuelve un diccionario con los que sí.

Finalmente, es la función “`extract_metadata_dictionary`” la que llama a estas tres funciones y une los diccionarios de metadatos EXIF y XMP de DJI o Parrot en un único diccionario de metadatos, que se devuelve como resultado de la función. No es necesario comprobar si el archivo de imagen es de DJI o de Parrot, ya que la función “`get_dji_metadata_v2`” devuelve un diccionario vacío si no encuentra metadatos XMP de DJI, y la función “`get_parrot_metadata`” hace lo mismo si no encuentra metadatos XMP de Parrot.

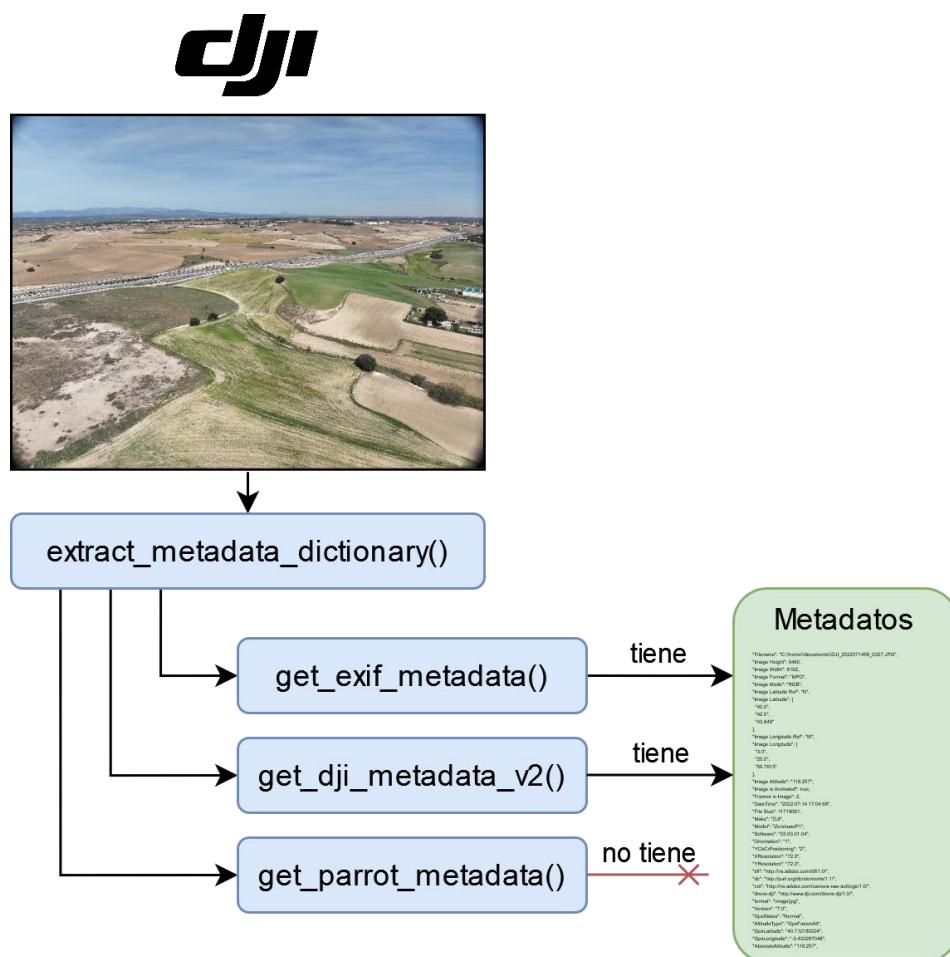


Figura 4.3.1.1: Diagrama de la extracción de datos de una imagen obtenida con un dron DJI

Parrot®

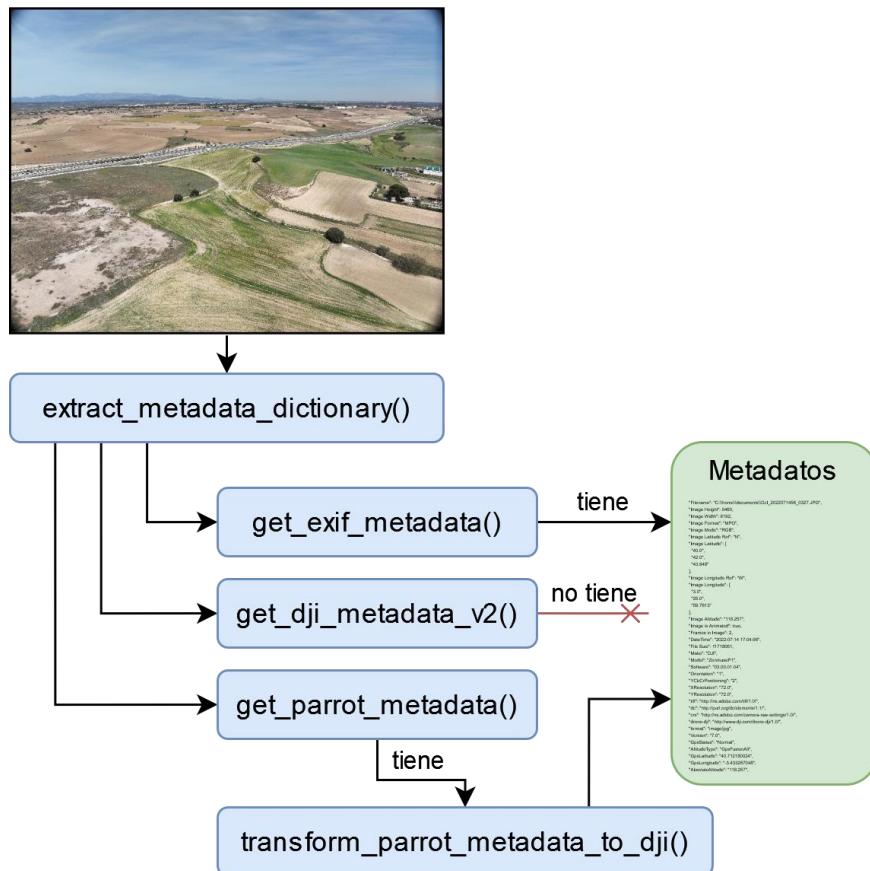


Figura 4.3.1.2: Diagrama de la extracción de datos de una imagen obtenida con un dron Parrot

A continuación, se muestra un ejemplo de los metadatos extraídos de una imagen de DJI, resultado de pasarla por la función “extract_metadata_dictionary”, junto a la descripción de cada uno de ellos:

Etiqueta	Valor	Descripción
Filename	C:\\documents\\demo\\tfg\\170456_0327.JPG	Ruta del archivo de imagen
Image Size	[8192, 5460]	Tamaño de la imagen en píxeles
Image Height	5460	Altura de la imagen en píxeles
Image Width	8192	Anchura de la imagen en píxeles
Image Format	MPO	El archivo de imagen puede contener más de una imagen (MPO = Multi Picture Object). En este caso no es así, pero DJI almacena las imágenes en este formato, por si se usan otros sensores de la cámara del dron (infrarrojos, térmicos, etc.) y es necesario almacenar más de una imagen en el mismo archivo.
Image Mode	RGB	Modo de color de la imagen
Image Latitude Ref	N	La latitud de la imagen está en el hemisferio norte
Image Latitude	[40.0, 42.0, 43.848]	Latitud de la imagen en grados, minutos y segundos
Image Longitude Ref	W	La longitud de la imagen está en el hemisferio oeste
Image Longitude	[3.0, 25.0, 59.7613]	Longitud de la imagen en grados, minutos y segundos
Image Altitude	118.257	Altitud de la imagen en metros sobre el nivel del mar
Image is Animated	true	La imagen no está animada como un GIF, pero puede contener más de una imagen al ser un archivo MPO.
Frames in Image	1	Número de imágenes en el archivo MPO
DateTime	2022:07:14 17:04:56	Fecha y hora de la captura de la imagen
File Size	11718061	Tamaño del archivo de imagen en bytes
ExposureBiasValue	N03	Valor de compensación de exposición, en este caso 0.3 pasos negativos
ExifOffset	260	Offset de los metadatos EXIF en el archivo de imagen en bytes
ImageDescription	default	Descripción de la imagen con su valor por defecto
Make	DJI	Fabricante de la cámara
Model	ZenmuseP1	Modelo de la cámara

Software	03.00.01.04	Versión del software de la cámara
Orientation	1	Orientación de la imagen, 1 significa horizontal
XResolution	72.0	Resolución horizontal de la imagen en píxeles por pulgada
YResolution	72.0	Resolución vertical de la imagen en píxeles por pulgada
format	image/jpg	Formato de la imagen
GpsStatus	RTK	Estado del sistema de posicionamiento GPS, en este caso usando RTK
AltitudeType	RtkAlt	Altitud obtenida haciendo uso del sistema RTK
GpsLatitude	40.71218002 4	Latitud del dron en grados decimales
GpsLongitude	- 3.433267048	Longitud del dron en grados decimales
AbsoluteAltitude	118.257	Altitud absoluta del dron en metros sobre el nivel del mar en metros
RelativeAltitude	18.027	Altitud relativa del dron en metros sobre el punto de despegue en metros
GimbalRollDegree	0.00	Orientación de la cámara en torno al eje X, con 0 indicando nivelación y valores positivos o negativos inclinación lateral, en grados decimales
GimbalYawDegree	-8.00	Orientación de la cámara en torno al eje Y, con 0 indicando el norte, 90 el este, -90 el oeste y ±180 el sur, en grados decimales
GimbalPitchDegree	-40.00	Orientación de la cámara en torno al eje Z, con 0 indicando que la cámara apunta hacia el horizonte, valores positivos indicando que apunta hacia arriba y valores negativos indicando que apunta hacia abajo, en grados decimales
FlightRollDegree	0.80	Rotación del dron alrededor del eje X, con 0 indicando nivelación y valores positivos o negativos inclinación lateral, en grados decimales
FlightYawDegree	-15.30	Orientación del dron, con 0 hacia el norte, 90 hacia el este, -90 hacia el oeste y ±180 hacia el sur, en grados decimales
FlightPitchDegree	-0.70	Inclinación del dron alrededor del eje Y, con 0 para vuelo nivelado, positivo hacia arriba y negativo hacia abajo, en grados decimales
FlightXSpeed	0.0	Velocidad del dron en el eje horizontal este-oeste en m/s
FlightYSpeed	0.0	Velocidad del dron en el eje horizontal norte-sur en m/s
FlightZSpeed	0.6	Velocidad del dron en el eje vertical, ascendente o descendente en m/s
CamReverse	0	Indica si la cámara está boca abajo o no

SelfData	CP_2_165781 0338_5_3	Identificador de la misión
RtkFlag	16	Estado del sistema RTK, 0 indica que ha fallado, 16 indica posicionamiento de un solo punto (precisión del orden de metros), 34 indica posicionamiento de punto flotante (precisión del orden de decímetros), 50 indica posicionamiento fijo (precisión del orden de centímetros)
RtkStdLon	1.39892	Desviación estándar de la longitud del posicionamiento RTK en cm
RtkStdLat	2.06608	Desviación estándar de la latitud del posicionamiento RTK en cm
RtkStdHgt	3.65655	Desviación estándar de la altitud del posicionamiento RTK en cm
RtkDiffAge	0.0007	Tiempo en segundos desde la última actualización del sistema RTK
SurveyingMode	1	1 implica que la imagen es adecuada para operaciones de mapeo debido a su alta precisión, 0 implica que no lo es
ShutterType	Mechanical	Tipo de obturador de la cámara
ShutterCount	20634	Número de veces que se ha activado el obturador de la cámara
CameraSerialNumber	3XMDJBW0018 DB9	Número de serie de la cámara
LensSerialNumber	01KY109G091 P	Número de serie del objetivo
DroneModel	Matrice 300 RTK	Modelo del dron
DroneSerialNumber	1ZNBJAX00C0 05V	Número de serie del dron
HasSettings	False	Indica si la imagen tiene ajustes de cámara aplicados
HasCrop	False	Indica si la imagen tiene un recorte aplicado
AlreadyApplied	False	Indica si los ajustes de cámara ya se han aplicado a la imagen

Al final del proceso de extracción de datos de las imágenes, antes de comenzar con las visualizaciones es necesario hacer un procesado de los datos extraídos, ya que algunos de ellos no están en el formato adecuado para su uso en las visualizaciones.

Para ello, usamos algunas funciones de “gpscoords.py”, detalladas a continuación:

- “dms_to_dd”: Función que transforma las coordenadas de grados, minutos y segundos a grados decimales.

- “transform_4326_to_3857”: Función que transforma las coordenadas del sistema de coordenadas geográfico basado en la altitud y longitud EPSG:4326 o WGS84 (World Geodetic System 1984) que usa el sistema GPS y aproxima la forma de la Tierra a un elipsoide, al sistema de coordenadas proyectado EPSG:3857 o Web Mercator, que usa la proyección de Mercator y aproxima la forma de la Tierra a un cilindro, de manera que se pueda representar en un mapa bidimensional, de la misma manera que lo hacen los mapas de Google Maps.
- “to_cartesian”: Función que convierte las coordenadas geográficas en grados decimales a coordenadas cartesianas en metros, para poder realizar cálculos de distancias y para poder representarlas en un mapa sin distorsiones. Estas coordenadas pueden después normalizarse para que el punto de origen sea el (0, 0) y así poder representarlas mejor en las distintas visualizaciones de la aplicación.

Además, como paso final previo a las visualizaciones, se hace una sencilla preparación de los datos dentro de “image_plots.py”, que consiste en la normalización de los datos que lo necesiten, como son las coordenadas cartesianas, para que el punto de origen sea el (0, 0) y así poder representarlas mejor en las distintas visualizaciones de la aplicación, o el tiempo de captura de las imágenes, para que la primera imagen del vuelo tenga un timestamp de 0 segundos, y el resto de imágenes tengan un timestamp relativo a esta primera imagen.

También se calculan algunas variables adicionales que pueden ser útiles para las visualizaciones pero que los metadatos no proporcionan, como la distancia recorrida por el dron o su aceleración.

Tras este breve procesado, los datos están listos para ser usados en las visualizaciones y en los análisis de datos.

4.3.2 Logs

La implementación de la extracción de datos de los logs se ha realizado en el archivo “flight_logs.py”, que contiene la clase “FlightLog” y una serie de funciones que hacen uso de las librerías NumPy y Pandas y se encargan de extraer los datos de los logs y procesarlos para que sean adecuados para su uso en la aplicación.

Pero antes de explicar el funcionamiento de esta clase y sus funciones, es necesario explicar la estructura de los logs de Pix4D, ya que siguen un formato que si bien no es propietario y cualquiera que use la aplicación puede tener acceso a ellos, no es un estándar formal con una documentación oficial, por lo que ha sido necesario realizar un análisis para entender su estructura y poder extraer los datos adecuadamente.

Los logs siguen una estructura muy simple, en la que cada línea del log contiene un tipo de evento representado por una etiqueta, junto a un timestamp y los datos correspondientes a ese evento, como podemos ver a continuación en este extracto simplificado de un log real:

```
SAT, 1681203581.102, 24
GPS, 1681203581.149, 40.5716339, -4.2125810, 14.86
VEL, 1681203581.149, 0.04, -0.01, 3.07
PIC, 1681203581.197, 4620013_1.JPG, 0
SAT, 1681203581.301, 23
GPS, 1681203581.353, 40.5716339, -4.2125810, 15.47
VEL, 1681203581.353, 0.03, -0.01, 3.05
```

En el extracto anterior podemos ver, por ejemplo, como la primera línea contiene un evento de tipo “GPS”, con un timestamp de 1681203581.149, y los datos correspondientes a ese evento, que son la latitud, longitud y altitud relativa del dron en ese momento. Un poco más abajo, podemos ver otro evento de tipo “GPS” con un timestamp un par de décimas de segundo posterior de 1681203581.353, y los nuevos datos.

Como podemos observar, al contrario que con las imágenes en las que cada una nos proporciona una instantánea de los datos del vuelo en ese momento, en los logs no se registran todos los eventos en todos los timestamps, sino que se registran sólo aquellos que van cambiando. Esto es muy positivo ya que gracias a ellos, los logs siempre nos van a proporcionar los datos más recientes de cada tipo de evento a cada momento, y no datos redundantes que no aportan información nueva. Sin embargo, tiene un inconveniente, y es que los datos no son continuos, hay hueco entre ellos, por lo que si quisieramos conocer, por ejemplo, los valores de “GPS”, “SAT” (número de satélites), y “VEL” (velocidad del dron) en el momento concreto en el que sucede el evento “PIC” (captura de imagen), no podríamos obtenerlos directamente del log, ya que no hay datos de esos eventos en ese momento. Esto es un problema, ya que para poder realizar un análisis completo de los datos de un vuelo, que incluya comparaciones y correlaciones entre distintos tipos de datos, o visualizaciones de datos en un momento concreto, necesitamos que los datos sean continuos.

Timestamp (s)	PIC	GPS Lat (°)	GPS Lon (°)	GPS Alt (°)	VEL X (m/s)	VEL Y (m/s)	VEL Z (m/s)	SAT (n)
1681203581.102								24
1681203581.149		40.5716339	-4.212581	14.86	0.04	-0.01	3.07	
1681203581.197	4620013 _1.JPG							
1681203581.250								
1681203581.301								23
1681203581.353		40.5716338	-4.212581	15.47	0.03	-0.01	3.05	

Figura 4.3.2.1: Tabla de datos con huecos

Para solucionar este problema, hemos recurrido a la interpolación de datos. De manera que podemos hacer una estimación de los valores de distintos eventos para un timestamp concreto, basándonos en los valores de los eventos anteriores y posteriores a ese timestamp. De esta manera, podemos obtener datos continuos y completos, que nos permiten realizar un análisis completo de los datos de un vuelo.

Timestamp (s)	PIC	GPS Lat (°)	GPS Lon (°)	GPS Alt (°)	VEL X (m/s)	VEL Y (m/s)	VEL Z (m/s)	SAT (n)
1681203581.102	-	40.5716339	-4.212581	14.86	0.04	-0.01	3.07	24
1681203581.149	-	40.5716339	-4.212581	14.86	0.04	-0.01	3.07	24
1681203581.197	4620013 _1.JPG	40.57163388	-4.212581	15.003529	0.037647	-0.01	3.065294	24
1681203581.250	-	40.57163385	-4.212581	15.16201	0.035049	-0.01	3.060098	24
1681203581.301	-	40.57163383	-4.212581	15.314511	0.032549	-0.01	3.055098	23
1681203581.353	-	40.5716338	-4.212581	15.47	0.03	-0.01	3.05	23

Figura 4.3.2.2: Tabla de datos interpolados

Como podemos apreciar en la figura 4.3.2.2, ahora que los datos se han interpolado, podemos conocer los valores de todos los eventos en cualquier momento.

Es importante destacar que la interpolación de datos no se realiza de forma directa, si no que se tiene una consideración temporal. Esto significa que además de los valores de la propia columna a interpolar, hemos aprovechado que los logs registran el tiempo de cada timestamp para indexar los datos en función del tiempo. De esta manera, realizamos una interpolación ponderada que asigna más peso a los valores que están más cercanos en el tiempo al punto que se está interpolando, lo que proporciona unos resultados más precisos y realistas.

Además, dependiendo del tipo de evento, se ha utilizado un tipo de interpolación u otro. Por ejemplo, en el caso de la tabla anterior, los eventos “GPS” o “SAT” se han interpolado de forma lineal, al tratarse de datos continuos que admiten valores intermedios.

Sin embargo, para el evento “SAT”, que cuenta el número de satélites a los que el dron está conectado en ese momento, no tiene sentido interpolar de forma lineal, ya que no se puede tener un número no entero de satélites. Por lo que aprovechando que el evento sólo se registra cuando hay un cambio en el número de satélites, se ha optado por una interpolación que continúa el último valor conocido hasta que se registra un nuevo valor, momento en el que se actualiza el valor interpolado.

En el caso del evento “PIC”, que indica cuando el dron ha tomado una fotografía y registra el nombre de la imagen, no tiene sentido realizar ningún tipo de interpolación ya que se trata de un evento puntual y único.

Para los casos del principio y el final de los logs, en los que no hay datos anteriores o posteriores para interpolar, se ha optado por extender el primer valor conocido hasta el principio del log, y el último valor conocido hasta el final.

Si bien esta interpolación no es perfecta, ya que no es posible saber con exactitud qué valores tenían los eventos en los timestamps intermedios, se trata de una aproximación suficientemente precisa como para poder realizar un análisis más completo de los datos del vuelo a partir del log.

Finalmente, y al igual que se explicó en la sección anterior sobre las imágenes, es necesario hacer un procesado sencillo de los datos, haciendo uso de algunas funciones de “gpscoords.py”, como “dms_to_dd”, “transform_4326_to_3857” o “to_cartesian”, para que los datos estén en el formato adecuado para su uso en las visualizaciones. De igual manera, se calculan algunas variables adicionales como la distancia recorrida por el dron o su aceleración, y se normalizan los datos que lo necesiten, como las coordenadas cartesianas o el tiempo de captura de las imágenes. Después de este procesado los datos ya están listos para ser usados en las visualizaciones, como podemos ver en la siguiente tabla:

Timestamp (s)	PIC	GPS Lat (m)	GPS Lon (m)	GPS Alt (m)	VEL X (m/s)	VEL Y (m/s)	VEL Z (m/s)	SAT (n)
0.000	-	0	0	0	0.04	-0.01	3.07	24
0.047	-	0	0	0	0.04	-0.01	3.07	24
0.095	4620013 _1.JPG	-0.00222	0	0.14353	0.037647	-0.01	3.065294	24
0.148	-	-0.0056	0	0.30201	0.035049	-0.01	3.060098	24
0.199	-	-0.0078	0	0.45451	0.032549	-0.01	3.055098	23
0.251	-	-0.0111	0	0.61	0.03	-0.01	3.05	23

Figura 4.3.2.3: Tabla de datos interpolados y transformados

Respecto a la implementación, hemos aprovechado que los logs incluyen una serie de eventos FMT (formato) al principio de cada fichero que contiene información sobre los tipos de eventos que se pueden registrar junto a los datos que aportará cada uno de ellos, para crear un parser que lea el log y extraiga los datos de los eventos que nos interesan.

El formato, al cual se le ha añadido el nombre completo de cada tipo de evento para que quede más claro en caso de que no se entiendan sus identificadores, es el siguiente:

- Versión: FMT, VERS, Time:double, Value:string
- GPS: FMT, GPS, Time:double, Lat:double, Lon:double, RelAlt:double
- Attitude (orientación del dron) : FMT, ATT, Time:double, Yaw:double, Pitch:double, Roll:double
- Gimbal (orientación de la cámara): FMT, GIM, Time:double, Yaw:double, Pitch:double, Roll:double, Camera:int
- Velocidad: FMT, VEL, Time:double, VelX:double, VelY:double, VelZ:double
- Número de satélites: FMT, SAT, Time:double, Value:int
- Link (calidad de conexión): FMT, LINK, Time:double, Percent:optional<int>
- Batería: FMT, BATT, Time:double, Percent:int
- Batería del mando: FMT, RBAT, Time:double, Percent:int
- Picture (fotografía): FMT, PIC, Time:double, MediaId:string, Camera:int
- Live picture session status (estado de la vista previa): FMT, LSES, Time:double, Camera:int, Status:string[Enabled,Started,Failed,Disabled]
- Waypoint (punto de paso): FMT, WPT, Time:double, Index:int
- Event type (tipo de evento): FMT, EVT, Time:double, Event:string[MissionStarted,MissionFinished,MissionPaused,MissionResumed,MissionClosed,StopCommandSent,PhotoExpected,PhotoCommandSent,PhotoWaypointReached]
- Flying state (estado de vuelo): FMT, FLST, Time:double, Event:string[Unknown,Landed,Takingoff,GoingToMission,InMission,InMissionPaused,Loitering,GoingToLand,Landing,ManualFlying]
- Storage (almacenamiento): FMT, STRG, Time:double, FreeStorageMB:int
- Compensación de exposición: FMT, EXPC, Time:double, ExposureCompensation:string
- Bridge event (evento de comunicación con la app): FMT, BREV, Time:double, Key:string, Value:string, Error:optional<string>, Error:optional<string>
- Compass sensor state (estado de la brújula/giroscopio): FMT, CSS, Time:double, Value:string, Value:string[Disconnected, Calibrating, Idle, DataException, SuperModulusSamll, SuperModulusWeak, SuperModulusDeviate, CalibrationFailed, InconsistentDirection, Unknown]

Toda la implementación se ha realizado dentro del constructor “`__init__`” de la clase “FlightLog” dentro del fichero “`flight_logs.py`”, que funciona de la siguiente manera:

En primer lugar, el constructor recibe como argumento la ruta del archivo de log, que se guarda en la variable “`file_path`” de la clase.

A continuación, se inicializan las variables “`parsed_data`” como lista, y “`dataframe`”, “`interpolated_dataframe`” e “`images_dataframe`” como objetos de la clase DataFrame de la librería Pandas.

Después, se llama a la función “`parse_flight_log`”, que divide el log en líneas y las guarda en “`parsed_data`”, tras lo cual se llama a la función “`process_parsed_data`”, que extrae los datos de cada línea del log teniendo en cuenta que ciertos eventos pueden tener más o menos parámetros asociados, los transforma de string al tipo de dato adecuado, y los guarda en “`dataframe`”.

A continuación, se llama a la función “`add_missing_columns_to_dataframe`”, que añade las columnas de datos que no proporciona directamente el log, como son las columnas de distancia o aceleración. Aprovechando la creación de nuevas columnas, esta función también se encarga de transformar y normalizar los datos a los formatos según sea necesario.

Finalmente, llamamos a la función “`interpolate_dataframe`”, que interpola los datos de “`dataframe`” y los guarda en “`interpolated_dataframe`”, que es el dataframe al que accederemos para realizar las visualizaciones y el análisis de los datos.

4.4 Selección de vuelo

Antes de proceder a realizar las visualizaciones y análisis de nuestros vuelos, es necesario seleccionar aquel que queremos analizar. Para ello, se ha desarrollado como parte de la interfaz de SkyStats, un menú de selección de vuelo y algunas otras opciones de análisis que se presenta como una barra lateral.

Esta barra lateral o sidebar, que aprovecha la implementación de Streamlit de la misma, contiene el logo de SkyStats y el nombre de la aplicación en la parte superior, un formulario donde se puede elegir el tipo de datos a analizar, y una entrada de texto donde se puede introducir la ruta de la carpeta que contiene los archivos relevantes. La barra lateral también se puede ocultar y mostrar de ser necesario, lo que resulta muy útil en caso de que se quiera maximizar el espacio de la pantalla para las visualizaciones o de usar la aplicación en un dispositivo con una pantalla pequeña.

Nada más iniciar la aplicación, el menú de selección de vuelo mostrará un mensaje al usuario indicando que debe introducir la ruta de la carpeta que contiene los archivos de vuelo. Al ser un mensaje con una intención neutral informativa, se ha decidido mostrarlo con un color azul:

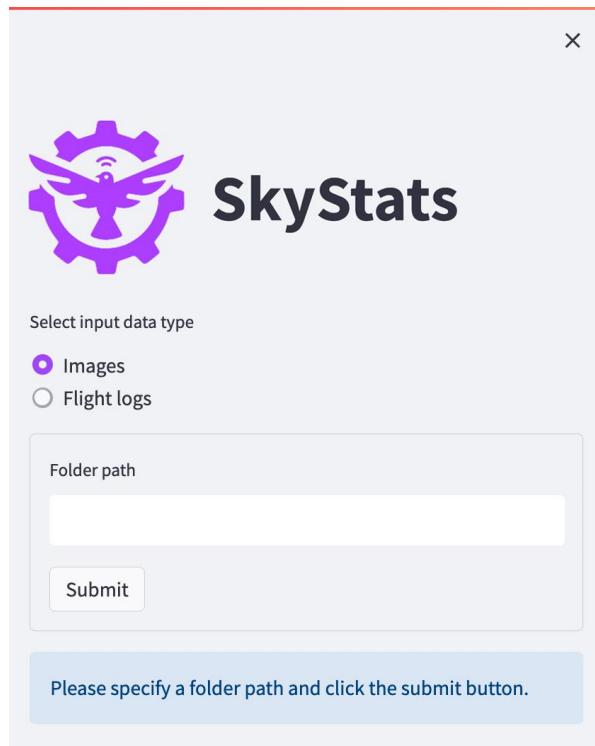


Figura 4.4.1: Menú de selección de vuelo con mensaje informando al usuario de los pasos a seguir

La carpeta seleccionada debería contener los archivos del tipo a analizar seleccionado. Estos archivos pueden estar en la raíz de la carpeta o en subcarpetas, ya que la aplicación los buscará recursivamente.

En caso de que la carpeta contenga archivos de otros tipos, además de los archivos del tipo seleccionado, la aplicación los ignorará, por lo que no hay problema en seleccionar una carpeta que contenga tanto imágenes como logs, o cualquier otro archivo.

Si la carpeta seleccionada contiene imágenes o logs en el formato adecuado (“jpg” o “jpeg” para las imágenes, “txt” para los logs), pero que no están soportados por SkyStats (por ejemplo, imágenes que no contienen los metadatos necesarios, logs que no están en el formato de Pix4D), la aplicación los ignorará.

Si la carpeta seleccionada no contiene ningún archivo soportado del tipo seleccionado o en caso de que la carpeta seleccionada no exista, la aplicación mostrará un mensaje de error indicándolo. Al ser mensajes de error, se mostrarán de color rojo:

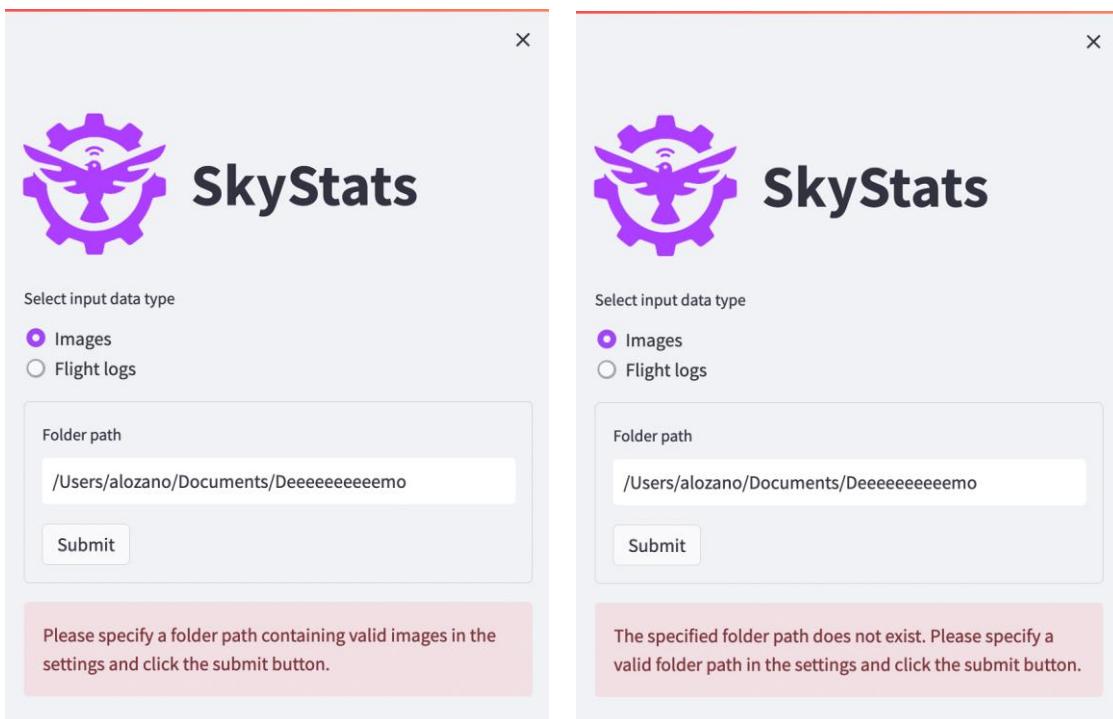


Figura 4.4.2 y 4.4.3: Menús de selección de vuelo con los distintos mensajes de error

Si la carpeta contiene los archivos adecuados para el tipo de datos seleccionado, el menú de selección de vuelo se comportará de una manera o de otra dependiendo de si se han seleccionado imágenes o logs.

4.4.1 Imágenes

Ya que las imágenes son archivos independientes, para poder analizar un vuelo a partir de estas es necesario agruparlas primero. Una solución sencilla a este problema sería simplemente agrupar las imágenes de un vuelo concreto en una subcarpeta dentro de la carpeta proporcionada al programa. Sin embargo, los fabricantes de drones no siempre proporcionan la posibilidad de clasificar las misiones por carpetas, ya que prefieren guardarlas en un solo directorio sin consideración alguna por el vuelo al que pertenecen.

Es por ello que se ha optado por una solución más robusta que además no requiere que las imágenes estén agrupadas en una carpeta concreta. Esta solución consiste en agrupar las imágenes en base a su parámetro “SelfData”, presente en los metadatos guardados en la partición XMP de las imágenes y que la aplicación PIX4Dcapture Pro utiliza para guardar un identificador del vuelo o misión al que pertenecen las imágenes. De esta manera, podemos agrupar las imágenes de un vuelo concreto independientemente de la carpeta en la que se encuentren, siempre y cuando tengan el parámetro “SelfData” en sus metadatos.

Para llevar a cabo esta agrupación, en primer lugar, el programa utiliza la función “add_folder” de la clase “Workspace” dentro de “workspace.py”, que recibe como argumento la ruta de la carpeta seleccionada y hace uso de la función “extract_images_from_folder” de la clase “ExtractionJob” presente en

“extractor.py”, que recorre recursivamente la carpeta seleccionada en busca de archivos con el formato adecuado.

Cuando una imagen es encontrada, se llama a la función “retrieve_missionkey_from_image_metadata” de la clase “Workspace”, que extrae el parámetro “SelfData”.

En el caso de que el parámetro “SelfData” sea uno nuevo, se creará un índice dentro del diccionario “missions” de la clase “Workspace” con el valor de “SelfData” como clave, y se añadirá la ruta de la imagen a una lista como valor de su misión correspondiente. Posteriormente, si una imagen con el mismo valor de “SelfData” es encontrada, se añadirá a la misma lista de imágenes.

De esta manera, para cuando haya acabado el recorrido de la carpeta, el diccionario “missions” contendrá una lista de imágenes para cada misión, y cada misión estará identificada por su valor de “SelfData”.

De no tener el parámetro “SelfData” en sus metadatos, las imágenes no se considerarán válidas para el programa. Mientras que, en el caso de tenerlo, pero vacío o con el formato incorrecto, las imágenes se agruparán en una misma misión, pero no se podrá extraer información de su identificador.

El parámetro “SelfData” es un parámetro que PIX4Dcapture Pro añade a las imágenes y que además de actuar como identificador único de un vuelo, también contiene información útil sobre el mismo separada por barras bajas “_”, detallada a continuación:

Ejemplo de parámetro “SelfData”:

“SelfData”: “CP_2_1666085234_9_1”

Desglose de los parámetros:

- CP: Capture Pro
- 2: Versión del formato
- 1666085234: Identificador del proyecto basado en el timestamp en formato Unix de la primera imagen del vuelo
- 9: Tipo de misión. En este caso, 9 corresponde a una misión de tipo “Cell tower helix”, que consiste en una misión en forma de espiral alrededor de una torre o edificación vertical y resulta especialmente útil para inspección de infraestructuras como torres de telecomunicaciones. En total hay 15 tipos de misiones distintas.
- 1: Orden de la misión dentro del proyecto

Para extraer toda la información posible de la misión, pasamos el parámetro “SelfData” por las funciones de “custom_tag_parser”, “custom_tag_formatter” y “custom_tag_regex” del fichero “mission.py”, que se encargan de separar los parámetros divididos por barras bajas, interpretarlos y devolverlos habiendo sustituido los parámetros numéricos por sus nombres completos.

Ya que además del valor “SelfData”, que hace de clave para la misión dentro del diccionario “missions”, también disponemos de sus imágenes correspondientes, accedemos a la primera imagen de cada misión y usamos la función “data_from_camera_code” del fichero “camera.py”, para obtener el modelo del dron asociado a la cámara que aparece en los metadatos de la imagen.

De esta manera, conseguimos transformar el parámetro “SelfData” en un string que ya no actúa tan solo de identificador, sino que es legible de cara al usuario y contiene información útil. En el caso del ejemplo anterior, esta sería la transformación:

```
"CP_2_1666085234_9_1": "2022-10-18 11:27:14 - Cell tower helix 1 - Phantom 4 Pro V2 (175 images)"
```

Este nuevo formato proporciona al usuario la siguiente información que permite que la misión sea fácilmente identifiable:

- Fecha y hora de la misión (en formato YYYY-MM-DD HH:MM:SS)
- Tipo de misión (“Cell tower helix”, “Grid”, “Manual”)
- Orden de la misión (en caso de haber varias misiones del mismo tipo en la misma sesión)
- Modelo del dron (“Phantom 4 Pro V2”, “Mavic 3 Enterprise”, “Anafi AI”)
- Número de imágenes de la misión

En caso de no tener un valor válido de “SelfData”, las imágenes se agrupan correctamente pero no se podría extraer información de su identificador, por lo que se mostraría de la siguiente manera:

```
"SelfData" vacío o no soportado: "No custom tag - Mavic 3 Enterprise (1371 images)"
```

Ahora que conocemos cómo se realiza la agrupación de imágenes, si en la interfaz anterior (Figura 4.4.1) introducimos un directorio válido con imágenes válidas y pulsamos el botón de “Submit”, el menú de selección de vuelo se actualizará mostrando la lista de vuelos encontrados, así como algunas opciones de análisis.

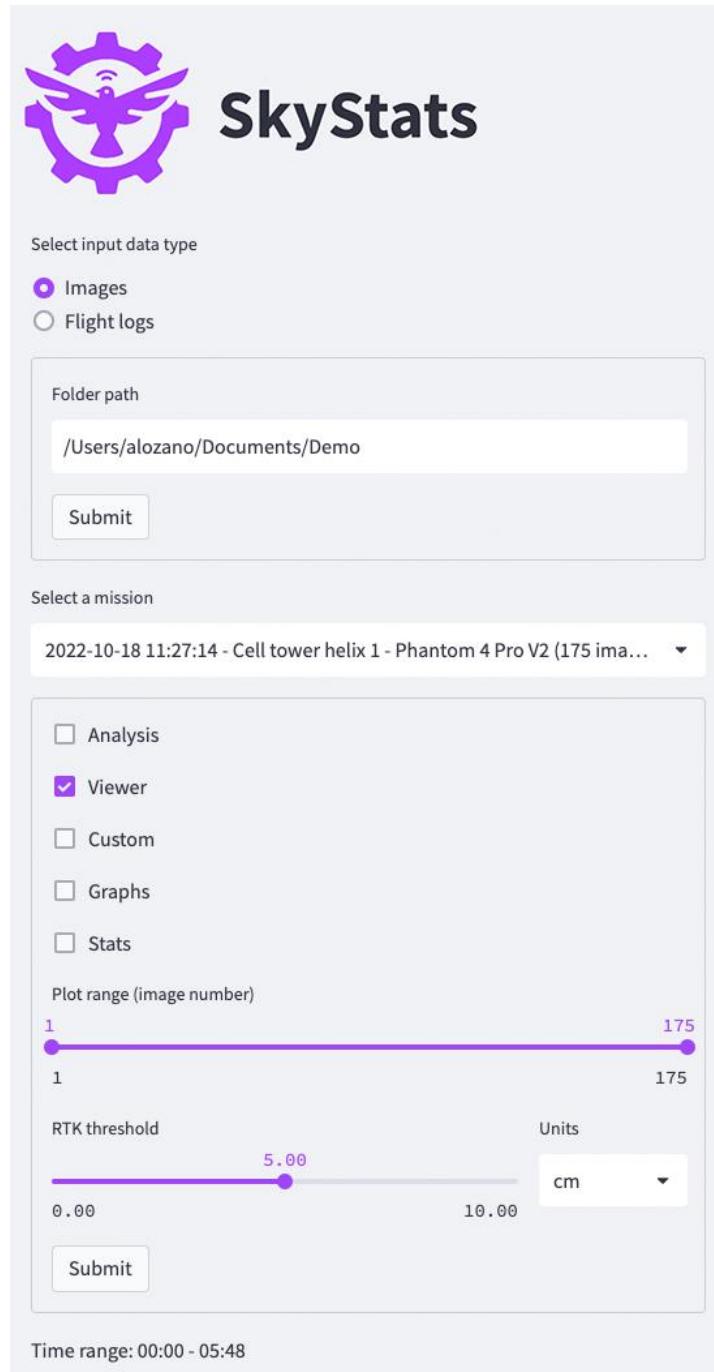


Figura 4.4.1.1: Menú de selección de vuelos y opciones de análisis

En primer lugar, tenemos un menú desplegable que permite seleccionar el vuelo que queremos analizar con el identificador explicado anteriormente. Los vuelos aparecen ordenados por fecha y hora, de más antiguo a más reciente.

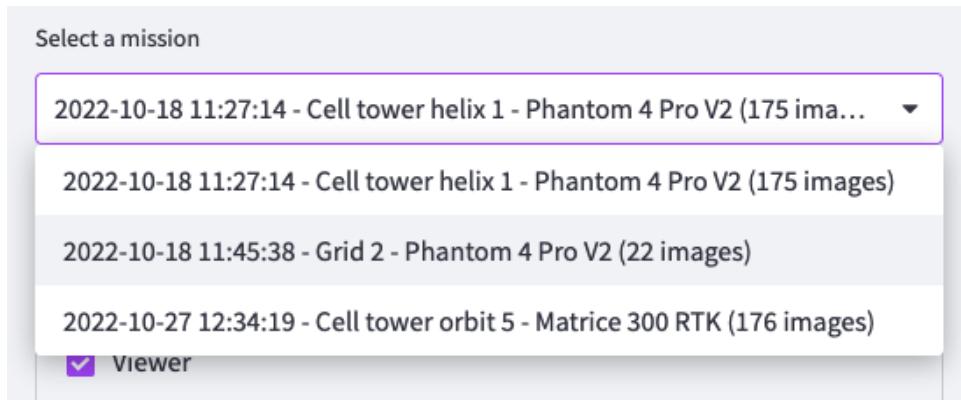


Figura 4.4.1.2: Menú desplegable de selección de vuelos (o misiones)

Cuando se seleccione un vuelo, se mostrará un mensaje indicando que se han cargado los datos correctamente. Este mensaje sirve de indicador para el usuario de que los datos de las imágenes se han terminado de extraer y procesar. Al tratarse de un mensaje de éxito, lo mostramos en color verde.

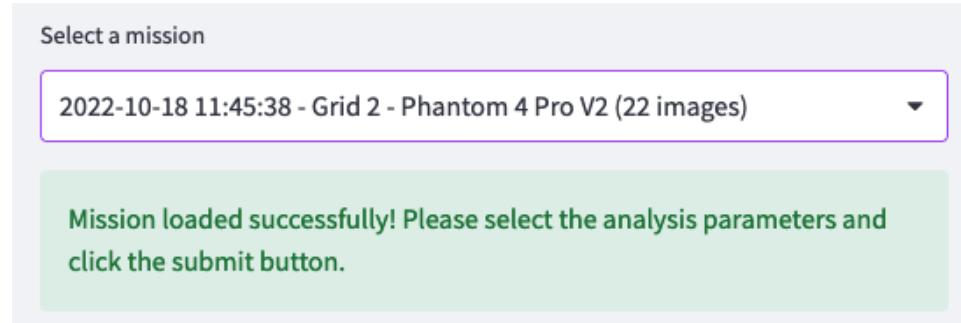


Figura 4.4.1.3: Mensaje que aparece al cargar con éxito las imágenes

Debajo de la selección de vuelos se nos muestran las opciones de análisis, que explicaremos en detalle en secciones posteriores. Ya que la opción del visor o “Viewer” es la más importante de la aplicación, aparece marcada por defecto para que el usuario pueda empezar a explorar sus datos inmediatamente después de seleccionar un vuelo. Al tratarse de checkboxes, podemos seleccionar varias opciones de análisis al mismo tiempo, por si el usuario quiere comparar la información que proporcionan distintos tipos de análisis. Sin embargo, debido a limitaciones del framework, es recomendable usar sólo una opción a la vez, ya que si se seleccionan varias, el rendimiento de la aplicación podría verse afectado.

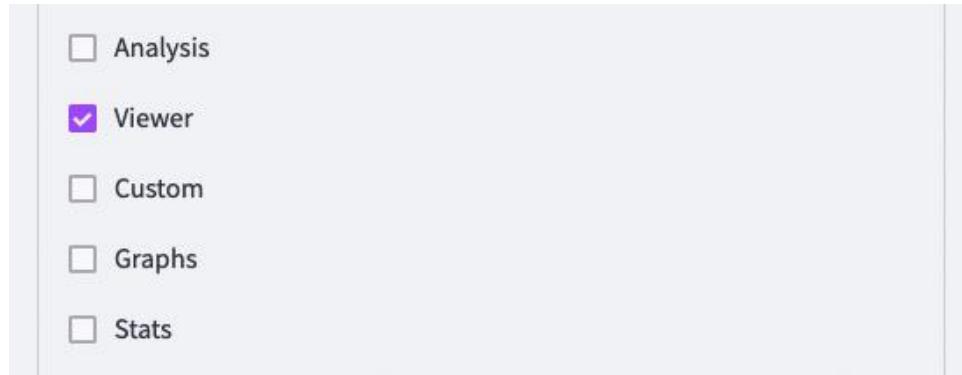


Figura 4.4.1.4: Opciones de análisis

El rango del análisis se puede ajustar gracias a un slider doble, en caso de que el usuario quiera analizar sólo una parte del vuelo. Por defecto, el rango de análisis es el vuelo completo, o sea, desde la primera imagen hasta la última. En la parte inferior de la barra lateral se muestra la equivalencia del rango seleccionado en minutos y segundos.

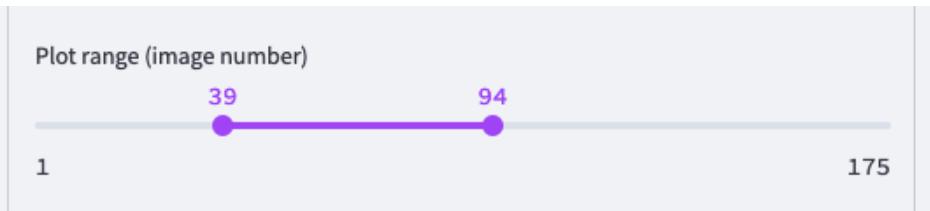


Figura 4.4.1.5: Slider del rango de análisis del vuelo. En esta misión de 175 imágenes sólo se analizarán desde la 39 hasta la 94

Para cuando trabajamos con imágenes de drones que han usado un sistema RTK, que se encarga de proporcionar una precisión de posicionamiento centimétrica, se puede ajustar el umbral de precisión de las mediciones del RTK con un slider que va desde 0.00 hasta 10.00 y un selector de unidades que permite elegir entre centímetros, decímetros y metros. Por defecto, el umbral de precisión es de 5 centímetros.

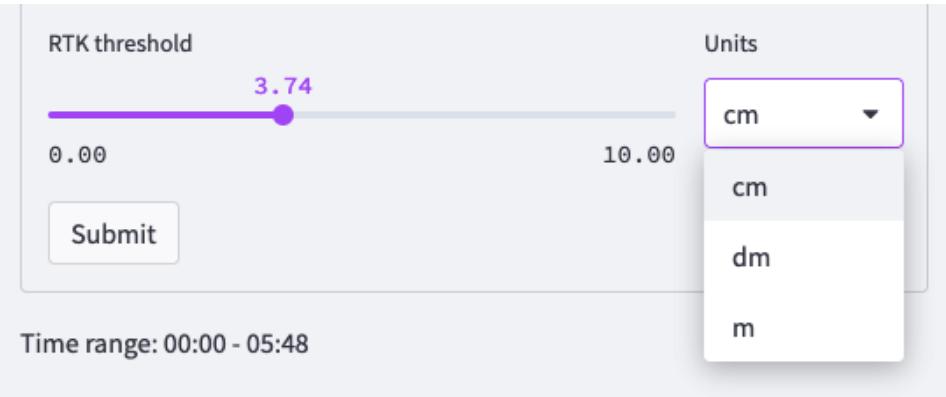


Figura 4.4.1.6: Slider del umbral de precisión RTK y el selector de unidades

Cuando trabajamos con datasets con muchas imágenes o con pantallas pequeñas que no permiten una selección precisa en los sliders, se puede ajustar su rango haciendo clic en ellos y desplazándolos usando las flechas izquierda y derecha del teclado.

Después de seleccionar un nuevo vuelo o de modificar los parámetros de análisis, es necesario pulsar el botón de “Submit” inferior para que los cambios se apliquen.

4.4.2 Logs

Al contrario que las imágenes los logs tienen un funcionamiento mucho más simple, ya que un vuelo produce siempre un único log.

Por lo tanto, el proceso a seguir para detectar los vuelos y mostrarlos como opciones en el menú de selección de vuelo es uno mucho más sencillo.

En primer lugar, recorremos recursivamente el directorio seleccionado en busca de archivos con el formato adecuado (“txt”). Una vez encontrados, se pasan como argumento a la función “check_valid_flight_log” de “flight_logs.py”, que se encarga de comprobar si el archivo es un log válido leyendo su primera línea y comprobando que se corresponde con la primera línea de un log de Pix4D, que siempre mostrará la línea “FMT, VERS, Time:double, Value:string” correspondiente a la especificación del formato de los logs.

Si el archivo es un log válido, se añade a la lista de logs de la clase “Workspace”, que contiene todos los logs encontrados en la carpeta seleccionada.

Al no tener un identificador único como en el caso de las imágenes ni contener información sobre el dron o la cámara, sólo se puede realizar un procesado mínimo del nombre de los logs para que sean más legibles de cara al usuario:

“light_log_helix.txt”: “2024-01-11 17:17:35 - flight_log_helix.txt - 526 KB”

Este nuevo formato proporciona al usuario la siguiente información que permite que las misiones sean fácilmente identificables entre ellas:

- Fecha y hora de la misión (en formato YYYY-MM-DD HH:MM:SS)
- Nombre del log (sin alterar)
- Tamaño del log (en KB)

De esta manera, el selector de vuelos quedaría así:

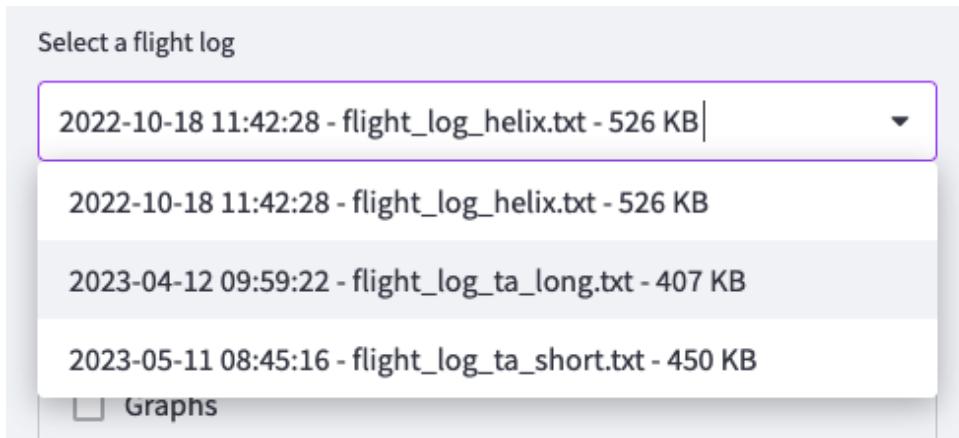


Figura 4.4.2.1: Menú desplegable de selección de vuelos

Las opciones de análisis son muy similares que las presentes en el caso de las imágenes, con las siguientes diferencias:

El rango de análisis funciona en base al tiempo de la misión en lugar de en base al número de imágenes, ya que los logs contienen información previa y posterior a la primera y última imagen.



Figura 4.4.2.2: Slider del rango de análisis del vuelo

Además, teniendo en cuenta que el usuario puede necesitar o no analizar ese vuelo anterior y posterior a la primera y última imagen, se ha añadido una checkbox que permite detectar ese tiempo extra, obtenido en base a los eventos de inicio y fin de misión que encontramos en los logs, e incluirlo o no en el rango de análisis.

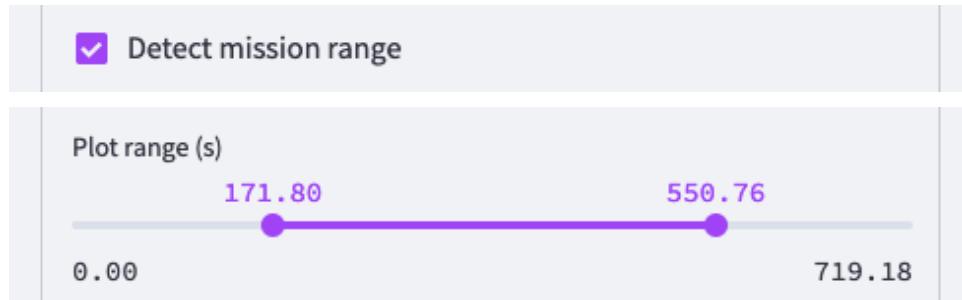


Figura 4.4.2.3: Slider del rango de análisis del vuelo al activar la detección del rango de misión

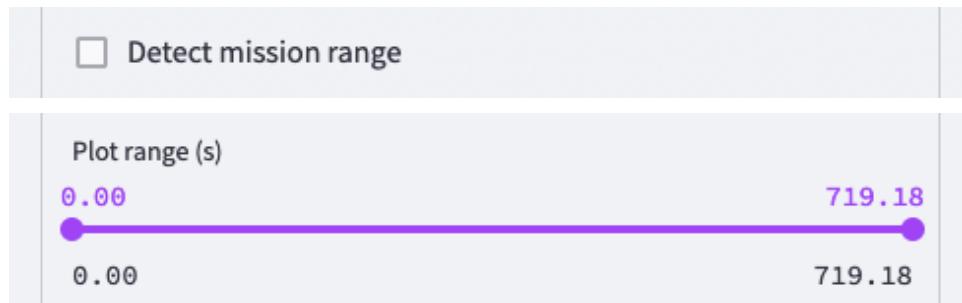


Figura 4.4.2.4: Slider del rango de análisis del vuelo al no activar la detección del rango de misión

Finalmente, puesto que los logs no contienen información sobre la desviación de la precisión del RTK, no se ha incluido la opción de ajustar el umbral de precisión ni el menú de “Analysis”, que sí están presentes en el caso de las imágenes.

4.5 Análisis básico

El análisis básico es una herramienta que permite al usuario comprobar si un vuelo cumple unos parámetros básicos, que le permitirán saber si el vuelo se ha realizado correctamente y si los datos son fiables antes de proceder a un análisis más profundo.

El análisis que realiza esta herramienta comprueba si se cumple la precisión mínima del sistema de posicionamiento RTK definida por el usuario, mostrando el número de imágenes que cumplen este umbral de precisión y su correspondiente porcentaje sobre el total de imágenes del vuelo.

Analysis

RTK

From a total of 176 images, 176 are below the threshold of 5.00 cm

This represents 100.00% of the total images

Figura 4.5.1: Funcionamiento del análisis básico. En este vuelo todas las imágenes están por debajo del umbral máximo de 5 cm de precisión RTK

Cabe destacar que, por falta de tiempo, no se han podido implementar todos los análisis que se querían implementar, como por ejemplo el análisis del número de satélites, o de la precisión de la cuadricula de imágenes en los vuelos de tipo grid, entre otros.

4.6 Visor interactivo

El visor es la herramienta central de la aplicación. Permite al usuario visualizar sus datos en entornos 2D y 3D interactivos y explorarlos en detalle, de una manera intuitiva y sencilla.

El visor tiene dos implementaciones distintas, una para imágenes y otra para logs, ya que cada uno de los dos tipos de datos tiene sus particularidades y requiere de un tratamiento distinto. Sin embargo, ambas implementaciones siguen la misma estructura y comparten la mayoría de sus funcionalidades, por lo que se explicarán conjuntamente en esta sección.

En el caso de las imágenes, la implementación se hace en la clase “ImagePlots” dentro del fichero “image_plots.py”, mientras que en el caso de los logs, la implementación se hace en la clase “FlightLogPlots” dentro del fichero “flight_log_plots.py”.

En ambos casos, la clase se inicializa con los datos del vuelo seleccionado, obtenidos según se explicó en la sección 4.3 sobre la extracción y procesado de datos. Después, se crea una estructura de datos “all_data_properties” conformada por una lista de diccionarios, que contiene información sobre cada una de las variables que se pueden visualizar en el visor, como su nombre, el tipo de datos que contiene, su unidad, etc. Esta estructura de datos no se utilizará para las visualizaciones, si no que se utilizará para generar los menús de opciones y de selección de datos del visor. Podemos ver un extracto de “all_data_properties” a continuación:

```

self.all_data_properties = [
    {'name': 'timestamp', 'data': self.timestamp, 'type': 'time',
     'label': 'Timestamp', 'unit': '(s)'},
    ...
    {'name': 'speed_x', 'data': self.speed_x, 'type': 'speed',
     'label': 'Latitude Speed', 'unit': '(m/s)'},
    ...
    {'name': 'flight_pitch', 'data': self.flight_pitch, 'type': 'gim',
     'label': 'Flight Pitch', 'unit': '(°)'},
    ...
    {'name': 'rtk_threshold_check', 'data': self.rtk_threshold_check,
     'type': 'rtk3', 'label': 'RTK Threshold Check',
     'unit': '(boolean)'},
]

```

Una vez inicializada la clase y dependiendo del tipo de visor seleccionado, se llamará a la función correspondiente para generar las visualizaciones. Todas las funciones de visualización toman todos los datos disponibles del vuelo junto a los parámetros seleccionados en el visor como argumentos, que usan para generar las visualizaciones.

En el caso del visor 3D, se llama a la función “plotter_3d_plotly”, que tiene el mismo nombre en las dos clases. Esta es la visualización 3D principal, que hace uso de la librería Plotly para generar un entorno 3D interactivo.

En el caso del visor de mapa 2D, se llama a la función “plotter_map_plotly”, que también tiene el mismo nombre en las dos clases. Esta visualización también hace uso de la librería Plotly, pero en este caso generando una visualización 2D interactiva desde una vista cenital, de forma similar a Google Maps, y por tanto sin tener en cuenta la altitud de los datos. Se aprovecha la capacidad de Plotly de conectarse con la API de Mapbox muy fácilmente con solo proporcionar el token de usuario, para darle al usuario la posibilidad de elegir entre distintos mapas de Mapbox sobre los que superponer los datos.

Finalmente, tenemos las visualizaciones 3D estáticas, que se generan llamando a las funciones “plot_3d_ortho”, “plot_3d_top”, “plot_3d_front” y “plot_3d_side”, que generan visualizaciones 3D estáticas desde distintos puntos de vista, y que se han implementado usando la librería Matplotlib. Estas visualizaciones se desarrollaron antes de considerar la posibilidad de usar Plotly para generar visualizaciones 3D interactivas, y aunque no son tan completas como las generadas con Plotly, se han mantenido en la aplicación ya que pueden ser útiles en algunos casos, como por ejemplo cuando se quieran generar imágenes para incluir en un informe o documento.

En las siguientes subsecciones explicaremos en detalle las funcionalidades de cada una de las visualizaciones, primero para el caso de las imágenes y después para el caso de los logs, ya que estos últimos tienen algunas funcionalidades adicionales.

4.6.1 Imágenes

La interfaz del visor, definida en su totalidad dentro del fichero “image_plots_ui.py” haciendo uso de Streamlit, es la siguiente:

Viewer

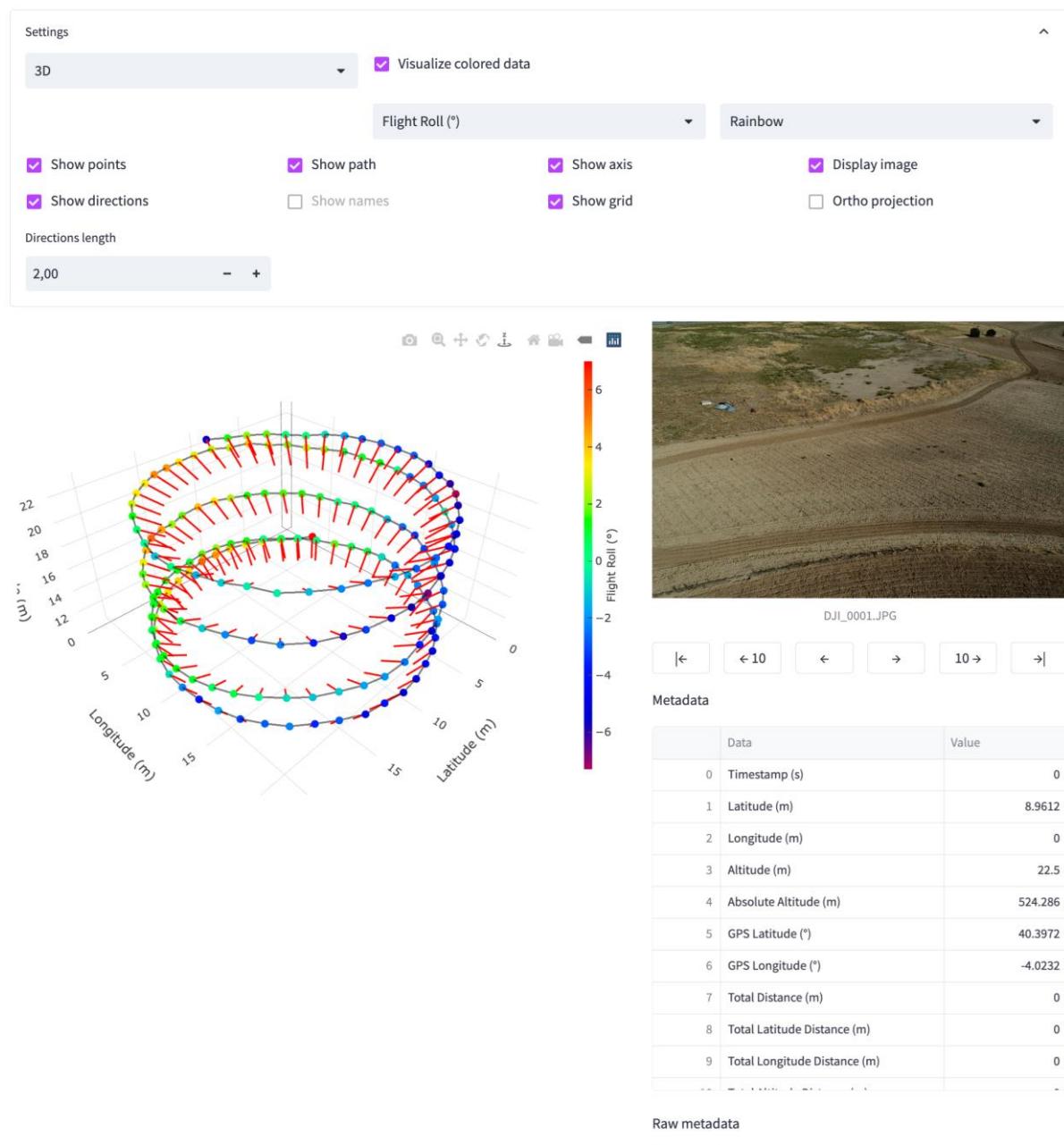


Figura 4.6.1.1: Interfaz del visor

Como se puede apreciar, la interfaz muestra en la parte inferior izquierda la visualización interactiva del vuelo, y en la parte inferior derecha se muestra el visor de imágenes junto a una tabla con sus metadatos.

Mientras tanto, en la parte superior tenemos el menú de opciones, compuesto por menús desplegables, checkboxes y un cuadro de entrada numérica. Estos elementos permiten al usuario ajustar la visualización a sus necesidades. Este menú puede minimizarse en caso de que no se necesite haciendo clic en la parte superior del mismo.

A continuación, explicaremos en detalle cada uno de los elementos de la interfaz.

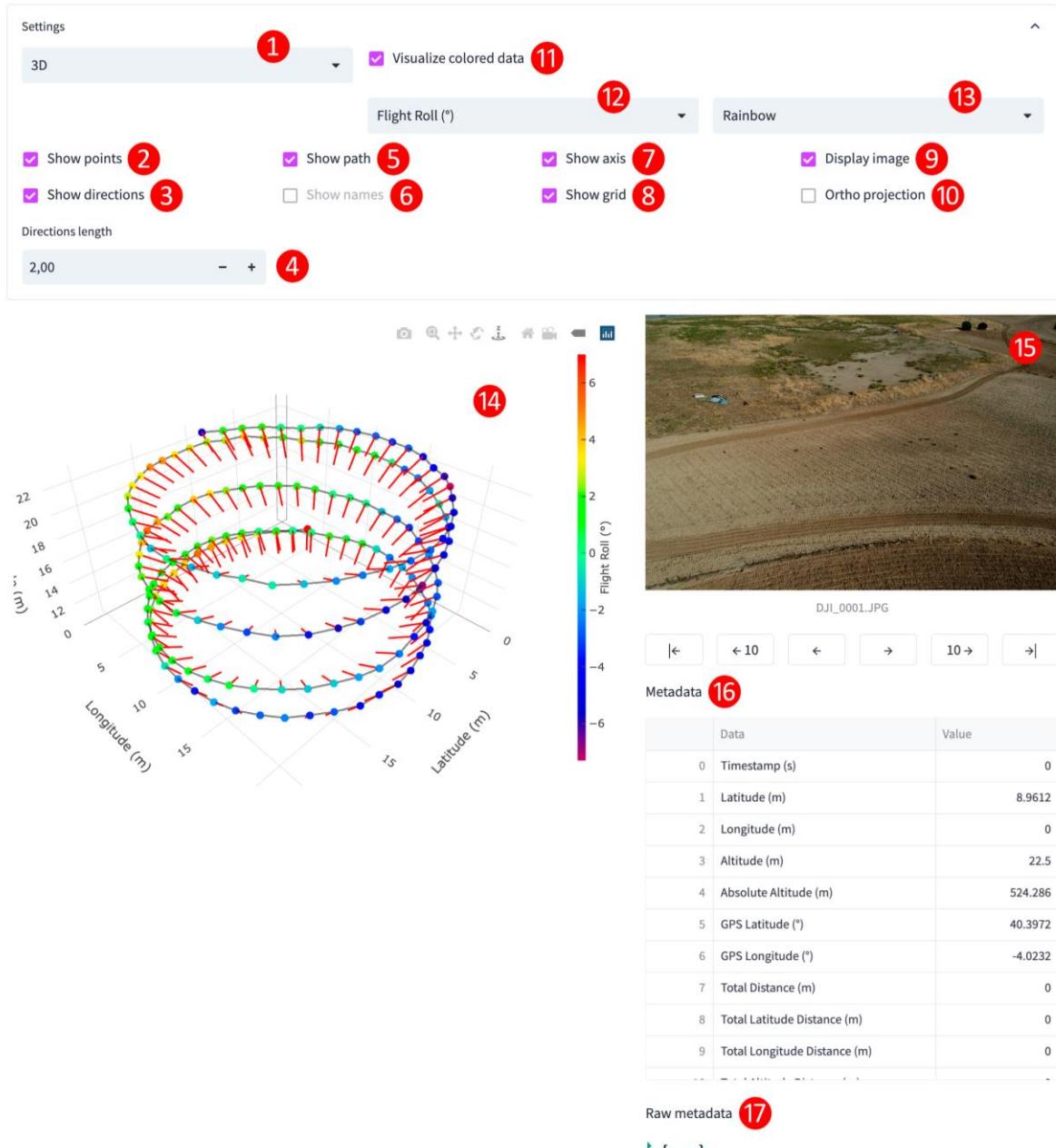


Figura 4.6.1.2: Interfaz del visor con elementos numerados para relacionarlos con las secciones siguientes

1. Mostrar puntos (Show points): Este menú desplegable permite al usuario elegir entre las tres visualizaciones disponibles: Visor 3D, Visor 3D estático y Visor de mapa 2D.

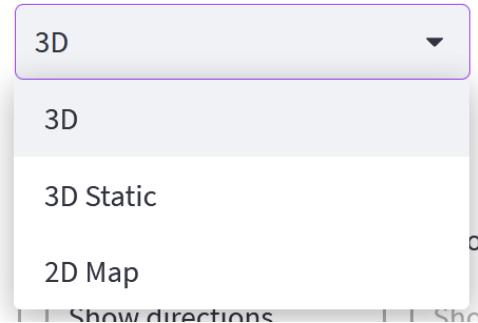


Figura 4.6.1.3: Interfaz del visor

- Visor 3D (3D): Esta opción permite al usuario usar el visor 3D interactivo, que permite visualizar los datos en un entorno 3D versátil y fácilmente navegable.

Esta es la opción por defecto.

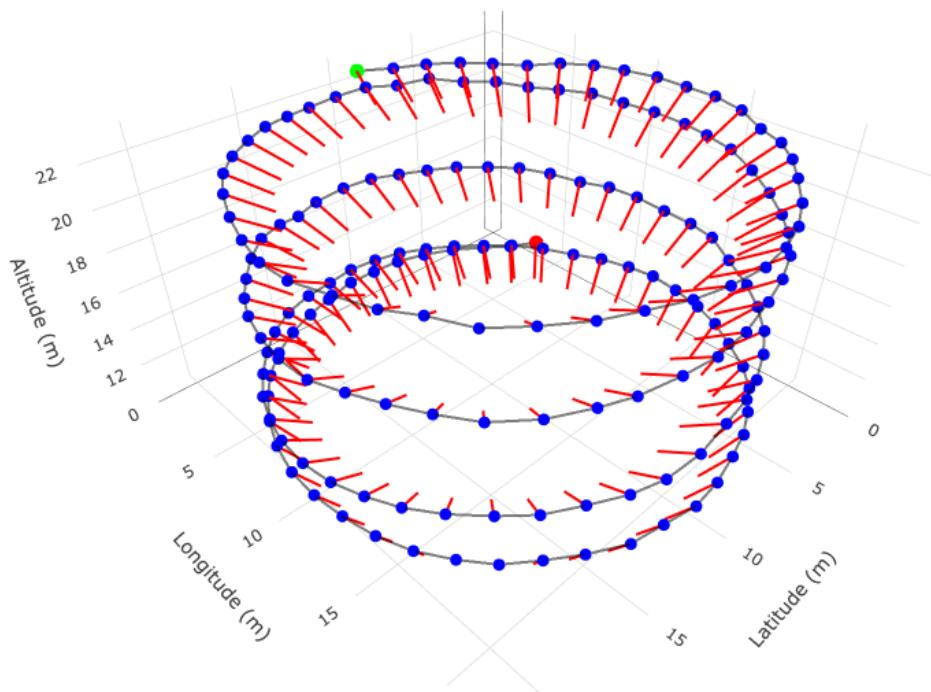


Figura 4.6.1.4: Visor 3D

- Visor 3D estático (3D Static): Esta opción permite al usuario usar el visor 3D estático, que genera una visualización estática en 3D de los datos desde 4 puntos de vista predefinidos.

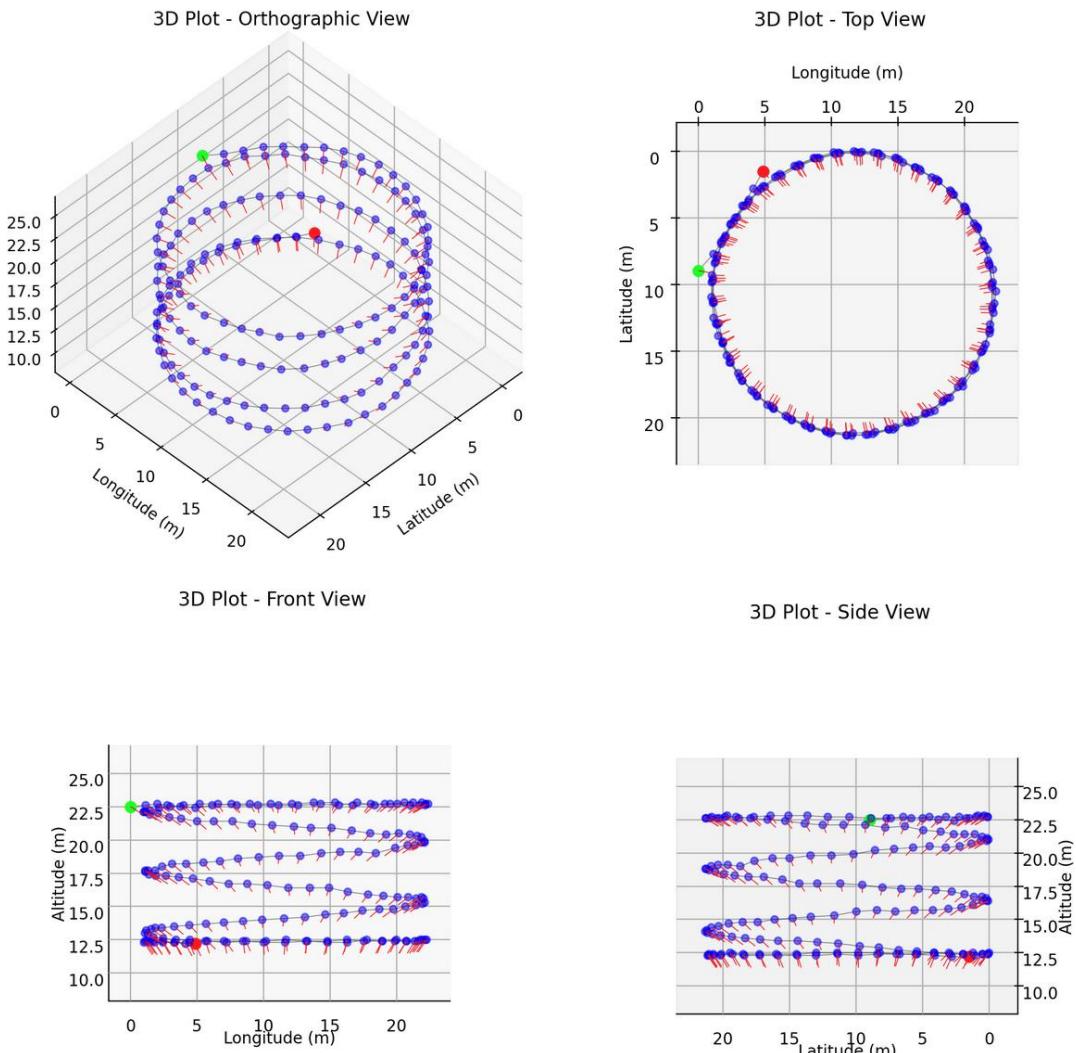


Figura 4.6.1.5: Visor 3D estático

- Visor de mapa 2D (2D Map): Esta opción permite al usuario usar el visor de mapa 2D interactivo.

Este visor requiere de una conexión a internet para poder cargar el mapa. En caso de que no haya conexión, esta visualización seguirá funcionando, pero no se mostrará el mapa en el fondo.

Los estilos de mapas soportados junto a sus proveedores son los siguientes:

- Mapbox: Outdoors, Satellite, Streets, Satellite + Streets, Light, Dark, Basic
- OpenStreetMap: OpenStreetMap
- Carto: Positron, Dark matter
- Stamen: Terrain

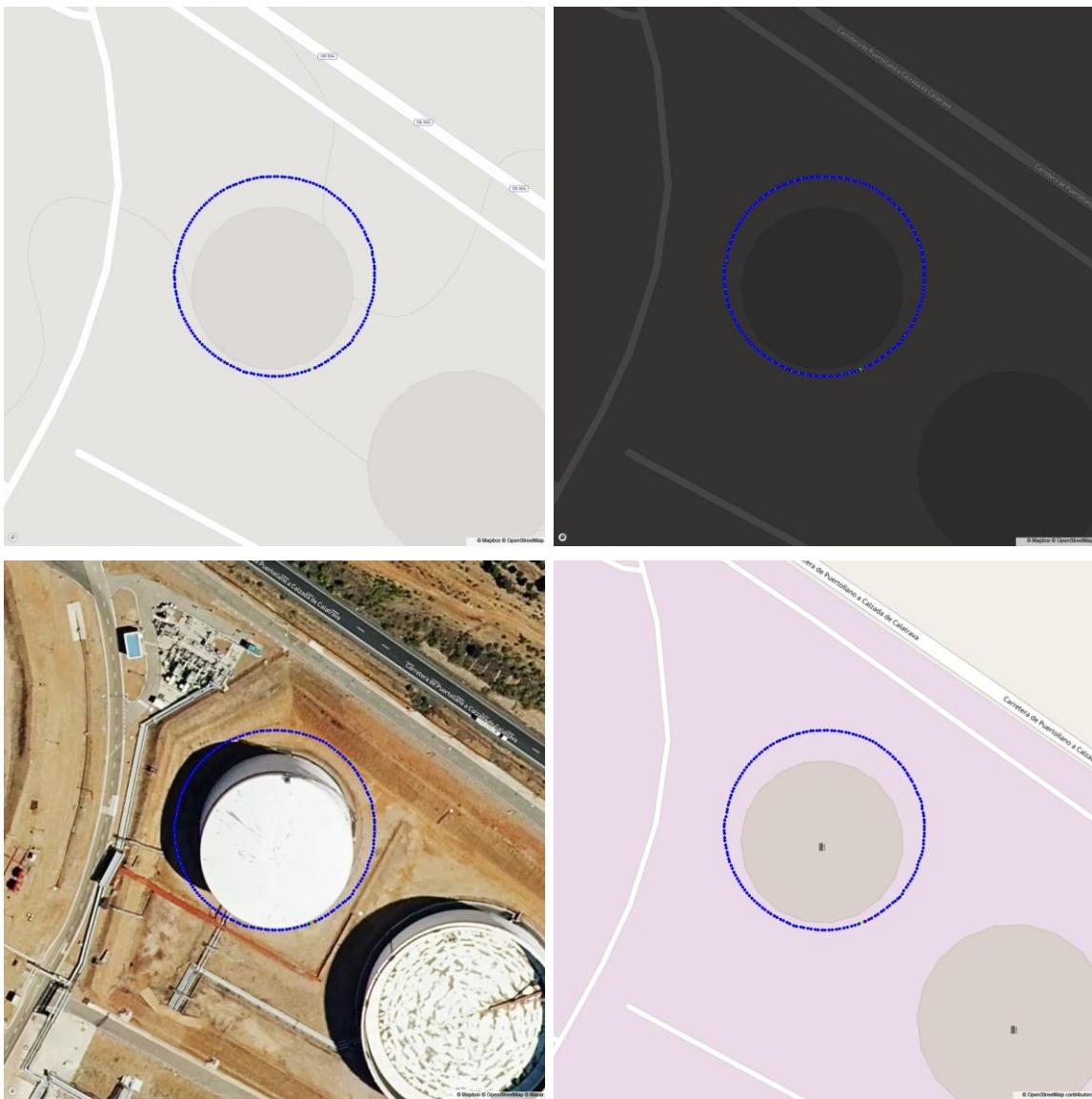


Figura 4.6.1.6, 4.6.1.7, 4.6.1.8 y 4.6.1.9: Visor de mapa 2D. De izquierda a derecha, de arriba a abajo: Mapbox Outdoors, Mapbox Dark, Mapbox Satellite + Streets, OpenStreetMap

2. Mostrar puntos (Show points): Esta opción permite al usuario elegir si quiere mostrar los puntos correspondientes a las imágenes en la visualización 3D, permitiendo así visualizar la posición de las imágenes en el espacio.

Para facilitar la comprensión de la visualización, los puntos se muestran en tres colores distintos:

- Verde: Primera imagen del vuelo
- Rojo: Última imagen del vuelo
- Azul: Resto de imágenes

Por defecto, esta opción está activada.

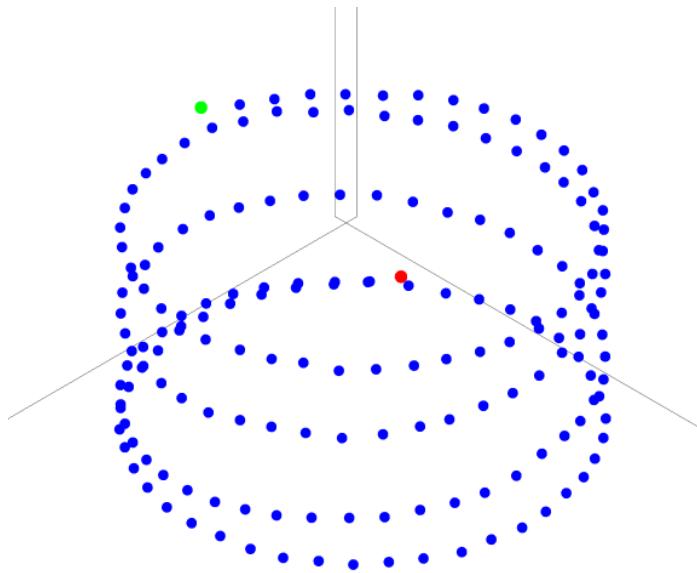


Figura 4.6.1.10: Visor 3D solo con la opción “Mostrar puntos” activada

3. Mostrar direcciones (Show directions): Esta opción permite al usuario elegir si quiere mostrar las direcciones de la cámara desde los puntos correspondientes a las imágenes en la visualización 3D, permitiendo así visualizar la dirección en la que apuntaba la cámara en el momento de la captura de la imagen.

Las líneas de las direcciones siempre aparecerán en color rojo.

Esta opción no está disponible en el visor de mapa 2D.

Por defecto, esta opción está activada.

Estas direcciones consisten en un vector que obtenemos a partir de los ángulos de orientación de la cámara (yaw, pitch y roll) que se encuentran en los metadatos de las imágenes.

Debido a que los conceptos de roll, pitch y yaw pueden resultar poco intuitivos, se adjunta a continuación una imagen que los explica de forma gráfica:

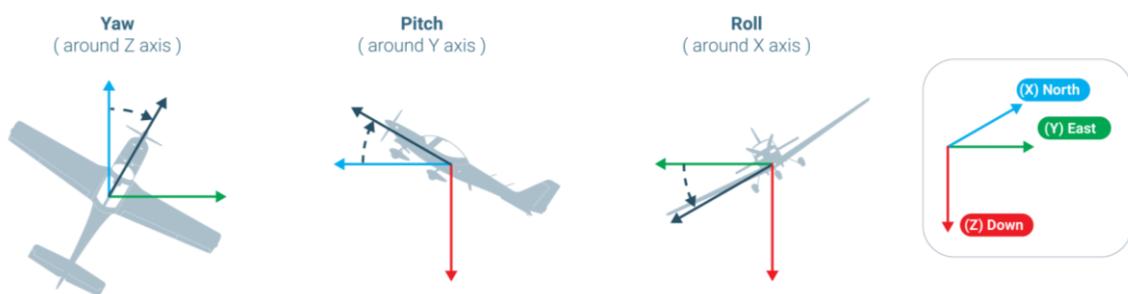


Figura 4.6.1.11: Visualización del funcionamiento de los ángulos de orientación yaw, pitch y roll [38]

El cálculo de los vectores de dirección comienza por la conversión de los ángulos de rotación de grados a radianes:

```
yaw_rad = deg2rad(gimbal_yaw)  
pitch_rad = deg2rad(gimbal_pitch)
```

Y la aplicación de las siguientes fórmulas, que son las adecuadas para la convención específica de ángulos que estamos usando [39]:

```
x_vector = sin(yaw_rad) * cos(pitch_rad)  
y_vector = cos(yaw_rad) * cos(pitch_rad)  
z_vector = sin(pitch_rad)
```

No usamos el roll para este cálculo, ya que suele tener un valor de 0 o muy cercano a 0 debido al bloqueo del gimbal o cardán, una restricción mecánica que impide que la cámara se mueva en el eje de roll [40].

Después, usamos estos vectores para calcular el punto final de la dirección en base al parámetro de longitud de la dirección:

Punto inicial (Posición del dron):

```
(x_start, y_start, z_start) = (x_pos[i], y_pos[i], z_pos[i])
```

Punto final (Dirección de la cámara extendida desde la posición del dron):

```
x_end = x_pos[i] + direction_length * x_vector  
y_end = y_pos[i] + direction_length * y_vector  
z_end = z_pos[i] + direction_length * z_vector
```

Dirección (Vector que une el punto inicial con el punto final):

```
(x_start, y_start, z_start) -> (x_end, y_end, z_end)
```

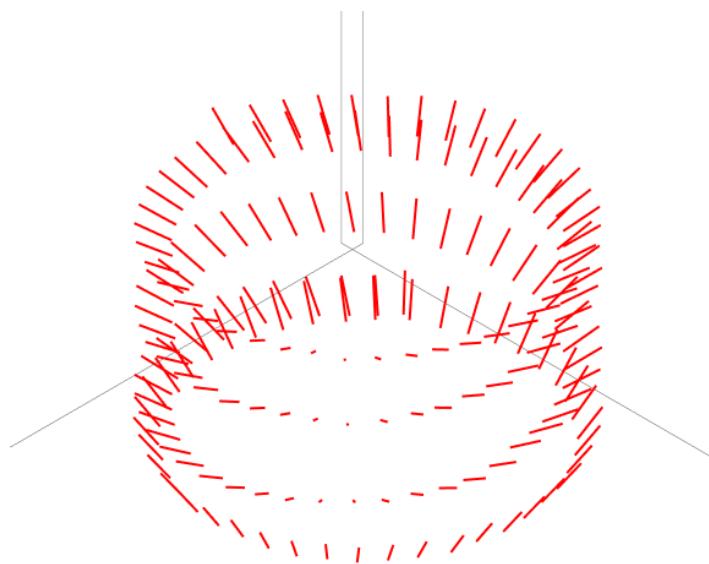


Figura 4.6.1.12: Visor 3D solo con la opción “Mostrar direcciones” activada

4. Longitud de las direcciones (Directions length): Este cuadro de entrada numérica permite al usuario ajustar la longitud de las direcciones de la cámara en la visualización 3D mencionadas en el punto anterior.

Por defecto, la longitud de las direcciones es de 2 unidades.

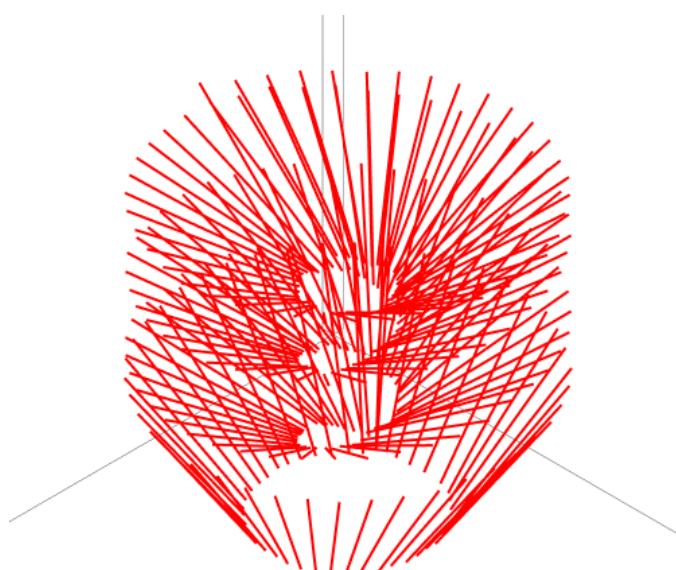


Figura 4.6.1.13: Visor 3D solo con la opción “Mostrar direcciones” activada y direcciones con 10 unidades de longitud

5. Mostrar recorrido (Show path): Esta opción permite al usuario elegir si quiere mostrar el recorrido del dron en la visualización 3D, mostrando una línea que une los puntos correspondientes a las imágenes en el orden en el que fueron capturadas.

La linea del recorrido siempre aparecerá en color gris.

Por defecto, esta opción está activada.

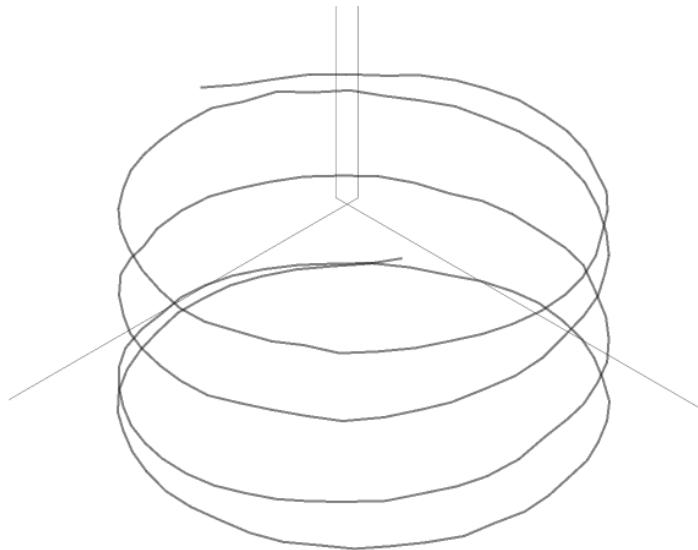


Figura 4.6.1.14: Visor 3D solo con la opción “Mostrar recorrido” activada

6. Mostrar nombres (Show names): Esta opción permite al usuario elegir si quiere mostrar los nombres de las imágenes en la visualización 3D, permitiendo así identificarlas fácilmente.

Esta opción solo está disponible en la visualización 3D estática, ya que en la visualización 3D interactiva y en el visor de mapa 2D se puede obtener el nombre de las imágenes pasando el cursor por encima de los puntos correspondientes.

Por defecto, esta opción está desactivada.

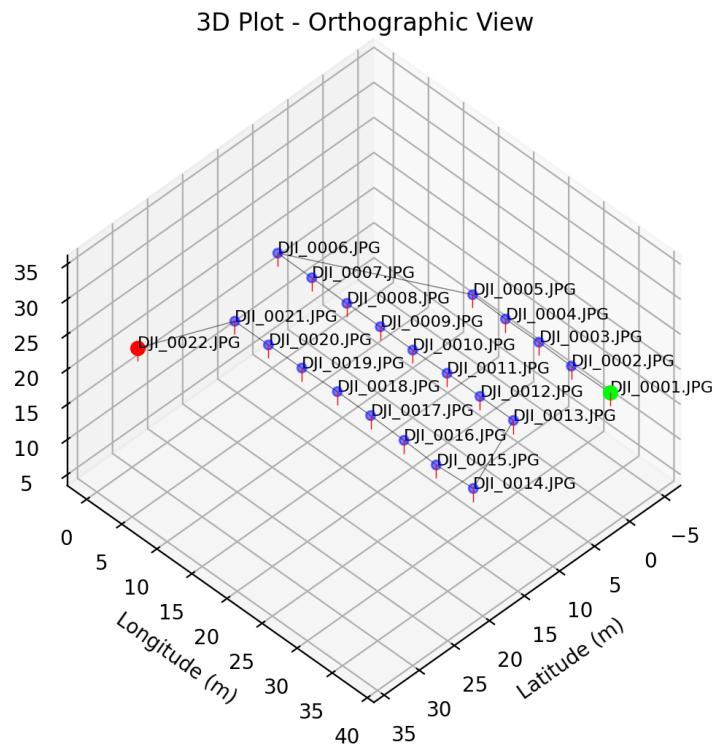


Figura 4.6.1.15: Visor 3D con la opción “Mostrar nombres” activada

7. Mostrar leyendas en los ejes (Show axis): Esta opción permite al usuario elegir si quiere mostrar las leyendas de los ejes en la visualización, que permiten saber qué variable (altitud, latitud o longitud) se está mostrando en cada eje junto a sus valores en metros.

Por defecto, esta opción está activada.

Esta opción no está disponible en el visor de mapa 2D, ya que no tiene ejes.

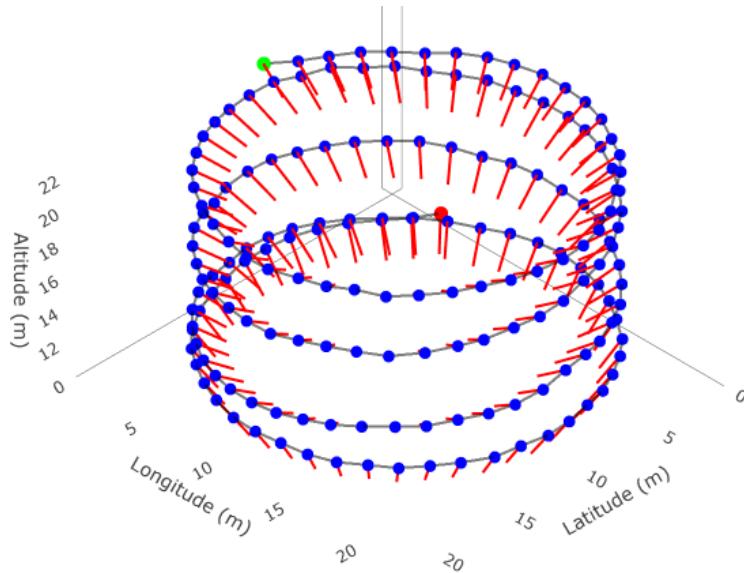


Figura 4.6.1.16: Visor 3D con la opción “Mostrar leyendas en los ejes” activada

8. Mostrar grid (Show grid): Esta opción permite al usuario elegir si quiere mostrar el grid en la visualización, que ayuda a ubicar los puntos en el espacio. Por defecto, esta opción está activada.
Esta opción no está disponible en el visor de mapa 2D, ya que no tiene ejes.

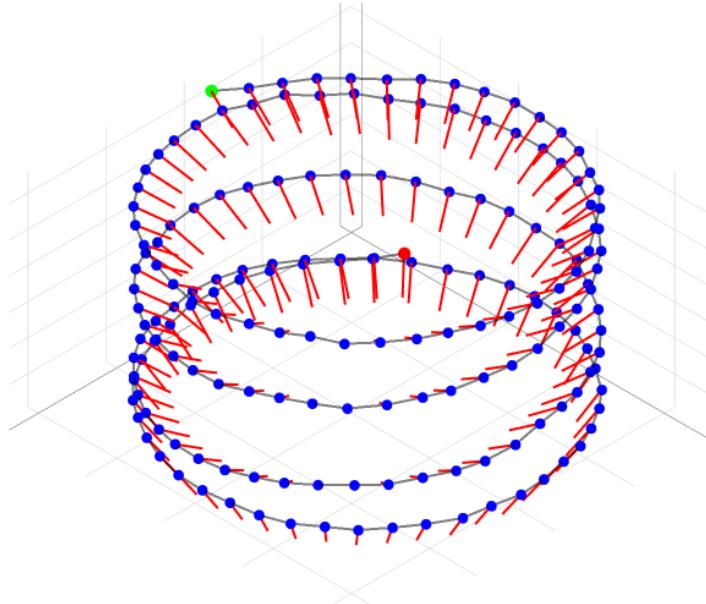


Figura 4.6.1.17: Visor 3D con la opción “Mostrar grid” activada

9. Mostrar imagen (Display image): Esta opción permite al usuario elegir si quiere mostrar el visor de imágenes y su tabla de metadatos en la parte inferior derecha de la interfaz.
Por defecto, esta opción está activada.
En caso de desactivarse, el visor de imágenes y su tabla de metadatos se ocultarán, y la visualización 3D ocupará todo el espacio disponible.
10. Proyección ortogonal (Orthogonal projection): Esta opción permite al usuario elegir si quiere usar una proyección ortogonal en la visualización 3D. En caso de desactivarse, se usará una proyección en perspectiva.
El visor 3D estático siempre usa una proyección ortogonal.
Esta opción no está disponible en el visor de mapa 2D, ya que la proyección en perspectiva no tiene sentido en una vista cenital.
Por defecto, esta opción está desactivada, ya que la proyección en perspectiva permite apreciar mejor la profundidad de la visualización.

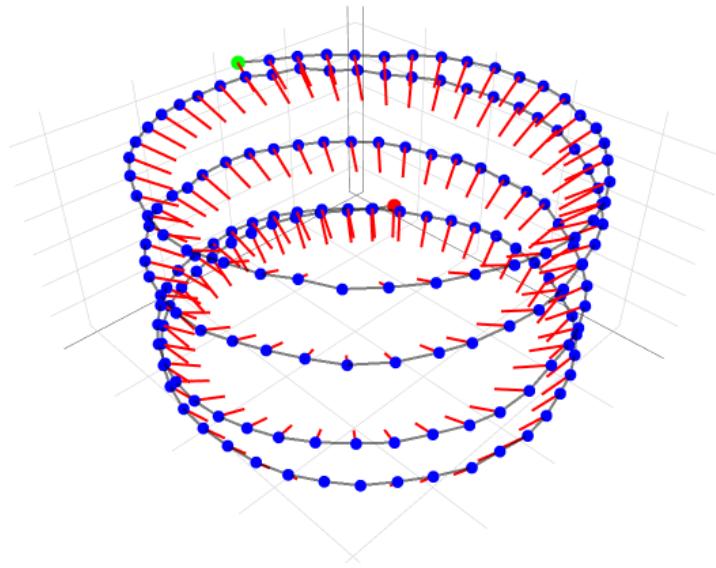


Figura 4.6.1.18: Visor 3D con la opción “Proyección ortogonal” desactivada

11. Visualizar datos coloreados (Visualize colored data): Esta opción permite al usuario cambiar el color por defecto de los puntos de la visualización 3D por colores que representan los valores de una variable seleccionada por el usuario. Esta opción tiene una gran utilidad al analizar los datos, ya que permite visualizarlos de forma mucho más intuitiva y fácil de interpretar.
- Mientras esta opción esté activada, se mostrarán dos menús desplegables en la parte superior de la interfaz, que permiten al usuario seleccionar la variable que quiere visualizar y la gama de colores que quiere usar para representarla. Además, se mostrará una leyenda en la parte derecha del visor 3D que indica el valor de cada color y cuando se pasa el cursor por encima de un punto, se mostrará un tooltip con el valor de la variable en ese punto en vez del nombre de la imagen.
- Esta opción no está disponible en el visor 3D estático.

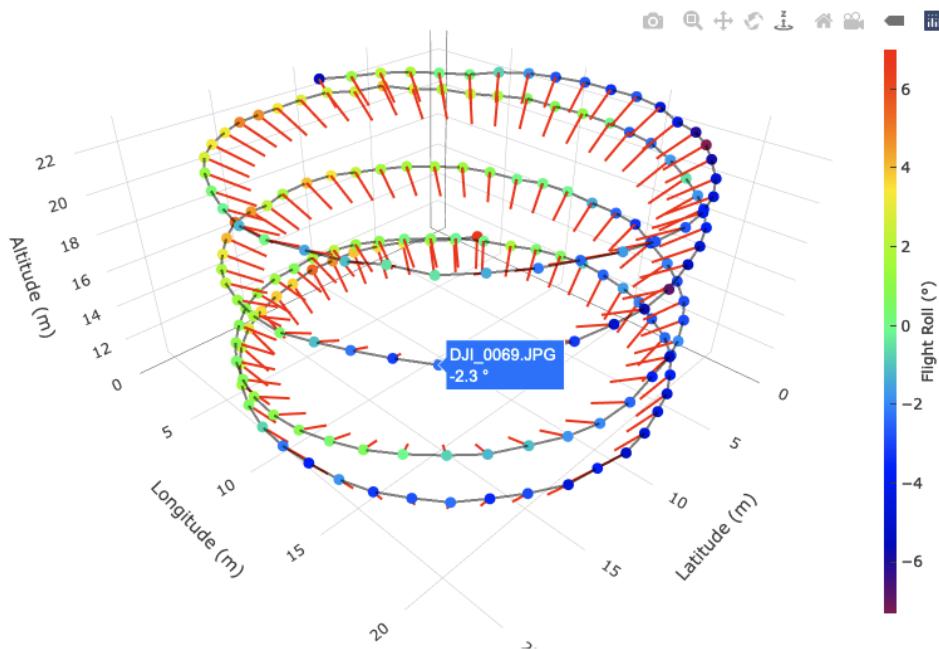


Figura 4.6.1.19: Visor 3D con la opción “Visualizar datos coloreados” activada. Concretamente, se está visualizando el dato “flight roll” con la gama de colores “rainbow”. Se puede ver el valor del “flight roll” de la imagen sobre la que se tiene el cursor.

12. Lista de datos para colorear: Este menú desplegable permite al usuario seleccionar las variables disponibles para visualizar en el visor 3D, junto a su unidad.

Esta lista se genera a partir de la estructura de datos “all_data_properties” mencionada anteriormente, y soporta las siguientes variables:

- Timestamp (s)
- Latitude , Longitude, Altitude, Absolute Altitude (m)
- GPS Latitude, GPS Longitude (°)
- Total Distance, Total Latitude Distance, Total Longitude Distance, Total Altitude Distance (m)
- Speed, Latitude Speed, Longitude Speed, Altitude Speed (m/s)
- Acceleration, Latitude Acceleration, Longitude Acceleration, Altitude Acceleration (m/s²)
- Gimbal Pitch, Gimbal Roll, Gimbal Yaw (°)
- Flight Pitch, Flight Roll, Flight Yaw (°)
- RTK Std. Deviation Latitude, RTK Std. Deviation Longitude, RTK Std. Deviation Altitude, RTK Std. Deviation Total (cm)
- RTK Difference Age (s)
- RTK Flag (status)
- RTK Threshold Check (boolean)

13. Lista de Listas de gamas de colores: Este menú desplegable permite al usuario seleccionar la gama de colores que quiere usar para representar la variable seleccionada en el menú anterior. La lista contiene las siguientes gamas de colores:

- Rainbow
- Viridis
- Bluered
- Plasma
- Inferno

14. Visor interactivo: Como hemos mencionado anteriormente, el visor interactivo es la visualización principal de SkyStats, que nos permite ver los datos en entornos 3D y 2D interactivos.

La visualización 3D estática generada con Matplotlib, produce cuatro imágenes estáticas, por lo que la interactividad es nula.

Sin embargo, las visualizaciones 3D y de mapa 2D generadas con Plotly permiten al usuario interactuar con ellas de las siguientes maneras:

- Movimiento de la cámara: El usuario puede mover la cámara en la visualización 3D interactiva haciendo clic izquierdo y arrastrando, rotar la cámara haciendo clic derecho y arrastrando, y hacer zoom con la rueda del ratón.
- Guardar imagen: El usuario puede guardar una imagen de la visualización 3D interactiva haciendo clic en el botón con el icono de cámara que aparece en la parte superior derecha de la visualización.
- Información al pasar el ratón por encima: El usuario puede obtener información sobre los puntos de la visualización 3D interactiva haciendo pasar el cursor por encima de ellos. Por defecto se muestra el nombre de la imagen, pero si se ha seleccionado una variable para colorear los datos, se mostrará el valor de la variable en ese punto.
- Información al hacer clic: Usando la librería “streamlit_plotly_events”, podemos crear un callback que se ejecuta cuando el usuario hace clic en un punto de las visualizaciones interactivas. Al hacer clic en una imagen, esta se mostrará en el visor de imágenes junto a sus metadatos.

15. Visor de imágenes: Esta ventana muestra la imagen seleccionada en la visualización interactiva, o en caso de que no se haya seleccionado ninguna, muestra la primera imagen del vuelo.

En caso de que se desee navegar por las imágenes del vuelo, se puede hacer uso de los botones de la parte inferior de la ventana, que permiten avanzar y retroceder una o diez imágenes, o ir directamente a la primera o última imagen del vuelo.

16. Tabla de metadatos transformados: Esta tabla muestra los metadatos procesados y transformados de la imagen seleccionada en el visor de imágenes. Los datos mostrados son los especificados en “all_data_properties”, los mismos con los que se pueden colorear los datos en las visualizaciones interactivas.

17. Metadatos sin procesar (raw): Este desplegable muestra los mismos datos que obtuvimos en el apartado 5.3.1 sobre la extracción de metadatos de imágenes, sin realizar ningún tipo de procesado ni transformación.

```
Raw metadata
{
  "Filename" :
    "C:\home\documents\education
    \uni\uni_2023_2024\tfg\Demo\DJ1_0037.JPG"
  ▶ "Image Size" : [...]
  "Image Height" : 5460
  "Image Width" : 8192
  "Image Format" : "MPO"
  "Image Mode" : "RGB"
  "Image Latitude Ref" : "N"
  ▶ "Image Latitude" : [...]
  "Image Longitude Ref" : "W"
  ▶ "Image Longitude" : [...]
  "Image Altitude" : "789.5"
  "Image is Animated" : true
  "Frames in Image" : 2
  "DateTime" : "2022:10:27 12:58:41"
  "File Size" : 5582570
```

Figura 4.6.1.20: Extracto del menú de metadatos sin procesar

4.6.2 Logs

La interfaz del visor de logs es la esencialmente la misma que la del visor de imágenes. Simplemente usamos un condicional que nos indica si el usuario ha seleccionado como datos de entrada imágenes o logs, y en función de ello llamamos a las funciones correspondientes para generar las visualizaciones.

Debido a las diferencias entre los datos de imágenes y los datos de logs, hay algunas diferencias en las opciones de análisis disponibles en el visor de logs, que explicaremos a continuación:

En primer lugar, cuando usamos logs como fuente de datos, el visor de imágenes desaparece, puesto que los logs no contienen esa información. La tabla de metadatos se mantiene, ya que sí que podemos conocer la información en el momento de la captura de las imágenes.

Además, los datos procesados de “all_data_properties” que definen las posibles opciones de coloreado de datos son ligeramente distintas:

- Timestamp (s)
- GPS Latitude, GPS Longitude (°)
- Latitude, Longitude, Altitude (rel. to takeoff) (m)
- Total Distance Latitude, Total Distance Longitude, Total Distance Altitude, Total Distance (m)
- Flight Pitch, Flight Roll, Flight Yaw (°)
- Gimbal Pitch, Gimbal Roll, Gimbal Yaw (°)
- Velocity Latitude, Velocity Longitude, Velocity Altitude, Velocity (m/s)
- Drone Battery, Remote Battery (%)
- Link Quality (%)
- Free Storage (MB)
- Connected satellites (n)
- Exposure Compensation (EV)

Por otra parte, el visor de logs ofrece una representación más precisa de los vuelos. Mientras que en el caso de las imágenes, tenemos tantas ubicaciones del recorrido del dron como imágenes, en el caso de los logs tenemos un registro de la ubicación espacial del dron varias veces por segundo, por lo que la representación del recorrido es mucho más precisa.

De la misma manera, los logs contienen información sobre el vuelo previo y posterior a la primera y última imagen, por lo que ahora el rango efectivo de análisis es el vuelo completo, desde el despegue hasta el aterrizaje, en lugar de desde la primera imagen hasta la última.

Haciendo uso de la opción de selección del rango de análisis automático explicado en la sección 4.4.2, podemos ajustar la visualización de la siguiente manera:

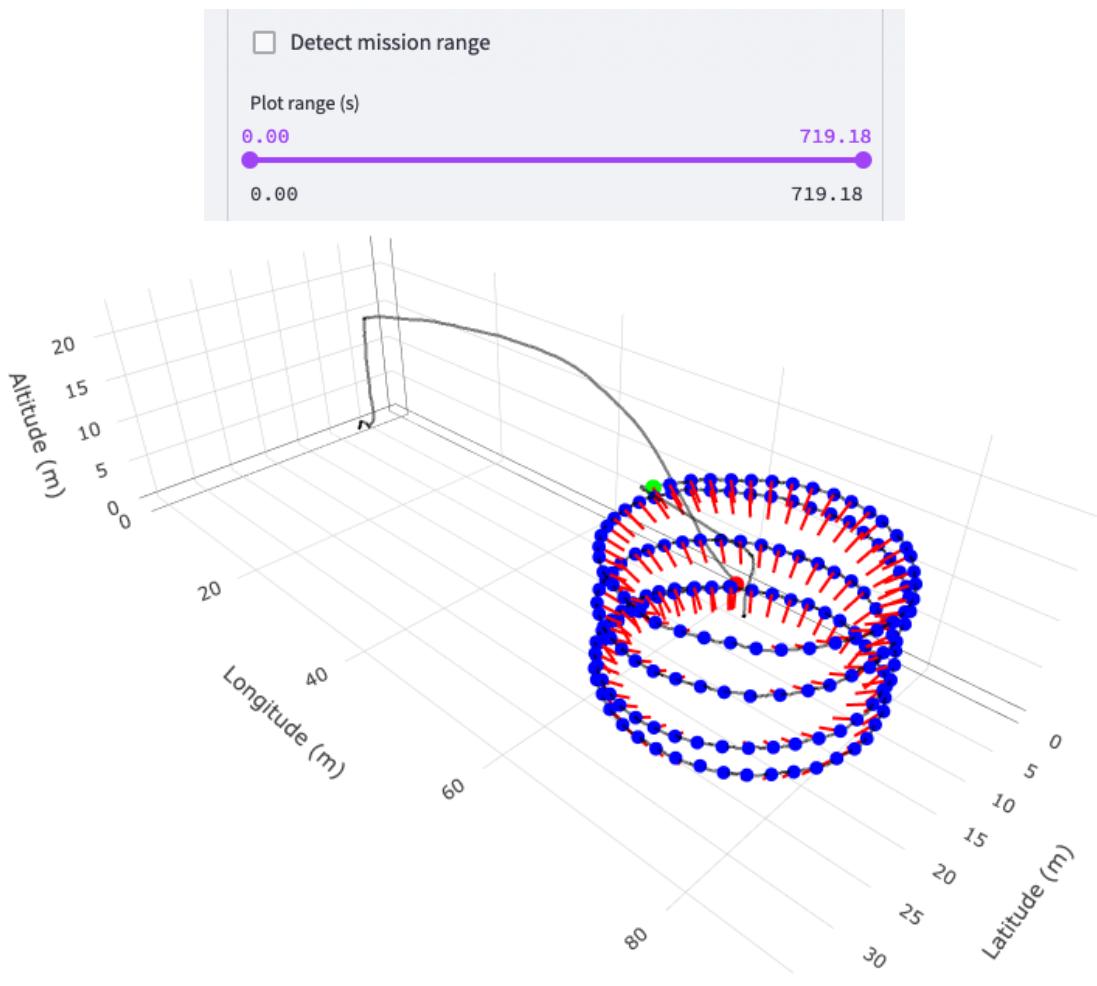


Figura 4.6.2.1: Visor 3D de un vuelo por logs sin detectar el rango de misión

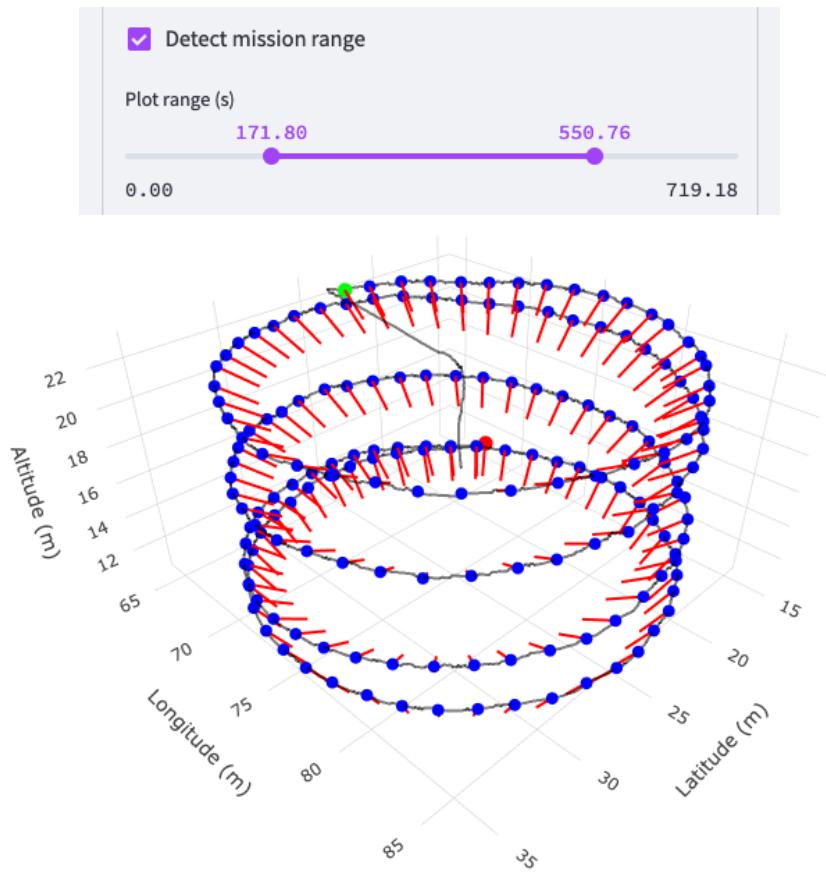


Figura 4.6.2.2: Visor 3D de un vuelo por logs detectando el rango de misión

Aprovechando que los logs contienen un evento de despegue, podemos usar la posición en ese momento para ubicar el vuelo respecto de la superficie terrestre y representarla en la visualización 3D.

Esta precisión no la teníamos con las imágenes, ya que conocíamos el dato de altitud relativa respecto del punto de despegue, pero no conocíamos ese punto de despegue al solo disponer de información del vuelo en el rango de tiempo en el que se capturaron imágenes.

Para ello, hemos creado una serie de funciones dentro de “mapbox_elevation.py” que usando la API de Mapbox, nos permiten obtener datos de elevación de la superficie terrestre. Por ello, para usar esta función, se necesita una conexión a internet.

En primer lugar, usamos la función “get_takeoff_altitude” para obtener la altitud en el momento del evento de despegue.

Después, llamamos a la función “get_elevation_grid”, que toma las coordenadas máximas y mínimas de latitud y longitud del vuelo, que utilizamos para generar un rectángulo que contenga al vuelo. En caso de que hayamos considerado oportuno añadir más espacio al rectángulo de superficie terrestre, se llama a la función “get_elevation_grid_offset” que nos permite añadir un borde al rectángulo, de tantas unidades o celdas como indiquemos. Después, dividimos ese rectángulo en una cuadrícula con un número de celdas que no supere a la resolución de los datos de elevación de Mapbox.

Mapbox proporciona los datos de elevación en forma de “tilesets” [41], que son imágenes en formato “png” con datos de elevación que cubren una zona rectangular de la superficie terrestre. Estos tilesets tienen una resolución máxima de 256x256 píxeles, y se pueden obtener en distintos niveles de zoom, siendo 15 el más alto disponible, el que mayor resolución espacial nos ofrece y el que usamos en la aplicación.

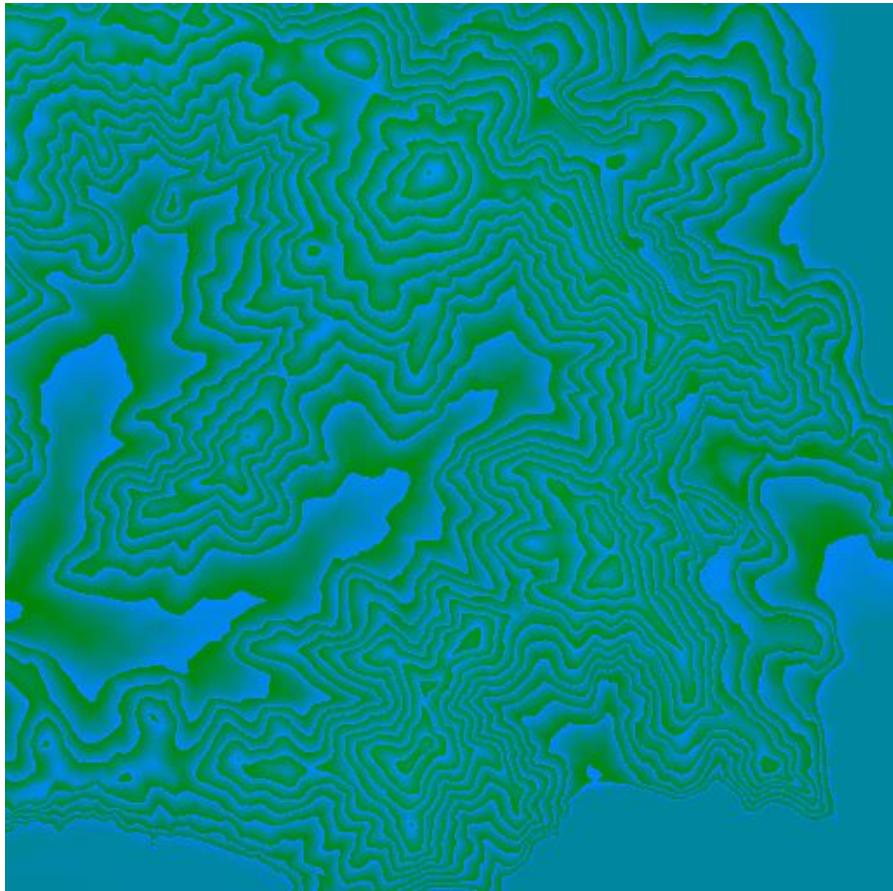


Figura 4.6.2.3: Tileset de elevación proporcionado por Mapbox [42]

Para obtener los datos de elevación de cada uno de los puntos de la cuadrícula, llamamos recursivamente a la función “get_elevation”, que hace lo siguiente:

En primer lugar, llamamos a la función “latlon_to_tile”, que toma unas coordenadas como argumento y utiliza la fórmula de conversión de coordenadas geográficas a coordenadas de tilesets de Mapbox para obtener las coordenadas del tileset que contiene el punto:

```
n = 2 ^ zoom  
xtile = n * ((lon + 180) / 360)  
lat_rad = radians(lat)  
ytile = n * (1 - (log(tan(lat_rad)) + (1 / cos(lat_rad))) / pi)) / 2
```

Después, llamamos a la función “tile_to_pixel”, que convierte las coordenadas de tileset a coordenadas de píxeles dentro del tileset, de la siguiente manera:

```
pixel_x = int((tile_x % 1) * 256)  
pixel_y = int((tile_y % 1) * 256)
```

Ahora que conocemos el píxel y el tileset exactos que contienen nuestro punto de interés, hacemos la petición a la API de Mapbox usando la librería “requests” de Python con la siguiente url:

```
url = f"https://api.mapbox.com/v4/mapbox.terrain-  
rgb/{zoom}/{tile_x}/{tile_y}.png?access_token={access_token}"
```

Finalmente, abrimos la imagen resultante con la librería “Pillow” como bytes, y la convertimos a un array de Numpy para poder acceder al valor del pixel que nos interesa y decodificarlo usando la función “decode_elevation”, que usa los valores RGB del pixel para obtener el valor de elevación en metros siguiendo la fórmula proporcionada por Mapbox:

```
elevation = -10000 + ((r * 256 * 256 + g * 256 + b) * 0.1)
```

La función “get_elevation” implementa además un sistema de caché que guarda los datos de elevación de cada tileset y en caso de detectarse que una petición va a resultar en la descarga de un tileset que ya se ha descargado previamente, se devuelve el valor de elevación de la caché en lugar de hacer la petición a la API de Mapbox.

Una vez tenemos los datos de elevación de todos los puntos de la cuadrícula, los convertimos en un array de Numpy, los normalizamos para que pasen a usar el mismo sistema de unidades local que usamos en el visor 3D interactivo, y los mostramos aprovechando el soporte de Plotly para las superficies 3D con una gama de colores que permita distinguir las diferencias en elevación del terreno.

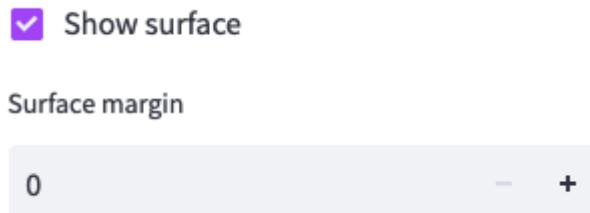


Figura 4.6.2.4: Opción para mostrar la superficie terrestre bajo el vuelo, junto a la entrada numérica para indicar el margen que queremos añadir

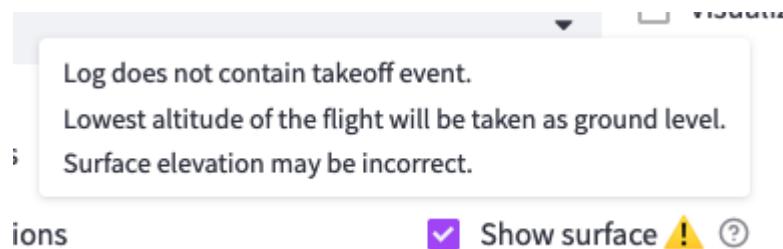


Figura 4.6.2.5: En caso de no encontrarse un evento de despegue, se mostrará un aviso de la posible falta de precisión de la visualización

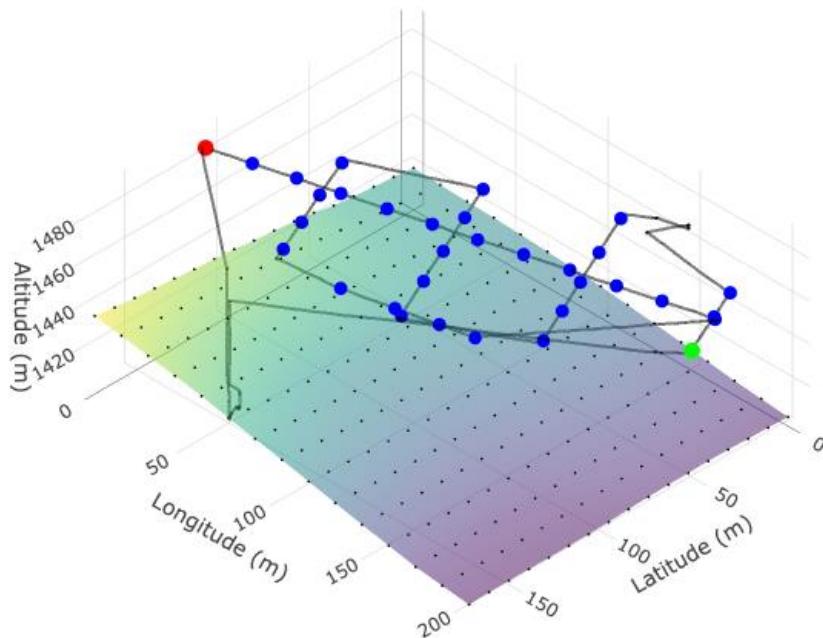


Figura 4.6.2.6: Visor 3D con superficie del terreno y un margen de 0 celdas, que cubre exactamente la extensión del vuelo

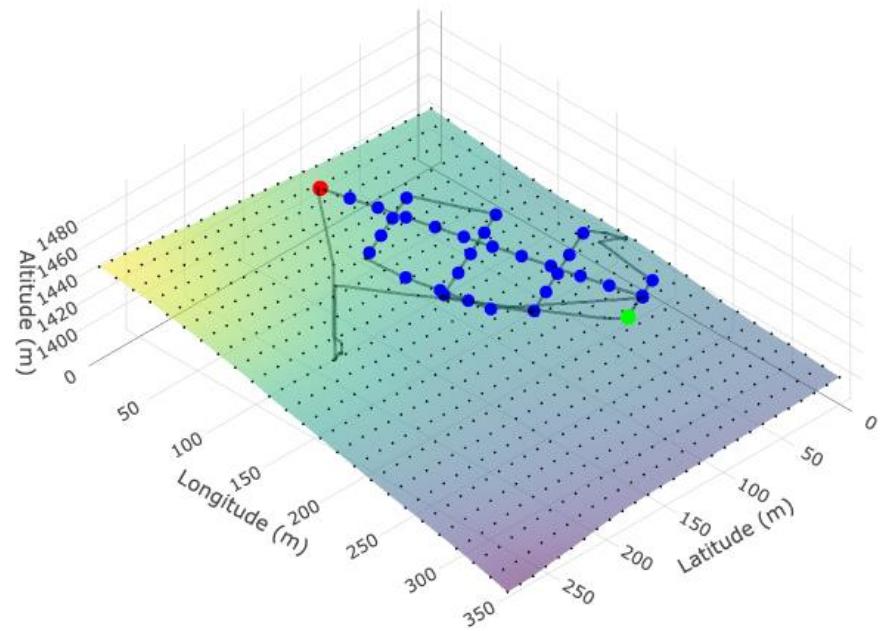


Figura 4.6.2.7: Visor 3D con superficie del terreno y un margen de 5 celdas, el valor por defecto

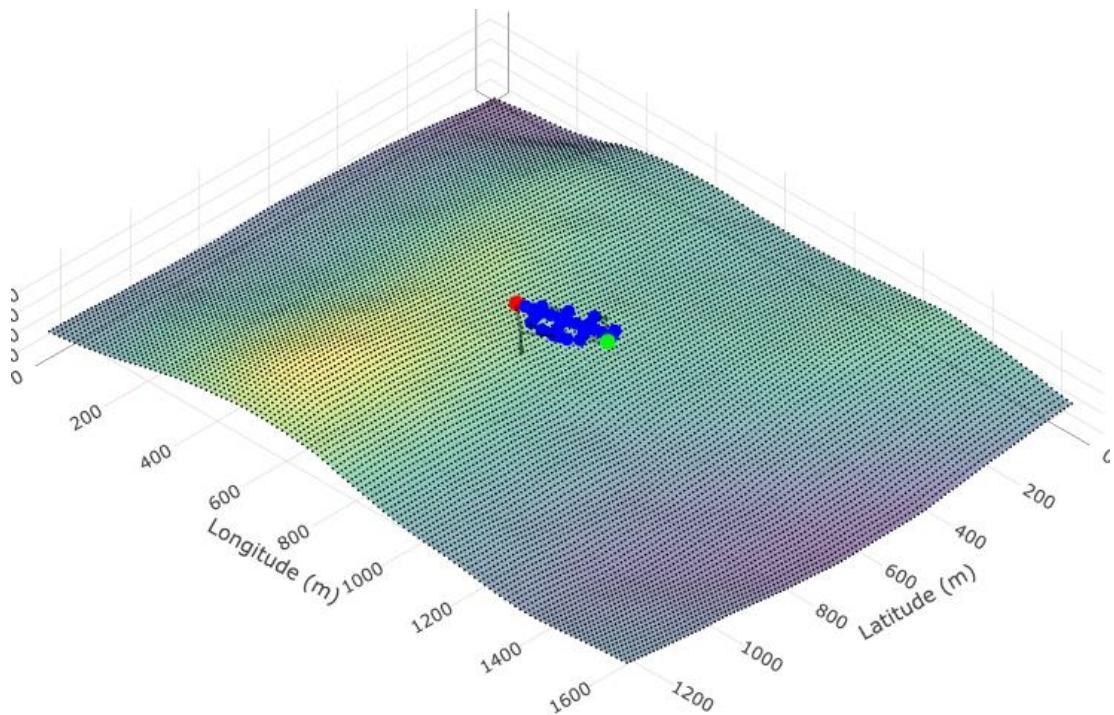


Figura 4.6.2.8: Visor 3D con superficie del terreno y un margen de 50 celdas

Una vez tenemos los datos de elevación, podemos aprovechar que también tenemos las direcciones de la cámara durante los eventos de captura de imágenes para calcular la proyección de las fotografías sobre la superficie terrestre, lo cual puede ser muy útil para comprobar el solapamiento de las imágenes, esencial en tareas de fotogrametría, cartografía e inspecciones agrícolas, entre otras.

Para ello, hemos creado una serie de funciones dentro de “mapbox_projection.py” que nos permiten calcular la proyección de las imágenes sobre la superficie terrestre.

En primer lugar, tenemos la función “find_elevation_grid_intersection”, que toma como argumentos unas coordenadas de posición, una dirección y los datos de elevación de la superficie, y calcula el punto de intersección entre la dirección y la superficie usando el algoritmo de intersección de Möller-Trumbore [43].

Este algoritmo permite calcular la intersección entre una dirección o rayo y un triángulo sin requerir de la precomputación de la ecuación del plano que contiene el triángulo, lo que hace muy eficiente computacionalmente e ideal para nuestro propósito, ya que vamos a calcular bastantes posibles intersecciones.

Su implementación ha sido muy sencilla, ya que el artículo original incluye una implementación en C que hemos traducido a Python.

La función “find_corner_intersections” se encarga de, en base a la posición del dron y la orientación y características de la cámara, calcular las cuatro direcciones que forman los límites del campo de visión de la cámara, y calcular la intersección de cada una de ellas con la superficie terrestre usando “find_elevation_grid_intersection”:

Calculamos los ángulos de campo de visión (FOV) de la cámara:

```
fov_width = 2 * arctan(sensor_width / (2 * focal_length))
fov_height = 2 * arctan((sensor_width / image_width) *
image_height / (2 * focal_length))
```

Calculamos los offsets de yaw y pitch para los puntos de las esquinas:

```
yaw_offsets = [-fov_width / 2, fov_width / 2, fov_width / 2, -
fov_width / 2]
pitch_offsets = [fov_height / 2, fov_height / 2, -fov_height / 2,
-fov_height / 2]
```

Calculamos las direcciones de las esquinas:

```
intersections = []
for yaw_offset, pitch_offset in zip(yaw_offsets, pitch_offsets):
    corner_yaw = yaw + deg(yaw_offset)
    corner_pitch = pitch + deg(pitch_offset)

    intersection = find_elevation_grid_intersection(point_x,
point_y, point_z, corner_yaw, corner_pitch, lat_grid,
lon_grid, elevation_grid)
    intersections.append(intersection)
```

Una vez tenemos los cuatro puntos de intersección, los ordenamos en el orden en el que se deben conectar para formar el polígono que representa la proyección de la imagen sobre la superficie terrestre, usando la función “order_trapezoid_points”.

En caso de que alguno de los puntos de intersección no se haya podido calcular (por ejemplo, porque la dirección de la cámara apuntaba hacia arriba), se devuelve un array vacío y esa proyección no se tiene en cuenta.

Es finalmente la función “get_projection” la que se encarga de, apoyándose en las funciones anteriores, calcular la geometría de la proyección de cada imagen sobre la superficie terrestre.

Esto lo hace mediante el uso de la librería SciPy, concretamente de su implementación de la triangulación de Delaunay [44][45], que nos permite generar una malla de polígonos triangulares a partir de nuestra malla de polígonos rectangulares original.

Después, se usa la librería Trimesh para convertir la malla de polígonos en un objeto de tipo “Trimesh” y poder aplicar fácilmente operaciones de cortes de malla.

Ya que los logs no proporcionan información sobre la cámara usada para tomar las imágenes, el usuario tendrá que seleccionar manualmente la cámara que se usó para tomar las imágenes. Para ello, se muestra un menú desplegable con todos los drones que soporta la aplicación, que tienen asociados una cámara y sus características (sensor, resolución, etc.) dentro de la estructura de datos “camera_data” dentro del fichero “camera.py”.

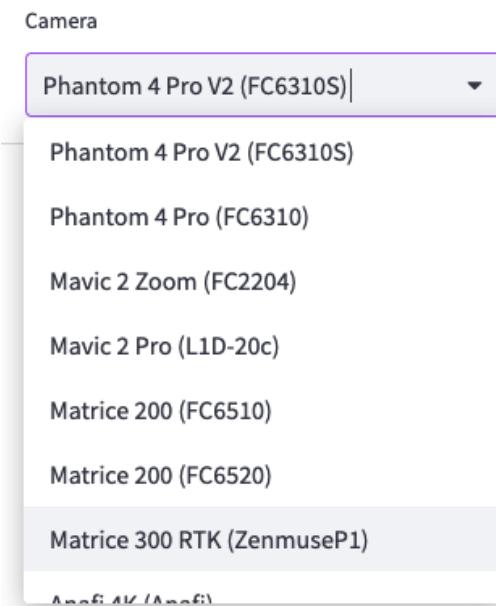


Figura 4.6.2.9: Menú de selección de dron para obtener los datos de la cámara usada para tomar las fotografías del vuelo

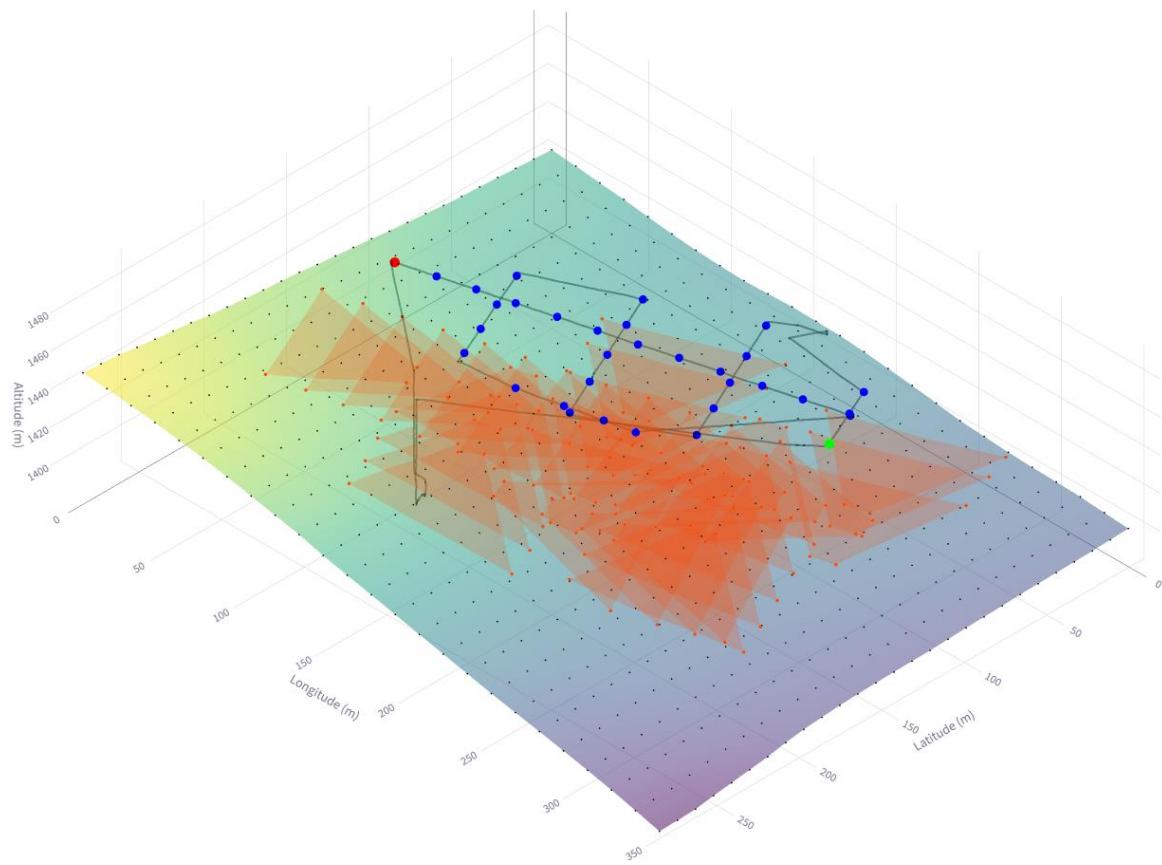


Figura 4.6.2.10: Visor 3D con superficie del terreno y la opción de visualización de proyecciones de las fotografías activadas.

Como podemos observar en la figura 4.6.2.10, se ha optado por usar un color salmón semi-transparente para representar las proyecciones de las fotografías, de manera que al producirse un solapamiento aumente la opacidad y se pueda apreciar visualmente en qué partes pueden hacer falta o sobran ciertas imágenes.

Respecto al resto de funcionalidades, el visor se comporta exactamente igual para los logs que para las imágenes.

4.6.3 Ejemplos de uso

Puesto que en los dos apartados anteriores el enfoque ha sido explicar las funcionalidades del visor, en esta sección mostraremos algunos ejemplos de uso de la herramienta, para demostrar la utilidad práctica de la misma.

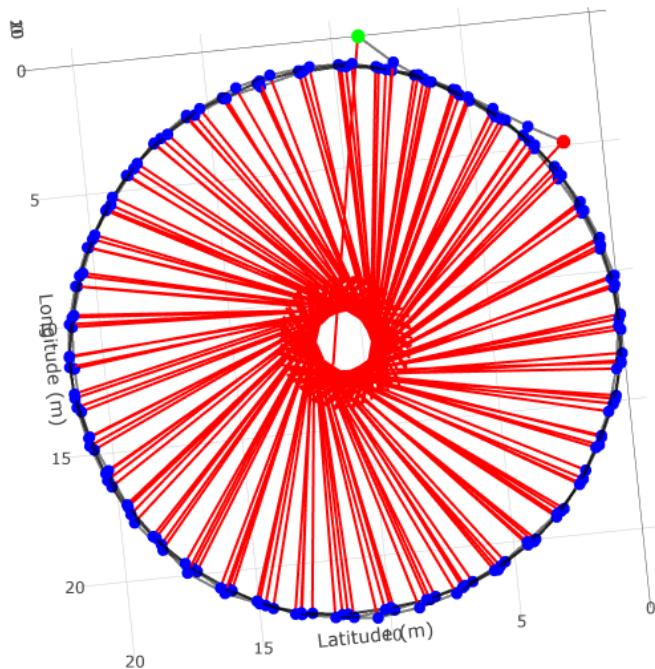


Figura 4.6.3.1: Ejemplo 1

El ejemplo de la figura 4.6.3.1 nos muestra un vuelo helicoidal visto desde arriba con una perspectiva ortogonal y con un valor alto de longitud de las direcciones.

Esta visualización nos permite comprobar que el círculo trazado por el vuelo es prácticamente perfecto, sin existir una pérdida de precisión de una órbita a la siguiente, a excepción del inicio y el final del vuelo, donde las imágenes primera y última se toman fuera del círculo.

Por su parte, podemos comprobar como las direcciones no apuntan al centro de la torre o estructura que se esté inspeccionando, sino que apuntan unos pocos grados hacia la izquierda. Este comportamiento puede ser intencional o no, pero el visor deja claro que sucede.

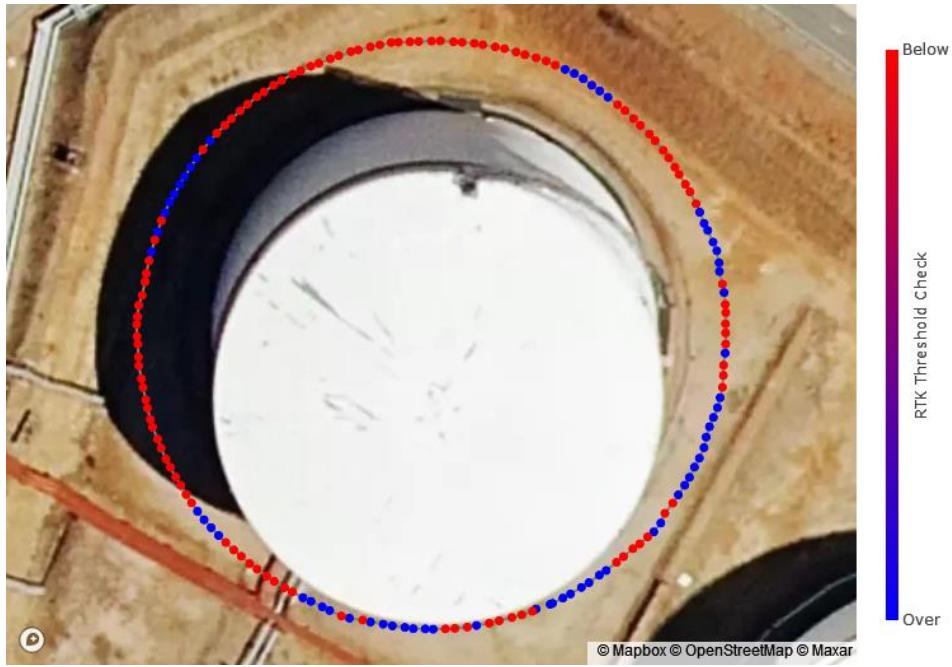


Figura 4.6.3.2: Ejemplo 2

En la figura 4.6.3.2 podemos ver un vuelo orbital alrededor de un tanque de almacenamiento de combustible, cuyos puntos de imagen hemos coloreado en base a la comprobación del umbral del RTK con la gama de colores “Bluered”. Se ha utilizado el visor de mapa 2D con la opción de datos satelitales de Mapbox.

Al tener los ajustes del umbral del RTK por defecto, sabemos que su valor está definido en 5 cm. Fijándonos en la leyenda en el lado derecho de la visualización, podemos ver que los puntos azules corresponden a imágenes que están por encima de ese umbral, mientras que los puntos rojos corresponden a imágenes que están por debajo y por tanto tienen una muy buena precisión de 5 cm o inferior.

Podemos observar como la mayoría de las imágenes con peor precisión han sido tomadas en la zona sureste del tanque, por lo que puede estar produciéndose algún tipo de interferencia con el sistema de posicionamiento RTK en esa zona.

Finalmente, y aunque la imagen satelital no es perfectamente cenital, podemos usar esta visualización para asegurar que la órbita se ha realizado alrededor del centro del tanque y no algo más a un lado o a otro.

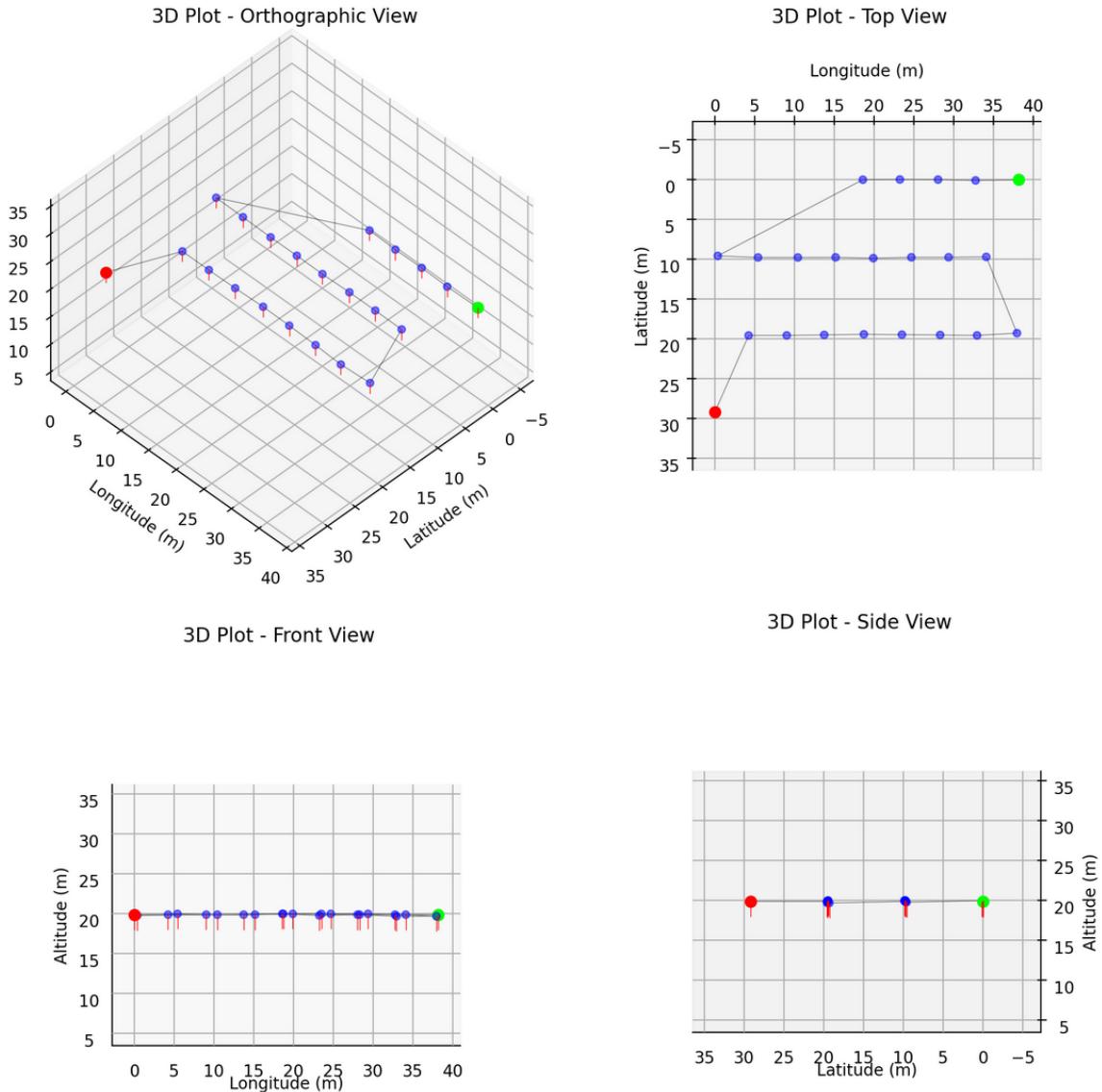


Figura 4.6.3.3: Ejemplo 3

En el ejemplo de la figura 4.6.3.3 podemos apreciar una misión de tipo grid para la que se ha decidido utilizar el visor 3D estático.

Los dos gráficos superiores nos permiten comprobar gracias a sus líneas de referencia que el vuelo se ha realizado paralelo a los ejes de coordenadas, y por lo tanto, haciendo líneas paralelas al eje de longitud terrestre, también conocido como meridiano.

También podemos comprobar muy fácilmente como las fotos del grid están separadas entre sí 5 metros, mientras que las líneas realizadas lo están 10 metros entre sí.

Finalmente, gracias a los gráficos inferiores podemos ver que, durante la misión, el grid se ha mantenido a una altura de 20 metros con una desviación mínima.

En caso de querer hacer un informe sobre el desempeño de este tipo de misiones, este gráfico da mucha información de forma concisa y visual.

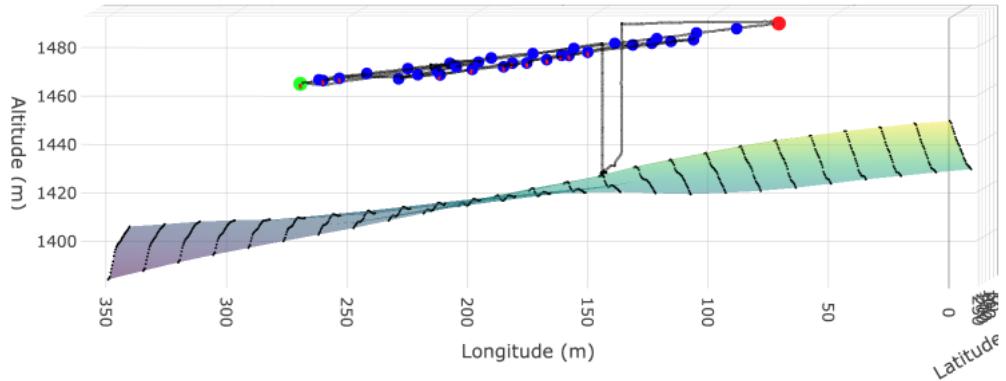


Figura 4.6.3.4: Ejemplo 4

En la figura 4.6.3.4, podemos ver lo que parece ser una misión de tipo grid desde el lateral, con la visualización de la superficie activada.

Se puede apreciar que no se trata de un grid al uso que mantiene al dron a la misma altura, sino que está haciendo uso de una funcionalidad de seguimiento del terreno que adapta la altura del dron en base a algún tipo de mapa topográfico.

Como podemos ver, el recorrido del dron comienza en el suelo y se dirige a la parte superior de la misión, tras lo cual saca una serie de fotografías que siguen el desnivel de la superficie correctamente.

Gracias a las medidas presentes en la visualización, podemos estimar que el dron ha descendido unos 25 metros entre la primera y la última foto, habiendo realizado un vuelo con una longitud de unos 200 metros antes de volver al punto inicial.

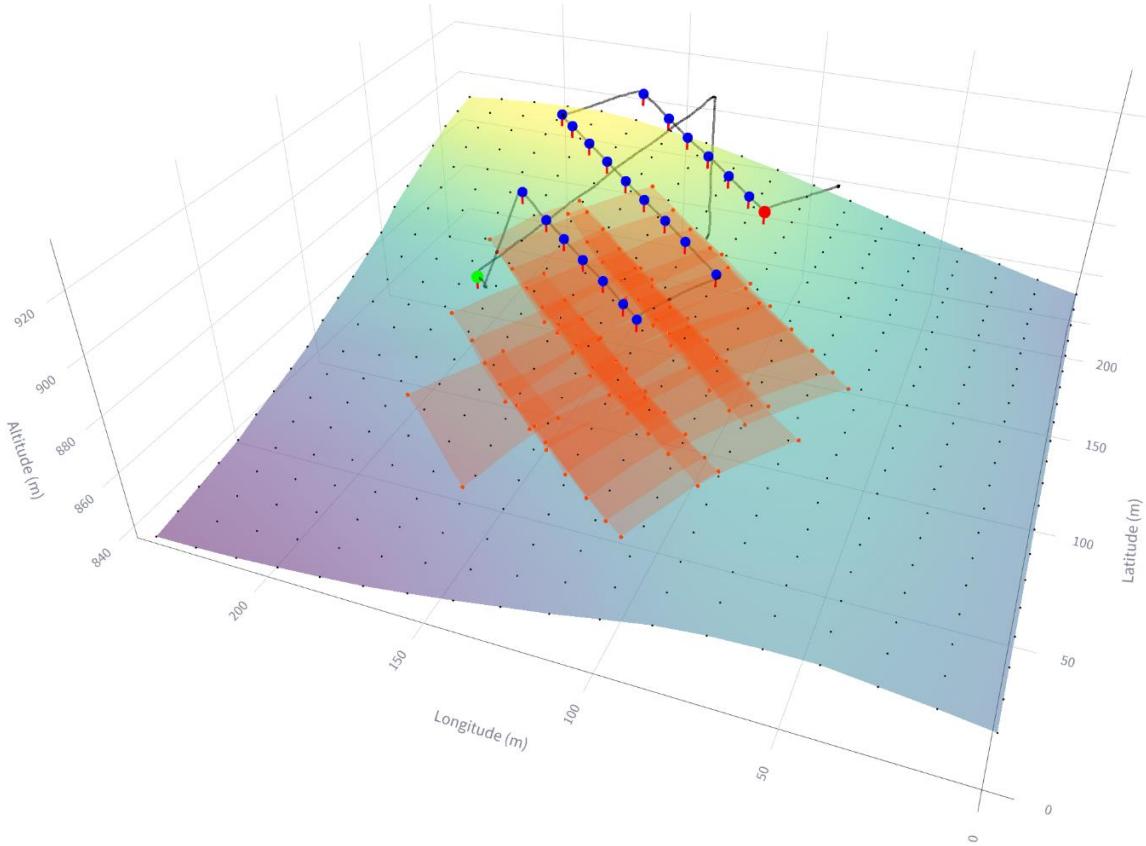


Figura 4.6.3.5: Ejemplo 5

Como último ejemplo, en la figura 4.6.3.5 podemos ver otra misión de tipo grid volada con seguimiento del terreno. Las opciones de visualización de la superficie y de las proyecciones están activadas. Además, conocemos que este vuelo se realizó con un dron de la marca Parrot y modelo ANAFI Ai, por lo que se ha seleccionado para que los cálculos de proyección sean los correspondientes a las características de la cámara de ese dron.

Como se puede observar, la cámara mira directamente hacia abajo en todas las fotografías y como consecuencia, las proyecciones son rectangulares con pequeñas deformaciones debido al relieve irregular.

Si nos fijamos en el solapamiento de las proyecciones, podemos ver que este es mayor en las proyecciones correspondientes a las imágenes pertenecientes al interior del vuelo, mientras que aquellas que se sitúan en el exterior tienen un menor solapamiento.

Por su parte, parece que hay bastante solapamiento entre las múltiples filas del vuelo, lo cual es una buena señal si los datos se van a usar para hacer algún tipo de reconstrucción por fotogrametría, ya que se necesitan datos solapados desde múltiples ángulos para generar un buen modelo.

De repetirse este vuelo, el piloto podría considerar hacer la superficie a fotografiar cuadrada o rectangular para mejorar el solapamiento en los bordes.

4.7 Gráficos

SkyStats ofrece la posibilidad de generar gráficos 2D tanto predefinidos como personalizados que permiten al usuario visualizar la progresión de los datos respecto del tiempo.

La implementación de los gráficos 2D se ha realizado haciendo uso de la librería Matplotlib, que ofrece muchas opciones de personalización y una gran versatilidad. Además, genera los gráficos como imágenes, por lo que se pueden copiar directamente desde la aplicación web para ser usados en informes.

Ya que las imágenes y los logs tienen sus propias particularidades y no comparten algunos datos, aunque sí la mayoría, se ha separado la implementación de los gráficos 2D en tres ficheros distintos:

- “shared_plots.py”: Contiene las funciones que generan los gráficos 2D que comparten imágenes y logs. Contiene la mayoría de las funciones de generación de gráficos 2D.
- “image_plots.py”: Contiene las funciones que generan los gráficos 2D específicos de las imágenes. Incluye la implementación de la función “custom_plot”, que permite al usuario generar gráficos 2D personalizados.
- “log_plots.py”: Contiene las funciones que generan los gráficos 2D específicos de los logs. También incluye la implementación de la función “custom_plot”, que permite al usuario generar gráficos 2D personalizados, pero modificada para funcionar con logs.

Debido a las grandes similitudes entre los gráficos de cada tipo de datos, la interfaz para ambos es muy parecida y consiste en ambos casos en una lista de pestañas que contienen los gráficos disponibles, junto a un menú desplegable que permite al usuario seleccionar el color de los gráficos.

Los ajustes del menú de selección de vuelo en la barra lateral vistos en las secciones 4.4.1 y 4.4.2 del documento, como el rango de tiempo del análisis o el umbral de precisión del RTK también afectan a los gráficos.

A continuación, mostraremos primero los gráficos de las imágenes y después los de los logs, haciendo hincapié en las diferencias entre ambos.

4.7.1 Imágenes

1. Gráficos de posición GPS:

Graphs

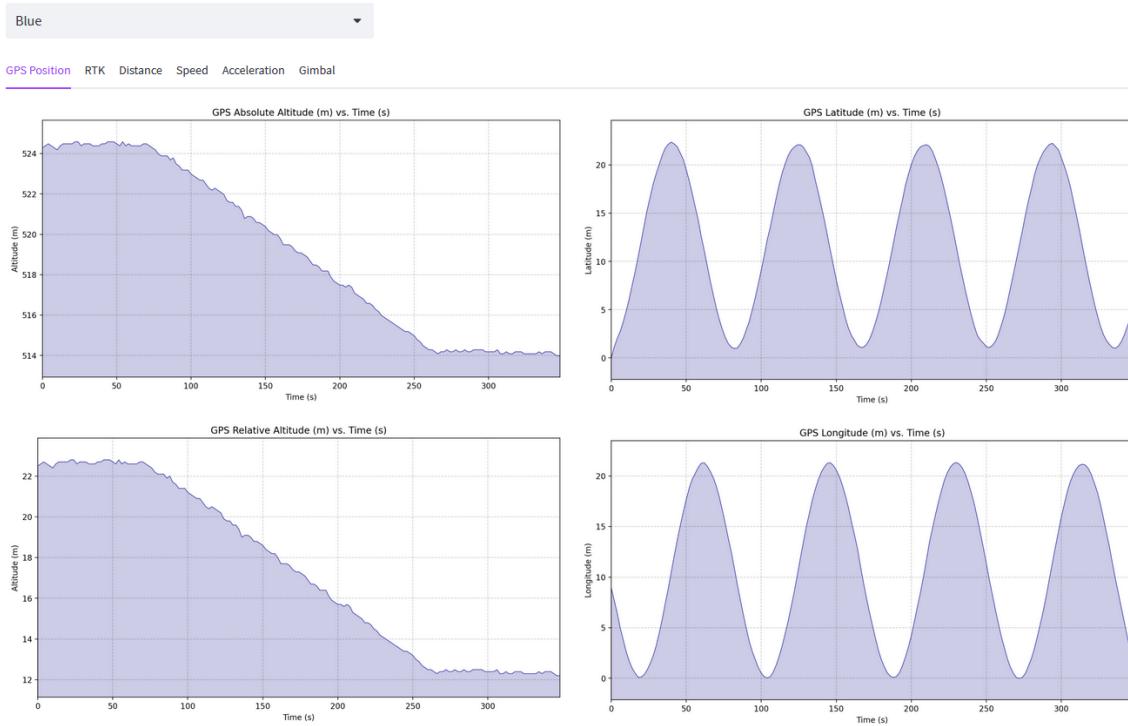


Figura 4.7.1.1: De izquierda a derecha y de arriba a abajo: Gráficos de la evolución de la posición GPS de altitud absoluta, latitud, altitud relativa y longitud respecto del tiempo

2. Gráficos de precisión RTK:



Figura 4.7.1.2: De izquierda a derecha y de arriba a abajo: Gráficos de la evolución de la desviación estándar total, de latitud, altitud, longitud, edad diferencial y estado o flag de precisión del RTK (cm, dm, m, sin posicionamiento) respecto del tiempo. Los gráficos cambian de color en base al umbral de RTK seleccionado.

3. Gráficos de distancia:



Figura 4.7.1.3: De izquierda a derecha y de arriba a abajo: Gráficos de la evolución de la distancia total, de latitud, altitud y longitud respecto del tiempo. En verde.

4. Gráficos de velocidad:

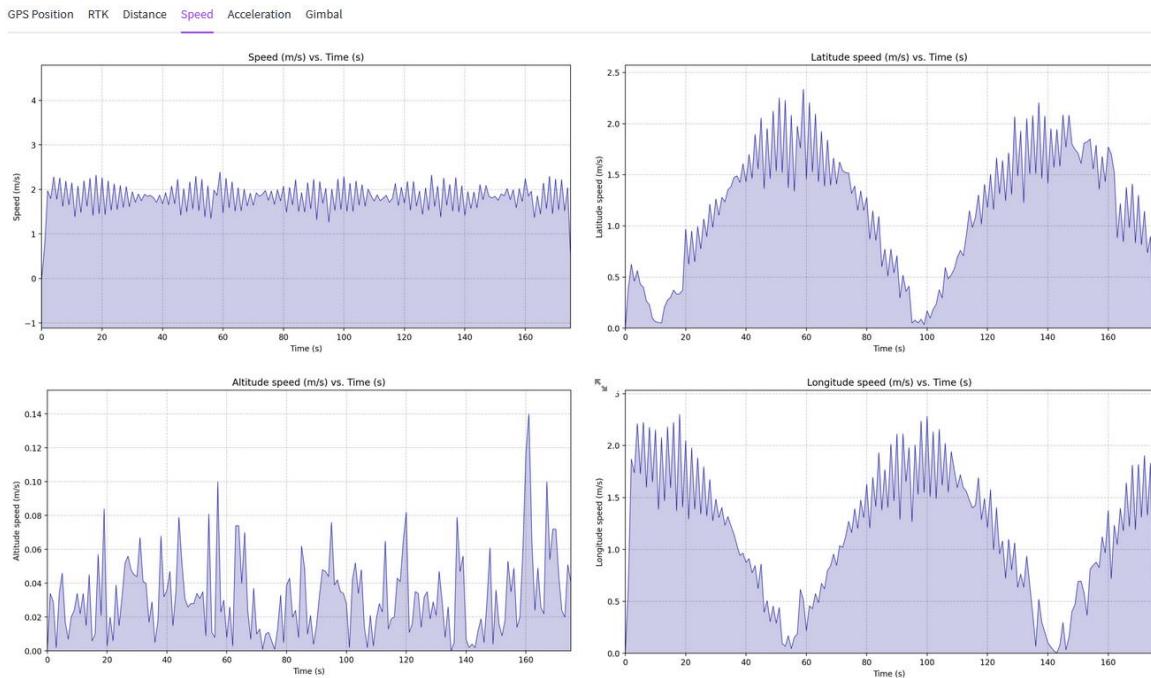


Figura 4.7.1.4: De izquierda a derecha y de arriba a abajo: Gráficos de la evolución de la velocidad total, de latitud, altitud y longitud respecto del tiempo.

5. Gráficos de aceleración:

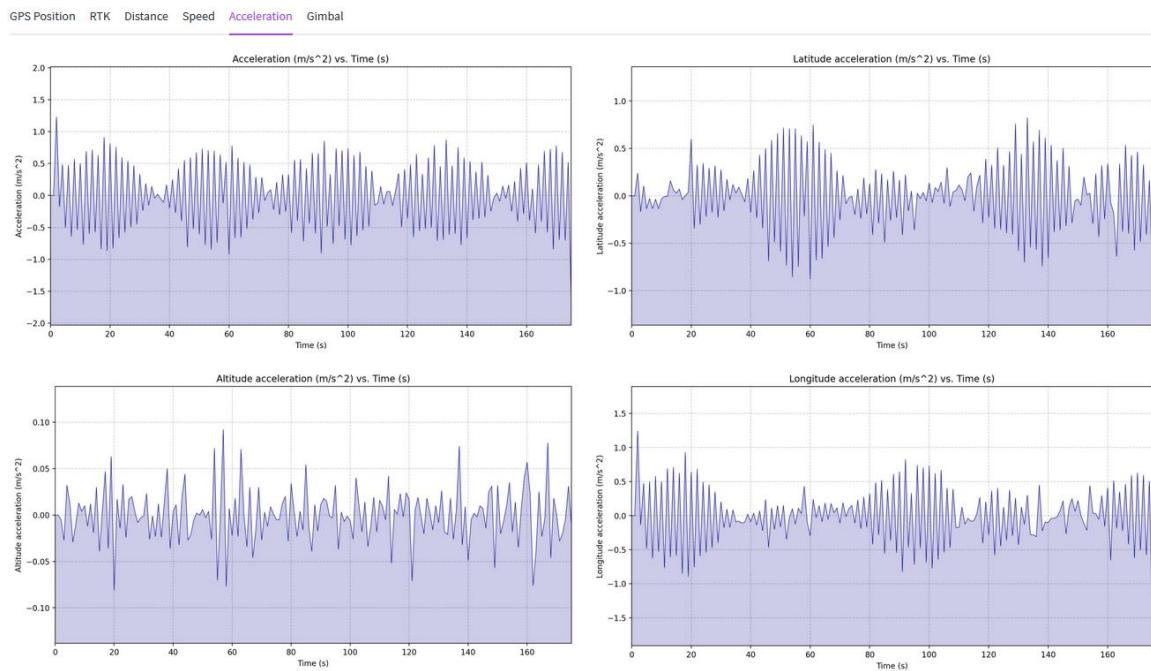


Figura 4.7.1.5: De izquierda a derecha y de arriba a abajo: Gráficos de la evolución de la aceleración total, de latitud, altitud y longitud respecto del tiempo.

6. Gráficos de orientación o gimbal:

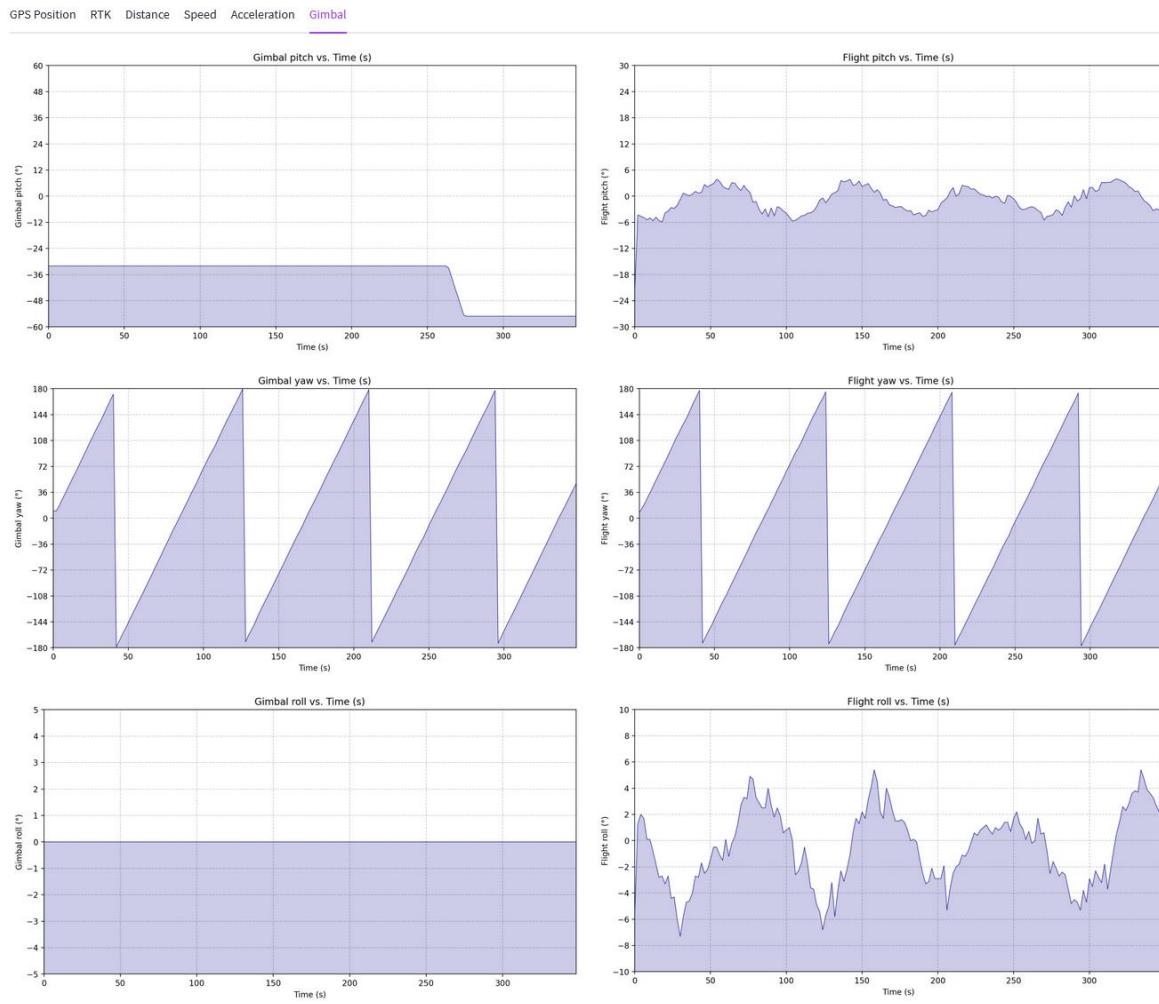


Figura 4.7.1.6: De izquierda a derecha y de arriba a abajo: Gráficos de la evolución de la orientación de la cámara (gimbal) y del dron (flight), concretamente de su pitch, yaw y roll, respecto del tiempo

4.7.2 Logs

Los logs comparten los gráficos de posición GPS, distancia, velocidad, aceleración y gimbal con las imágenes. Ya que no tienen información sobre el sistema de posicionamiento RTK, no tienen la opción de mostrar esos gráficos.

A continuación, mostraremos los gráficos específicos de los logs.

1. Gráficos de posición GPS:

Usamos la posición de despegue del dron para añadir un gráfico de posicionamiento que nos indica la altitud respecto del relieve del terreno

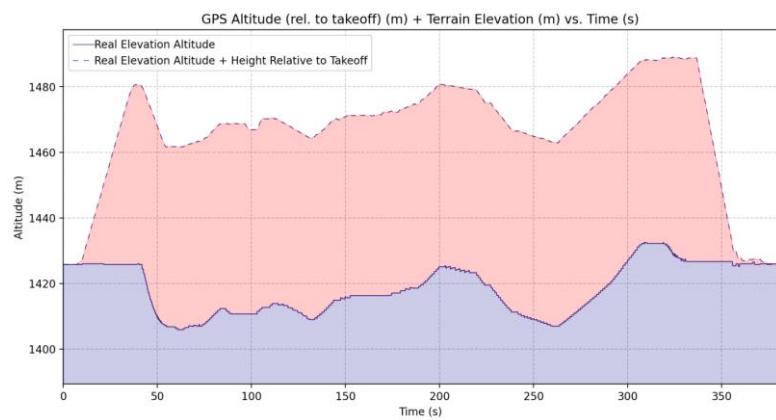


Figura 4.7.2.1: Gráfico de altitud respecto del relieve del terreno respecto del tiempo

2. Gráficos de batería:

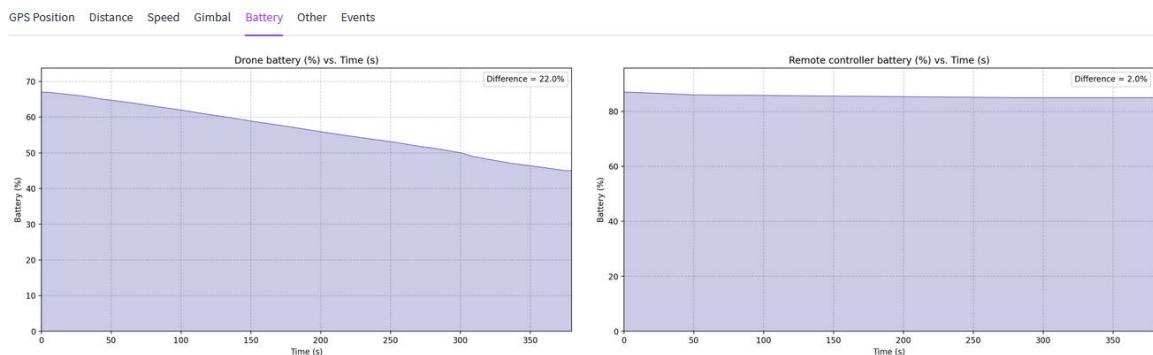


Figura 4.7.2.2: Gráfico del porcentaje de batería del dron y del mando respecto del tiempo

3. Gráficos de eventos:

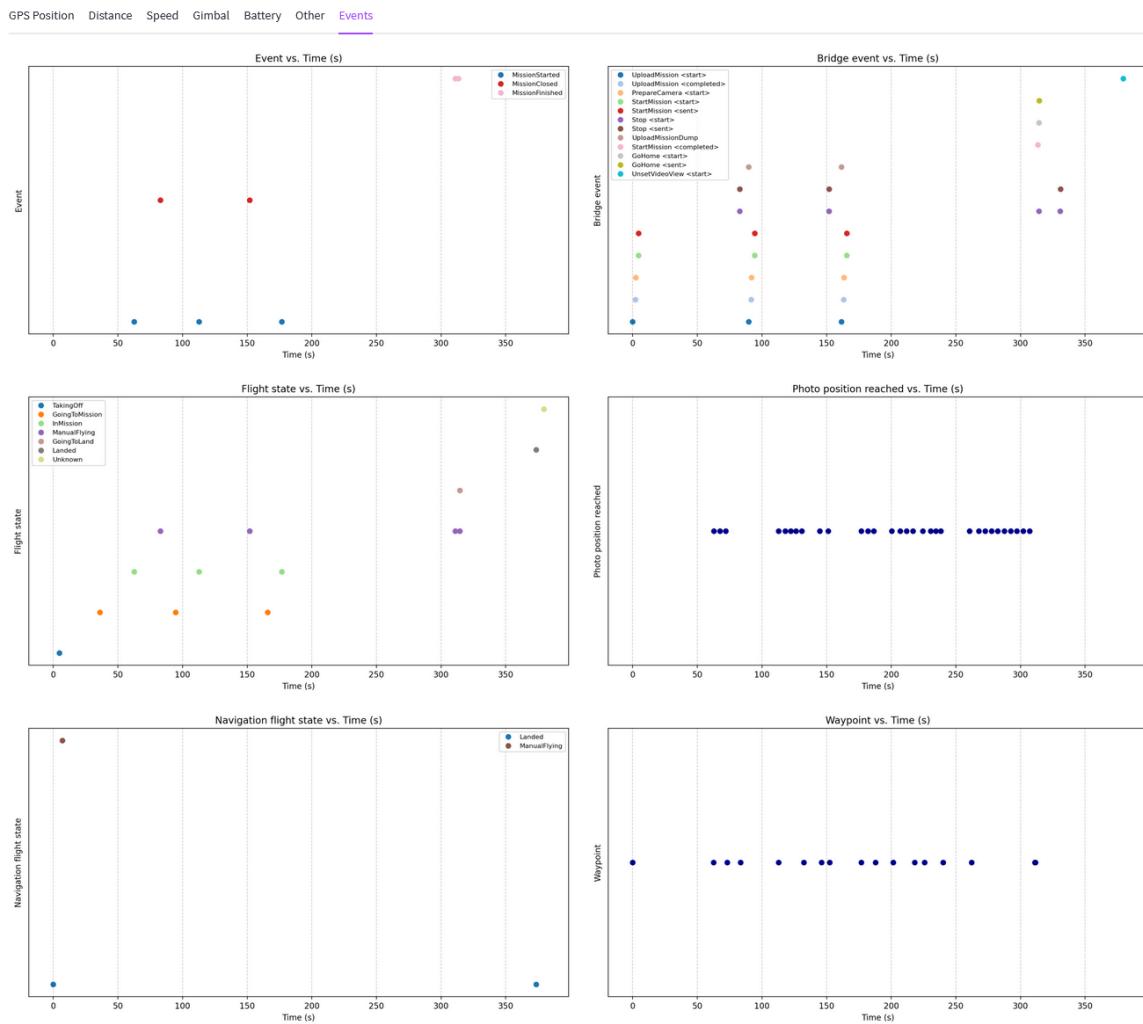


Figura 4.7.2.3: De izquierda a derecha y de arriba a abajo: Gráficos de eventos de los siguientes tipos respecto del tiempo: inicio y final de misión, eventos de comunicación con la aplicación (bridge events), estado del vuelo, eventos de posición de foto, estado de navegación (aterrizado o volando) y eventos de puntos de ruta (waypoints)

4. Otros gráficos:

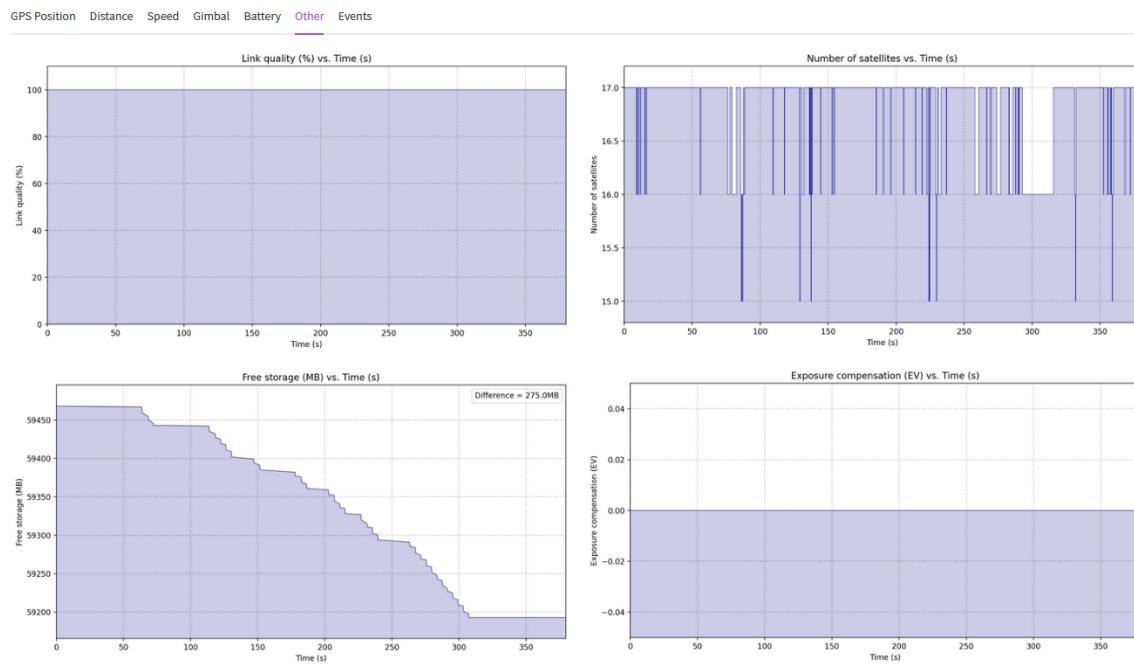


Figura 4.7.2.4: De izquierda a derecha y de arriba a abajo: Gráficos de la calidad de la conexión con el mando, número de satélites a los que tiene acceso el dron, del almacenamiento libre y de la compensación de exposición

4.7.3 Personalizados

Además de los gráficos predefinidos, SkyStats ofrece la posibilidad de generar gráficos personalizados, que permiten al usuario seleccionar una o varias variables de las ya empleadas en los gráficos predefinidos, mediante un menú desplegable de selección múltiple, y generar un gráfico con ellas.

Estos gráficos añaden nuevas escalas en el eje Y para cada variable seleccionada. Gracias a esto, podemos comparar la evolución de varias variables en un mismo gráfico, lo cual puede resultar muy útil para analizar la relación entre ellas.

Custom Graphs

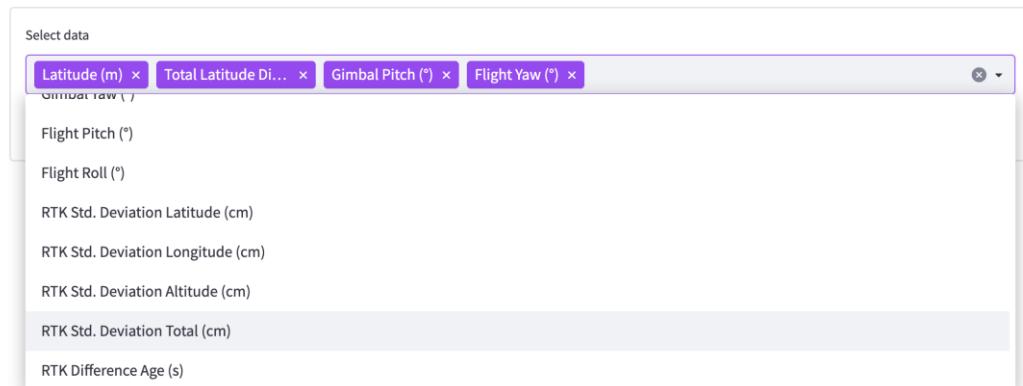


Figura 4.7.3.1: Menú de selección múltiple de variables

Custom Graphs

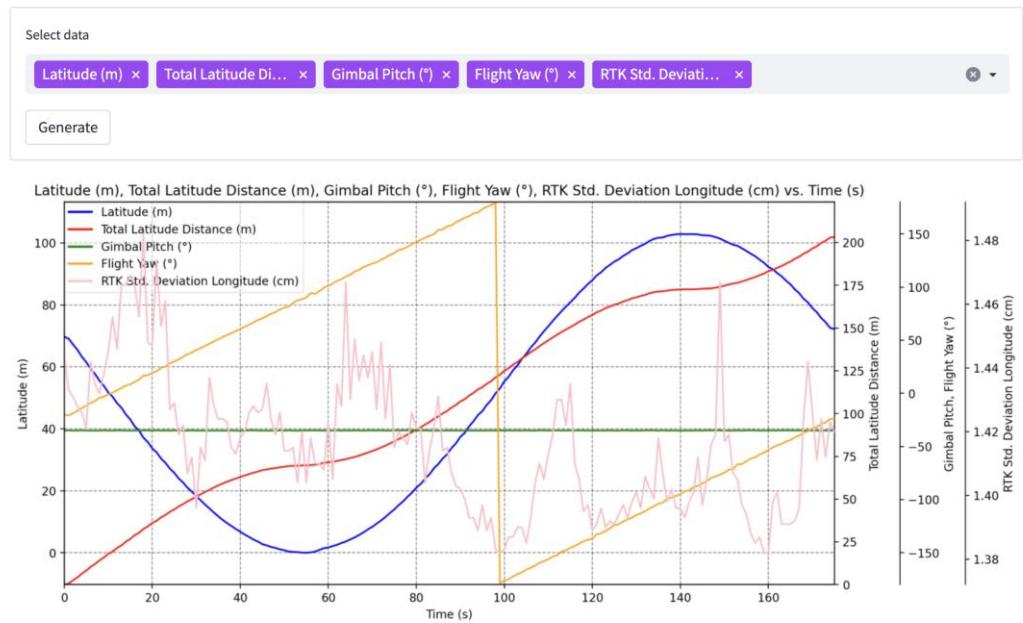


Figura 4.7.3.2: Gráfico personalizado de ejemplo 1

Custom Graphs

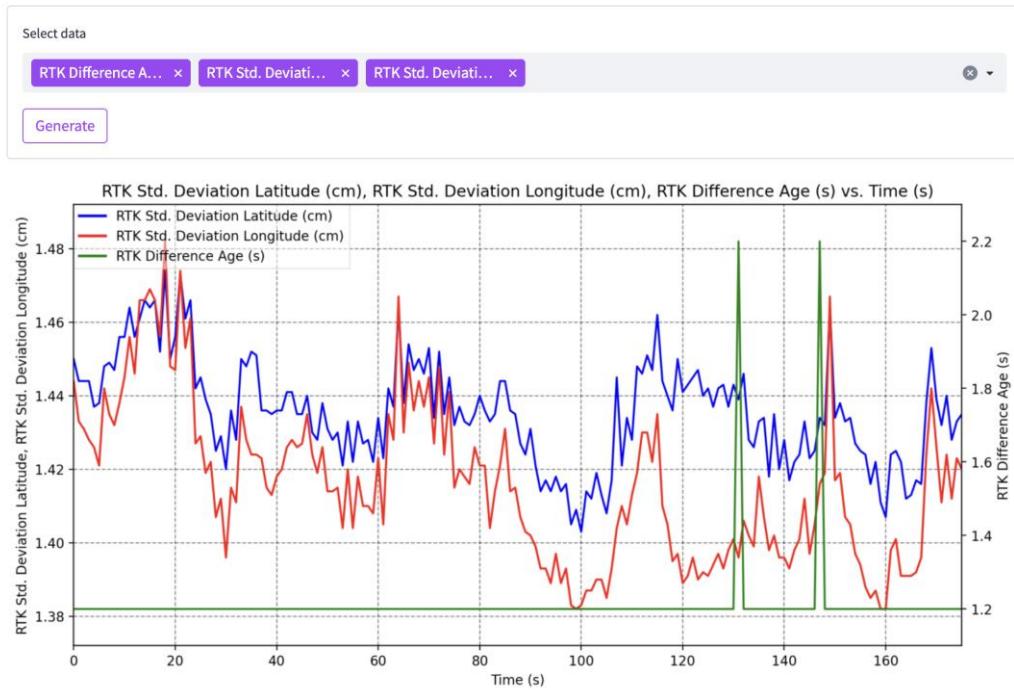


Figura 4.7.3.3: Gráfico personalizado de ejemplo 2

4.8 Análisis estadístico

SkyStats también ofrece la posibilidad de realizar un análisis estadístico de los datos, que permite al usuario obtener información sobre las variables del vuelo.

En el caso de los logs, podemos escoger entre usar los datos interpolados o sin interpolar para el análisis. En el caso de los datos interpolados, será más sencillo establecer relaciones entre las variables, pero en el caso de los datos sin interpolar, los datos serán más precisos y el análisis se realizará en menos tiempo.

4.8.1 Básico

Como tenemos los datos almacenados en un objeto dataframe de la librería Pandas, podemos usar la función “describe” para obtener un análisis estadístico básico de las variables, que nos proporciona la siguiente información:

- count: Número de valores no nulos de la variable.
- mean: Media aritmética de los valores de la variable.
- std: Desviación estándar de los valores de la variable.
- min: Valor mínimo de la variable.
- 25%: Primer cuartil de los valores de la variable.
- 50%: Mediana de los valores de la variable.
- 75%: Tercer cuartil de los valores de la variable.
- max: Valor máximo de la variable.

Statistics

Data 

Raw 

Basic

	ll (°)	Flight Yaw (°)	Gimbal Pitch (°)	Gimbal Roll (°)	↓ Gimbal Yaw (°)	Velocity Latitude (m/s)	Velocity Longitude (m/s)	Velocity Altitude (m/s)	Velocity (m/s)	Drone Battery (%)
count	730	3,730	2,820	2,820	2,820	814	814	814	814	23
max	17.6	179.9	0	0.5	179.9	0.8	2.6	3.1	7.6922	95
std	.051	99.2421	10.1468	0.0316	100.5957	1.7832	0.792	0.6255	1.7642	6.7823
75%	1.9	79.075	-32	0	84.95	0.2	0.4	0	0.6325	89.5
50%	0.1	19.15	-32	0	14.65	-0.1	0	0	0.5477	84
mean	.776	4.3122	-37.9498	0.0032	4.6707	-0.6217	-0.0483	0.0312	1.2143	84
25%	.575	-79.475	-50.95	0	-80.2	-0.5	-0.4	0	0.5	78.5
min	27.4	-179.7	-55.4	-0.1	-179.6	-7.5	-3.5	-2.3	0	73

Figura 4.8.1.1: Tabla de estadísticas básicas y selector de datos procesados o sin procesar

La tabla resultante es interactiva, por lo que podemos ordenarla en función de los valores de una variable haciendo clic en su nombre, y podemos seleccionar grupos de celdas haciendo clic y arrastrando el cursor, y copiar sus valores como si se tratase de una tabla de Excel.

Podemos usar esta información para hacernos una idea de la distribución de los datos de cada variable, y de esta manera, poder detectar valores atípicos o anomalías en los datos.

4.8.2 Avanzado

En caso de querer profundizar más en el análisis de los datos, podemos usar la librería “pandas-profiling” y su implementación para el framework Streamlit, “streamlit-pandas-profiling”, que nos permite generar un informe completo de análisis de los datos, conocido como análisis exploratorio de datos.

Dado que la generación de este informe puede ser computacionalmente intensiva si tenemos muchos datos, se da la opción de generarla a través de un botón, en lugar de generarla automáticamente al seleccionar la opción de análisis estadístico.

Advanced

Generate Profile Report (may take a while)

 Generating profile report...

Figura 4.8.2.1: Botón de generación del informe junto a su rueda de carga

El informe generado nos proporciona la siguiente información, que puede complementar al resto de métodos de análisis y visualización que proporciona SkyStats para obtener una visión completa del vuelo:

1. Resumen general: Ofrece una visión general de los datos analizados y un informe de alertas sobre comportamientos observados en los datos (valores constantes, alta correlación entre variables, distribuciones uniformes, etc.).

Overview

The screenshot shows the 'Overview' section of a data analysis tool. At the top, there are three tabs: 'Overview' (selected), 'Alerts' (with 52 notifications), and 'Reproduction'. Below the tabs, there are two main sections: 'Dataset statistics' and 'Variable types'.

Dataset statistics		Variable types	
Number of variables	32	Numeric	27
Number of observations	176	Categorical	5
Missing cells	0		
Missing cells (%)	0.0%		
Duplicate rows	0		
Duplicate rows (%)	0.0%		
Total size in memory	44.1 KiB		
Average record size in memory	256.7 B		

Figura 4.8.2.2: Resumen general de las variables

Overview

The screenshot shows the 'Alerts' section of the data analysis tool. At the top, there are three tabs: 'Overview' (selected), 'Alerts' (with 52 notifications), and 'Reproduction'. Below the tabs, there is a single section titled 'Alerts'.

Alert Description	Type
Gimbal Roll has constant value "0.0"	Constant
RTK Flag has constant value "50.0"	Constant
RTK Threshold Check has constant value "1.0"	Constant
Timestamp is highly overall correlated with Latitude and 6 other fields	High correlation
Latitude is highly overall correlated with Timestamp and 8 other fields	High correlation
Longitude is highly overall correlated with GPS Latitude and 1 other fields	High correlation
Altitude is highly overall correlated with Absolute Altitude	High correlation
Absolute Altitude is highly overall correlated with Altitude	High correlation
GPS Latitude is highly overall correlated with Longitude and 1 other fields	High correlation
Longitude Speed has unique values	Unique
Flight Yaw has unique values	Unique
Altitude Speed has 2 (1.1%) zeros	Zeros
Acceleration has 2 (1.1%) zeros	Zeros
Latitude Acceleration has 2 (1.1%) zeros	Zeros
Longitude Acceleration has 2 (1.1%) zeros	Zeros
Altitude Acceleration has 5 (2.8%) zeros	Zeros
Flight Roll has 2 (1.1%) zeros	Zeros

Figura 4.8.2.3: Extracto del análisis de comportamientos observados en las variables

2. Variables: Permite al usuario hacer un análisis estadístico avanzado de las variables individuales, que se pueden seleccionar desde un menú desplegable.

Variables

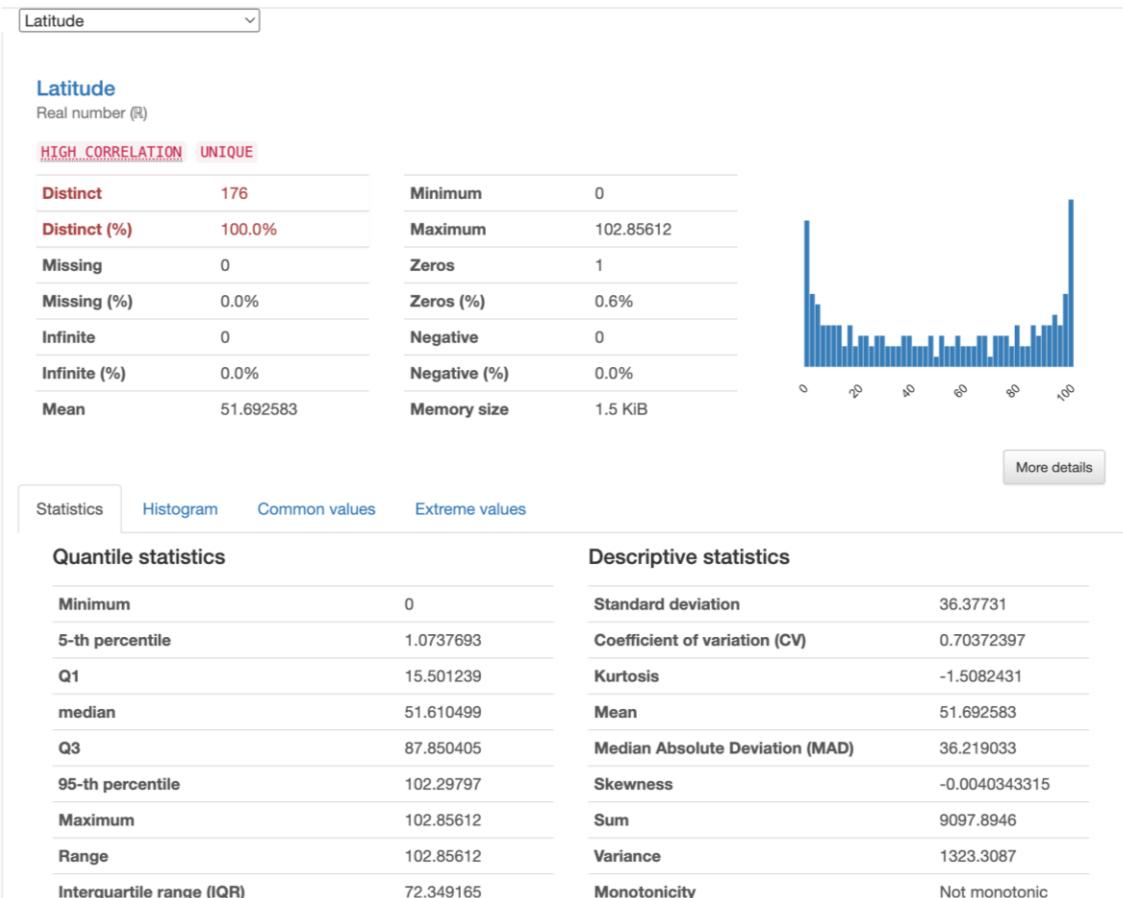


Figura 4.8.2.4: Análisis estadístico de la variable latitud

3. Interacciones: Permite al usuario comparar y visualizar las interacciones entre dos variables, lo cual puede ser útil para observar patrones y relaciones entre ellas.

Interactions

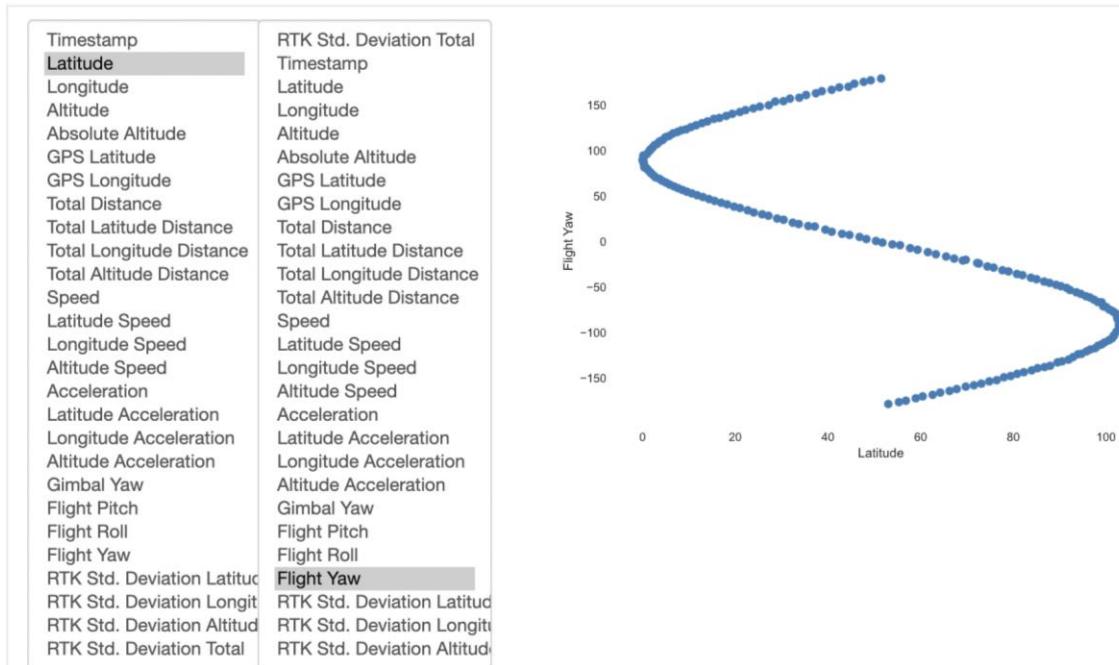


Figura 4.8.2.5: Análisis visual de interacción entre las variables latitud y yaw del vuelo

4. Correlaciones: Permite al usuario visualizar las correlaciones entre todas las variables haciendo uso de un mapa de calor o “heatmap”.

Correlations

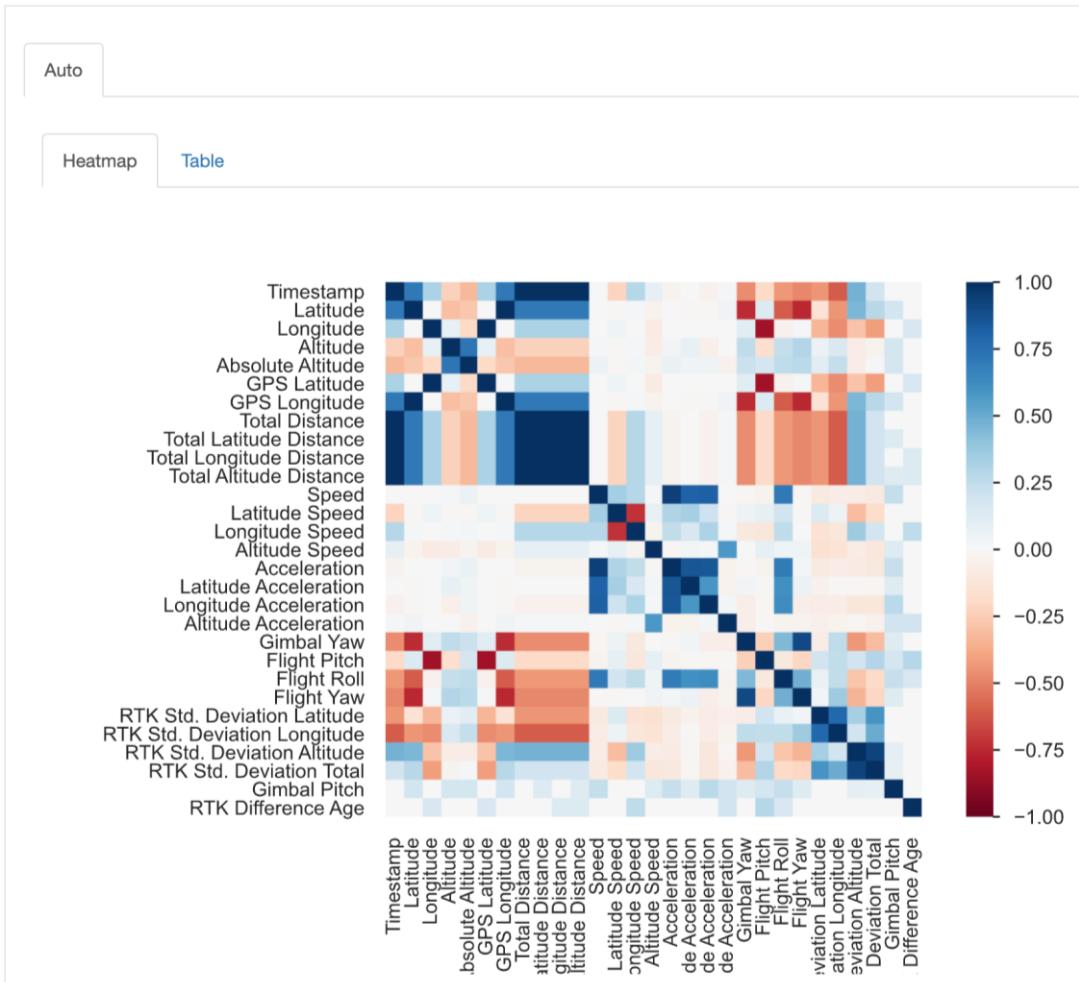


Figura 4.8.2.6: “Heatmap” de correlaciones entre variables

4.9 Instalación

Para facilitar el uso de SkyStats por parte de usuarios menos técnicos, se ha desarrollado un instalador de un solo clic, que se encarga de instalar todas las dependencias necesarias para ejecutar la aplicación, y de crear un acceso directo para ejecutarla con facilidad.

Antes de ejecutar el instalador, es necesario tener instalado Python 3.9 (la versión usada para el desarrollo) o superior.

En caso de usarse Windows 11, es necesario añadir la carpeta de instalación de Powershell a la variable de entorno PATH, ya que en Windows 11 parece no incluirse por defecto.

Para usuarios de Windows, se ha creado un script “WINDOWS SETUP.bat”, que se puede ejecutar fácilmente haciendo doble clic sobre este.

Esto abrirá una ventana de la terminal de Windows que solicitará permisos de administrador para ejecutar el script de PowerShell “windows-setup.ps1”, dentro de la carpeta “scripts”, que hace lo siguiente:

Instala “virtualenv”, una herramienta que permite crear entornos virtuales de Python:

```
python -m pip install virtualenv
```

Crea un entorno virtual de Python dentro de la carpeta “env”:

```
python -m venv ..\env
```

Activa el entorno virtual, de manera que todas las dependencias se instalarán en él:

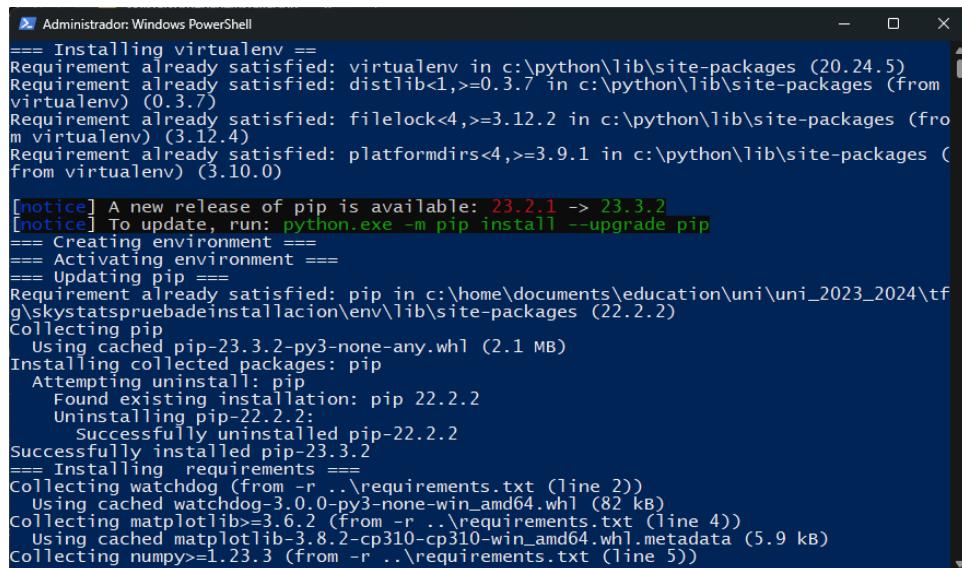
```
..\env\Scripts\Activate.ps1
```

Actualiza pip, el gestor de paquetes de Python, a la última versión:

```
python -m pip install --upgrade pip
```

Instala las dependencias (librerías de Python) necesarias para ejecutar la aplicación:

```
python -m pip install -r ..\requirements.txt
```



```
Administrator: Windows PowerShell
== Installing virtualenv ==
Requirement already satisfied: virtualenv in c:\python\lib\site-packages (20.24.5)
Requirement already satisfied: distlib<1,>=0.3.7 in c:\python\lib\site-packages (from virtualenv) (0.3.7)
Requirement already satisfied: filelock<4,>=3.12.2 in c:\python\lib\site-packages (from virtualenv) (3.12.4)
Requirement already satisfied: platformdirs<4,>=3.9.1 in c:\python\lib\site-packages (from virtualenv) (3.10.0)

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
== Creating environment ==
== Activating environment ==
== Updating pip ==
Requirement already satisfied: pip in c:\home\documents\education\uni\uni_2023_2024\tfg\skystatspruebadeinstalacion\env\lib\site-packages (22.2.2)
Collecting pip
  Using cached pip-23.3.2-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.2.2
    Uninstalling pip-22.2.2:
      Successfully uninstalled pip-22.2.2
Successfully installed pip-23.3.2
== Installing requirements ==
Collecting watchdog (from -r ..\requirements.txt (line 2))
  Using cached watchdog-3.0.0-py3-none-win_amd64.whl (82 kB)
Collecting matplotlib>=3.6.2 (from -r ..\requirements.txt (line 4))
  Using cached matplotlib-3.8.2-cp310-cp310-win_amd64.whl.metadata (5.9 kB)
Collecting numpy>=1.23.3 (from -r ..\requirements.txt (line 5))
```

Figura 4.9.1: Captura del proceso de instalación del entorno virtual

Después, se creará un archivo “SkyStats.bat” que llama al script “run.ps1”, que hace lo siguiente:

Activa el entorno virtual:

```
..\env\Scripts\Activate.ps1
```

Ejecuta la aplicación:

```
python app.py
```

“app.py” se encarga de lanzar el comando de ejecución de un servidor local de Streamlit, con el siguiente comando, que lanza la aplicación en la dirección “<http://localhost:8501>”:

```
streamlit run src/image_plots_ui.py --browser.serverAddress -  
localhost
```

Finalmente, se crea un acceso directo a “SkyStats.bat” con el ícono “logo_256.ico” de la carpeta “assets”, que se puede mover al escritorio, a la barra de tareas o a cualquier otra ubicación para ejecutar la aplicación con un solo clic.

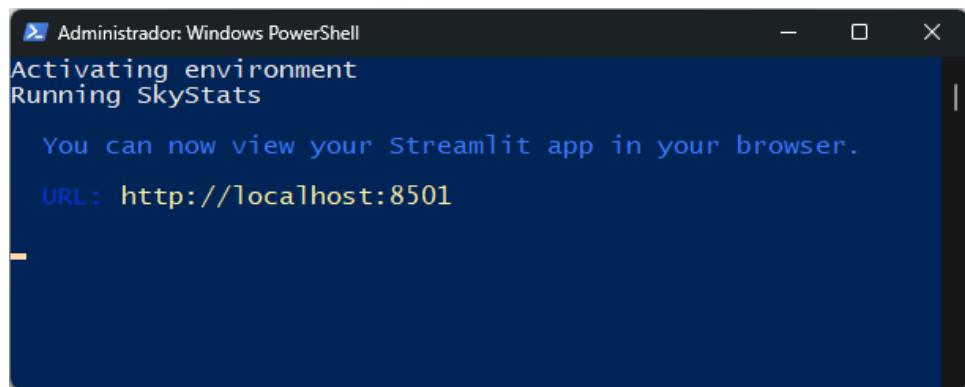


Figura 4.9.2: Captura de la ventana de ejecución después de hacer clic al acceso directo de SkyStats

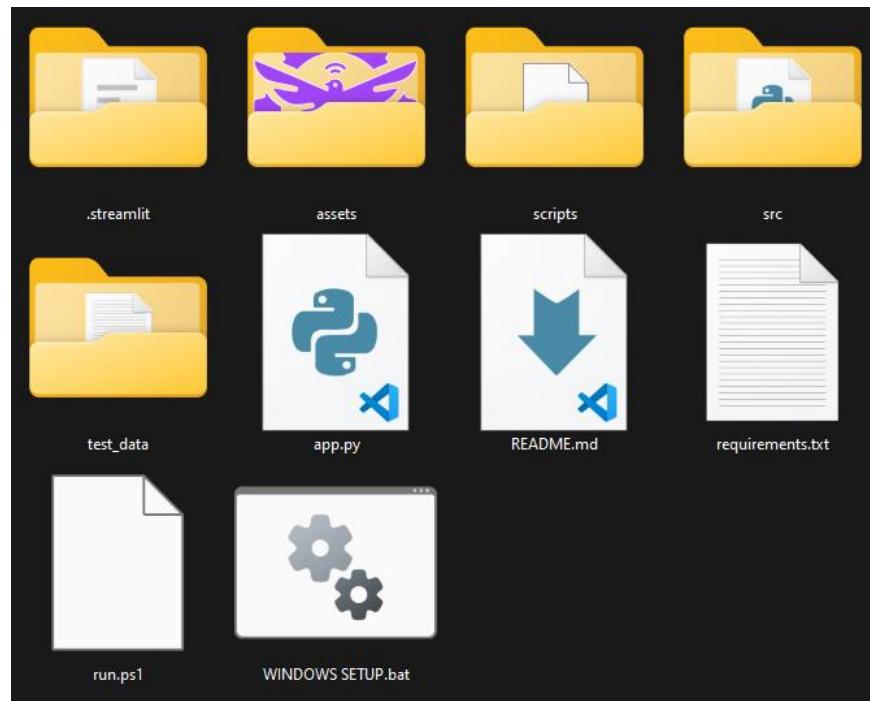


Figura 4.9.3: Carpeta del proyecto previa a la instalación

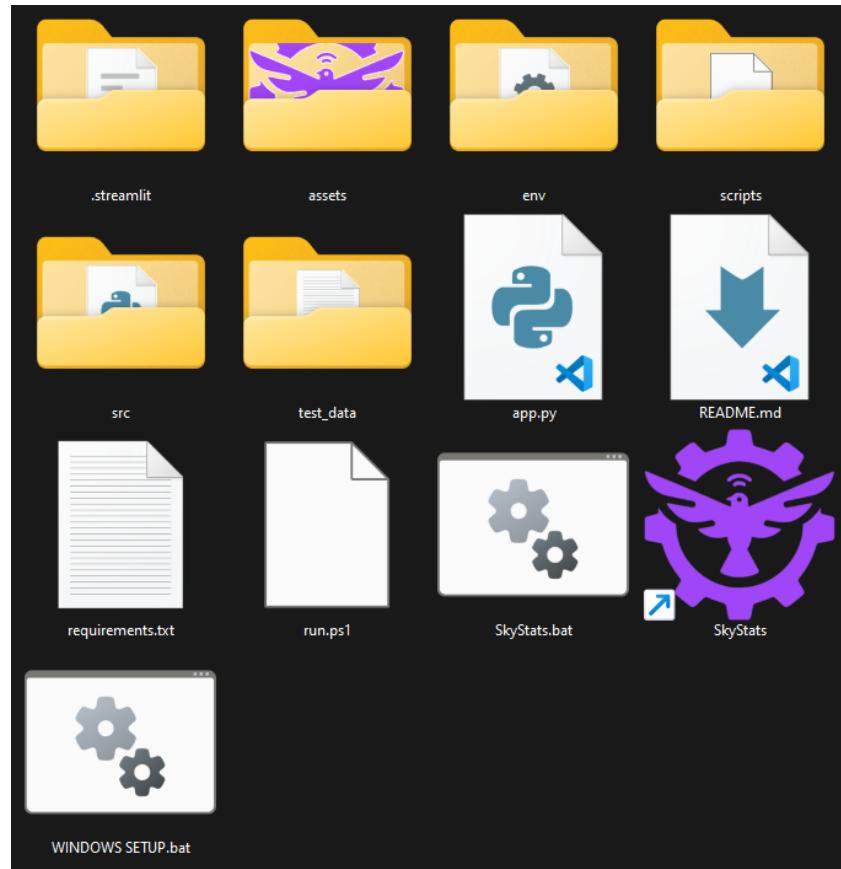


Figura 4.9.4: Carpeta del proyecto posterior a la instalación

En el caso de usar Linux o MacOS, se ha creado un script “unix-setup.sh”, que se ubica dentro de la carpeta “scripts” y realiza un proceso similar al de Windows, pero usando la terminal de Bash en lugar de PowerShell y teniendo en cuenta las diferencias entre los sistemas operativos.

El proceso de instalación y el funcionamiento del programa se ha probado en Windows 10, Windows 11, Ubuntu 20.04 y MacOS Monterey y Ventura, y funciona correctamente en todos ellos.

En el caso de querer realizar el proceso de instalación manualmente, se han incluido las instrucciones necesarias en el fichero “README.md” del proyecto.

5 Resultados y conclusiones

Tras completar el desarrollo de SkyStats, y haberlo usado para analizar cientos de vuelos reales y simulados, llegando a acumular un total de 177 GB de datos de prueba, la conclusión es muy positiva.

En primer lugar, se ha cumplido con creces el objetivo principal de la aplicación, que era optimizar el flujo de análisis de los vuelos. Se han desarrollado varios acercamientos al problema que supone analizar datos de tantos tipos distintos, tanto visuales como estadísticos, y se han implementado de manera que se complementan entre sí y permiten al usuario obtener una visión completa de los datos.

Además, la interactividad con la aplicación ha pasado de ser uno de los aspectos que más dudas me generaban, debido a la dificultad de implementación de los sistemas de eventos y callbacks en las visualizaciones interactivas, a ser uno de los aspectos más destacables de la aplicación, que permiten al usuario interactuar con los datos de una manera muy intuitiva y natural y que poco le tiene que envidiar a implementaciones de este tipo de visualizaciones por parte de aplicaciones profesionales.

Y respecto a las alternativas existentes, se ha logrado crear una aplicación que ofrece unas funcionalidades muy similares a las aplicaciones profesionales tratadas en el apartado 2 sobre el estado del arte, ofreciendo unas opciones de visualización parecidas a las proporcionadas por Droneviewer y Airdata, con la ventaja de que las presentes en SkyStats son interactivas y más personalizables.

Además, se ha logrado el objetivo de que la aplicación funcione sin necesidad de una conexión a internet (exceptuando las funciones que llaman a la API de Mapbox) y de que funcione en múltiples sistemas operativos, incluyendo Windows, Linux y MacOS, lo cual convierte a SkyStats en una aplicación muy versátil y accesible.

Por su parte, como ejercicio académico, el proyecto ha sido muy positivo, ya que me ha permitido poner en práctica muchos de los conocimientos sobre desarrollo de software adquiridos durante la carrera y me ha permitido ampliar mi entendimiento de Python y de algunas de sus librerías como NumPy, Pandas, Matplotlib o Plotly.

Además, el proyecto me ha permitido usar por primera vez un framework a un nivel más avanzado, ya que aunque ya había usado Streamlit previamente, lo había hecho a un nivel muy básico, y en este proyecto he podido profundizar en su uso y en su funcionamiento interno, lo cual me ha dado una visión más amplia de los frameworks en general.

También me ha permitido profundizar en el uso de Git y GitHub, que ya había usado previamente, pero no con tanta profundidad como en este proyecto.

De cara al futuro, el proyecto podría mejorarse de varias maneras:

- Mejorar el rendimiento de la aplicación:
 - Aunque se ha intentado optimizar el código en la medida de lo posible, habiéndose reescrito una buena parte del mismo para mejorar su rendimiento, sigue habiendo ciertas funciones que podrían mejorarse, como la generación de las visualizaciones 3D interactivas con superficies terrestres o la generación del informe de análisis exploratorio de datos.
 - Debido a que el framework Streamlit está pensado para aplicaciones más sencillas con una ejecución más lineal, algunas partes de la interfaz vuelven a cargarse cada vez que se interactúa con la interfaz, lo cual puede hacer que la aplicación se ralentice en algunos momentos. Esto se podría mejorar usando un framework más avanzado como Qt, que permitiría tener un mayor control sobre la interfaz y sobre el flujo de ejecución de la aplicación, aunque para el alcance de este proyecto y las limitaciones de tiempo, Streamlit ha sido una buena elección.
- Investigar posibles alternativas al método de instalación actual, ya que si bien este es bastante sencillo, sigue requiriendo que el usuario tenga instalado Python y que tenga ciertos conocimientos técnicos, por lo que no es tan accesible como podría serlo. Se podrían explorar alternativas como la creación de un ejecutable que instale la aplicación y sus dependencias, o la creación de una aplicación web que permita subir los datos y analizarlos en la nube.
- Terminar funciones como el análisis básico que debido a las restricciones de tiempo ha quedado con una funcionalidad muy escueta.
- Añadir soporte para más drones, ya que actualmente la aplicación sólo soporta drones de DJI y Parrot. También se podría crear un SDK limitado, que facilite la integración de drones de otros fabricantes mediante la implementación de una interfaz genérica.

6 Análisis de impacto

Durante el desarrollo de este TFG se han tenido en cuenta el impacto que los resultados obtenidos pueden tener en diferentes ámbitos, haciendo especial hincapié en los objetivos de desarrollo sostenible de la Agenda 2030 de las Naciones Unidas [46].

En lo personal, la realización de este trabajo ha sido una oportunidad significativa para mi desarrollo profesional, al haberme permitido poner en práctica y desarrollar muchos de los conocimientos adquiridos durante la carrera en un proyecto real, y al haberme ayudado a desarrollar mis habilidades de investigación, análisis y resolución de problemas.

Respecto al impacto empresarial, al haberse diseñado parte de la aplicación durante mi estancia en la empresa Pix4D, puedo confirmar que la aplicación, incluso en un estado de desarrollo temprano, fue bien recibida por los pilotos de drones de la empresa, que la usaron para analizar vuelos reales con resultados satisfactorios, ahorrando tiempo y esfuerzo a la hora de analizar los vuelos.

Desde entonces, para la realización de este TFG, se ha seguido trabajando en la aplicación, mejorando su rendimiento y el funcionamiento de algunas herramientas.

En cuanto a los aspectos sociales, económicos, medioambientales y culturales, la aplicación no tiene un impacto directo en ninguno de ellos, al tratarse de una herramienta técnica con un uso muy específico. Sin embargo, sí que se puede afirmar que la aplicación puede tener un impacto indirecto, ya que al facilitar el análisis de los vuelos, se puede mejorar la eficiencia de todo tipo de operaciones de vuelo, desde inspecciones de infraestructuras hasta misiones de búsqueda y rescate, pasando por misiones de cartografía, de inspección agrícola o forestal, de seguridad, o de conservación de patrimonio mediante el uso de fotogrametría, entre otras.

Como consecuencia, ayuda indirectamente a lograr los siguientes objetivos de desarrollo sostenible recogidos en la agenda 2030:

- 2. Hambre cero: Mediante el uso de drones para la inspección agrícola, se puede mejorar la eficiencia de la agricultura, lo cual puede ayudar a reducir la escasez de alimentos.
- 6. Agua limpia y saneamiento: Mediante el uso de drones para la inspección de infraestructuras, se puede mejorar la eficiencia de la gestión de las infraestructuras de agua, lo cual puede ayudar a mejorar el acceso al agua potable.
- 7. Energía asequible y no contaminante: La inspección de infraestructuras energéticas renovables como parques eólicos o plantas solares mediante el uso de drones puede ayudar a mejorar la eficiencia de la generación de energía renovable.
- 8. Trabajo decente y crecimiento económico y 9. Industria, innovación e infraestructura: El uso de drones ha supuesto una revolución en múltiples industrias, facilitando la realización de tareas que antes eran muy costosas o incluso imposibles, lo cual ha ayudado a mejorar la eficiencia de las empresas, la seguridad de los trabajadores y como consecuencia ha ayudado a mejorar el crecimiento económico.

- 13. Acción por el clima, 14. Vida submarina y 15. Vida de ecosistemas terrestres: La inspección mediante drones de superficies naturales como bosques o zonas costeras puede ayudar a detectar cambios en el clima y a tomar medidas para mitigarlos, logrando así reducir el impacto del cambio climático en la flora y fauna.

Por otro lado, el uso de drones también tiene aspectos negativos, como el impacto medioambiental que supone su uso, debido a la contaminación acústica y por el uso de baterías, o el impacto social que supone la pérdida de puestos de trabajo debido a la automatización de tareas que antes realizaban personas, como la inspección de infraestructuras o la agricultura.

Los drones enfocados al consumidor también se están fabricando con fines militares, lo cual puede tener un impacto negativo en el ámbito social y económico, ya que puede facilitar la guerra y la vigilancia masiva.

7 Bibliografia

- [1] V. Forgeard, “The Ultimate List: 50 Reasons Why Are Drones Important” Brilliantio, 04-Sep-2023. [Online]. Available: <https://brilliantio.com/why-are-drones-important>.
- [2] J. Walker. (2019, January 30). Industrial Uses of Drones – 5 Current Business Applications [Online]. Available: <https://emerj.com/ai-sector-overviews/industrial-uses-of-drones-applications>.
- [3] Occupational Safety and Health Administration. (n.d.). Communication Towers [Online]. Available: <https://www.osha.gov/communication-towers>.
- [4] PwC Poland. (2016). Clarity from above: transport infrastructure [Online]. Available: <https://www.pwc.pl/en/publikacje/2016/clarity-from-above-transport-infrastructure.html>.
- [5] P. Murginski, “Drones and the European Green Deal: Embracing Technology for a Sustainable Future,” Deloitte, [Online]. Available: <https://www2.deloitte.com/bg/en/pages/tax/articles/drones-and-the-european-green-deal-embracing-technology-for-a-sustainable-future.html>.
- [6] European Commission. (2022, November). Drone Strategy 2.0 [Online]. Available: https://transport.ec.europa.eu/system/files/2022-11/COM_2022_652_drone_strategy_2.0.pdf.
- [7] Zion Market Research. (2023, March 15). Global Drone Market Is Predicted to Gain Revenue of About USD 260.5 Billion By 2030 [Online]. Available: <https://www.zionmarketresearch.com/news/global-drone-market-size>.
- [8] Spherical Insights & Consulting. (2023, May). Global Commercial Drone Market Size, Share, Analysis - 2032 [Online]. Available: <https://www.sphericalinsights.com/reports/commercial-drone-market>.
- [9] Pix4D, “PIX4Dcapture Pro: Professional drone flight and mission planning mobile app for 3D mapping,” Pix4D, [Online]. Available: <https://www.pix4d.com/es/producto/pix4dcapture/>.
- [10] MarketsandMarkets Research Pvt. Ltd., “DJI (China) and Parrot Drone SAS (France) are the Major Players in the Small Drone Market (2022-2027)” GlobeNewswire, Nov. 16, 2022. [Online]. Available: <https://www.globenewswire.com/en/news-release/2022/11/16/2556829/0/en/DJI-China-and-Parrot-Drone-SAS-France-are-the-Major-Players-in-the-Small-Drone-Market-2022-2027.html>.
- [11] C.-C. Yang, H. Chuang, and D. Kao, “Drone forensic analysis using relational flight data: A case study of DJI Spark and Mavic Air,” Procedia Computer Science, vol. 192, pp. 1359–1368, Jan. 2021, doi: [10.1016/j.procs.2021.08.139](https://doi.org/10.1016/j.procs.2021.08.139).
- [12] R. Kumar and A. K. Agrawal, “Drone GPS data analysis for flight path reconstruction: A study on DJI, Parrot & Yuneec make drones,” Forensic Science International: Digital Investigation, vol. 38, p. 301182, Sep. 2021, doi: [10.1016/j.fsid.2021.301182](https://doi.org/10.1016/j.fsid.2021.301182).
- [13] T. C. Mallick, M. A. I. Bhuyan, and M. S. Munna, “Design & implementation of an UAV (Drone) with flight data record,” ICISSET, Oct. 2016, doi: [10.1109/iciset.2016.7856519](https://doi.org/10.1109/iciset.2016.7856519).

- [14] R. Barna, K. Solymosi, and E. Stettner, “Mathematical analysis of drone flight path,” Journal of Agricultural Informatics, Dec. 2019, [doi: 10.17700/jai.2019.10.2.533](https://doi.org/10.17700/jai.2019.10.2.533).
- [15] “Automated Drone Inspections,” Drone Harmony, [Online]. Available: <https://droneharmony.com/>.
- [16] “DroneViewer,” DroneViewer, [Online]. Available: <https://mydroneviewer.com/>.
- [17] M. Singer Phantom Help, “DJI Flight Log Viewer” Phantom Help, [Online]. Available: <https://www.phantomhelp.com/logviewer/upload/>.
- [18] M. Singer Phantom Help, “DJI drone flight log viewer | Flight Reader” Flight Reader, [Online]. Available: <https://www.flighthreader.com/>.
- [19] “Drone Data Management and Flight Analysis” Airdata UAV, [Online]. Available: <https://airdata.com/>.
- [20] “Python 3.12.1 documentation” Python, (2024) [Online]. Available: <https://docs.python.org/3/>.
- [21] Matplotlib (2024). “Matplotlib: Visualization with Python” [Online]. Available: <https://matplotlib.org/>.
- [22] NumPy (2023). “The fundamental package for scientific computing with Python” [Online]. Available: <https://numpy.org/>.
- [23] pandas (2024). “pandas documentation” [Online]. Available: <https://pandas.pydata.org/docs/>.
- [24] Pillow (2024). “Pillow documentation” [Online]. Available: <https://pillow.readthedocs.io/en/stable/>.
- [25] Dash (2024). “Plotly: Low-Code Data App Development” [Online]. Available: <https://plotly.com/>.
- [26] SciPy (2023, November 18). “SciPy documentation” [Online]. Available: <https://docs.scipy.org/doc/scipy/>.
- [27] trimesh (2024). “trimesh documentation” [Online]. Available: <https://trimesh.org/>.
- [28] Watchdog (2023). “Watchdog documentation” [Online]. Available: <https://python-watchdog.readthedocs.io/en/stable/>.
- [29] Streamlit (2024). “Streamlit Docs” [Online]. Available: <https://docs.streamlit.io/>.
- [30] Mapbox (2023). “Maps: API Docs” [Online]. Available: <https://www.mapbox.com/>.
- [31] Adobe (2024). “Adobe Illustrator | Software de ilustración digital y vectores gráficos” [Online]. Available: <https://www.adobe.com/es/products/illustrator.html>.
- [32] Google (2023). “Material Design” [Online]. Available: <https://m3.material.io/>.
- [33] E. Contributors, “Exiv2 - Image metadata library and tools,” Nov. 11, 2023. [Online]. Available: <https://exiv2.org/tags.html>.

- [34] Adobe, “File management, metadata integration | Adobe Extensible Metadata Platform (XMP).” Available: <https://www.adobe.com/products/xmp.html>.
- [35] OASIS, Organization for the Advancement of Structured Information Standards, “Extensible Metadata Platform (XMP),” Cover Pages, Jun. 23, 2004. Available: <http://xml.coverpages.org/xmp.html>.
- [36] Parrot, “Embedded Photo Metadata - GroundSDK Tools 7.0,” developer.parrot.com, 2021. Available: <https://developer.parrot.com/docs/pdraw/photo-metadata.html#xmp-metadata>.
- [37] DJI, “Zenmuse P1 User Manual, v1.2” (2021). [Online]. Available: https://dl.djicdn.com/downloads/Zenmuse_P1/20210510/Zenmuse_P1%20User%20Manual_EN_v1.2_3.pdf
- [38] Pix4D Support. (2020). “Yaw, Pitch, Roll and Omega, Phi, Kappa angles” [Online]. Available: <https://support.pix4d.com/hc/en-us/articles/202558969-Yaw-Pitch-Roll-and-Omega-Phi-Kappa-angles>
- [39] Wikipedia, (2024). “Rotation matrix”. [Online]. Available: https://en.wikipedia.org/wiki/Rotation_matrix
- [40] Wikipedia, (2024). “Gimbal lock”. [Online]. Available: https://en.wikipedia.org/wiki/Gimbal_lock
- [41] Mapbox, “Access Elevation Data.” Mapbox Tilesets Guide. [Online]. Available: <https://docs.mapbox.com/data/tilesets/guides/access-elevation-data/>.
- [42] Mapbox, “Realistic terrain with custom styling - maps for developers,” Medium, Jun. 26, 2018. Available: <https://blog.mapbox.com/realistic-terrain-with-custom-styling-ce1fe98518ab>.
- [43] Möller, T., & Trumbore, B. (2005). “Fast, minimum storage ray/triangle intersection” [Online]. Available: <https://cadxfem.org/inf/Fast%20MinimumStorage%20RayTriangle%20Intersection.pdf>
- [44] Wikipedia contributors, “Delaunay triangulation,” Wikipedia, Oct. 09, 2023. Available: https://en.wikipedia.org/wiki/Delaunay_triangulation.
- [45] Jonathan Richard Shewchuk, “Triangulation algorithms and data structures,” CMU School of Computer Science, Aug. 12, 1996. Available: <https://www.cs.cmu.edu/~quake/tripaper/triangle2.html>.
- [46] M. J. Gamez, “Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible,” Desarrollo Sostenible, May 24, 2022. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>.

Este documento esta firmado por

	Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jan 15 04:25:13 CET 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)