PLANIFICADOR AVANZADO DE MISIONES COOPERATIVAS DE UAVS PARA ENTORNOS INDUSTRIALES, A PARTIR DE MODELOS 3D

Autores:

ALEJANDRO MUÑOZ CUEVA
PATRICIA LÓPEZ TORRES
ALICIA ARCE
JUAN AGUILAR GUISADO
RUBÉN RODRÍGUEZ APARICIO
JOSÉ LUIS BARRERA TRANCOSO
RICARDO GALÁN DE VEGA

Planificador avanzado de misiones cooperativas de UAVs para entornos industriales, a partir de modelos 3D

Alejandro Muñoz Cueva, Patricia López Torres, Alicia Arce, Juan Aguilar Guisado, Rubén Rodríguez Aparicio, José Luis Barrera Trancoso y Ricardo Galán de Vega

FUNDACIÓN AYESA

Área temática: Aplicaciones.

El avance reciente en las tecnologías relativas a los UAVs (*Unmanned Aerial Vehicles*) hace que sea posible el desarrollo de aplicaciones cada vez más inteligentes. Debido al auge experimentado por los UAVs, es tecnológicamente posible emplear dichas plataformas aéreas para examinar e inspeccionar distintos tipos de entornos industriales de manera autónoma, facilitando así las labores de mantenimiento y reduciendo los riesgos para el operario y los costes para la planta. Este trabajo propone una herramienta capaz de realizar planificaciones cooperativas de alto nivel para misiones a realizar por uno o varios UAVs en entornos industriales, mediante la implementación y ejecución conjunta de diferentes algoritmos inteligentes.

Los algoritmos desarrollados planifican de forma inteligente las misiones para vuelos autónomos cooperativos en entornos industriales, basándose en modelos 3D de las instalaciones, obteniendo el conjunto óptimo de acciones y medidas necesarias que minimiza el consumo de recursos y los riesgos de los agentes implicados. Los modelos 3D pueden incluir información de la planta, por ejemplo, zonas catalogadas de incertidumbre o riesgo. Esta parametrización junto con las características de las plataformas aéreas (sensores a bordo, dimensiones, etc.) permitirá planificar una trayectoria óptima libre de colisiones en el espacio. La herramienta se completa con una interfaz en la que el operario puede definir de forma fácil las misiones a realizar, así como las plataformas aéreas disponibles, los sensores embarcados y sus características, sacando así partido de la flexibilidad de los algoritmos desarrollados, y adaptándolos a la situación deseada.

1. INTRODUCCIÓN

Las tecnologías relativas a los UAVs (*Unmanned Aerial Vehicles*) han experimentado un avance exponencial en los últimos años. Cada vez se dedican más esfuerzos a desarrollar herramientas y tecnologías asociadas a estas aeronaves, de forma que ya es posible su empleo en múltiples campos de aplicación, y se espera que esta tendencia siga en aumento.

En la actualidad, es tecnológicamente posible emplear los UAVs para inspeccionar de forma autónoma distintos tipos de entornos industriales, consiguiendo reducir los posibles riesgos a los que tendría que hacer frente un operario que realizara estas inspecciones de forma tradicional.

Además, una planificación óptima de las tareas a realizar por los UAVs, así como la forma de realizarlas, puede permitir reducir los costes asociados a estas tareas y aumentar la seguridad. A medida que se incrementan las capacidades tecnológicas de los UAVs, se produce también un incremento en la necesidad de elaborar una planificación óptima de las misiones a realizar, con el fin de aprovechar al máximo las ventajas que estas aeronaves ofrecen.

Existen soluciones software en el mercado que asisten al operario de un UAV a la hora de planificar una misión o una trayectoria, como el *Mission Planner*. Sin embargo, en la mayoría de casos, este tipo de herramientas aportan resultados no particularizados, lo que impide encontrar soluciones óptimas para escenarios complejos.

Este trabajo propone una herramienta capaz de realizar planificaciones cooperativas de alto nivel para misiones a realizar por uno o varios UAVs en entornos industriales, mediante la implementación y ejecución conjunta de diferentes algoritmos inteligentes. Para ello, la herramienta se basa en modelos 3D de las instalaciones en las que se pretende realizar la inspección, así como en los requisitos de las misiones a realizar, definidos por el usuario a cargo del mantenimiento. Además, los algoritmos desarrollados tienen en cuenta el número de UAVs que conforman la flota disponible, así como las características y restricciones de cada uno de ellos (sensores de los que dispone, dimensiones, autonomía, etc), con el objeto de planificar las acciones a realizar por cada uno de los UAVs y el camino que deben recorrer. Estas planificaciones están basadas en el conocimiento del entorno a través del modelo 3D de la planta, de forma que se consigan completar las inspecciones definidas de forma óptima, minimizando el consumo de recursos y los riesgos para el operario.

Los algoritmos inteligentes desarrollados permiten una amplia flexibilidad a la hora de definir los requisitos de las misiones, por ejemplo, mediante la definición de restricciones adicionales, como pueden ser restricciones temporales, la inclusión de puntos de recarga, o la distinción de distintos tipos de inspecciones a realizar, buscando la solución óptima en cada caso.

1.1. Estado del arte de herramientas software para modelado 3D

El uso de los modelos 3D en la actualidad está en auge gracias al avance de la tecnología y al incremento de potencia de cómputo de los últimos años. Ahora es posible desarrollar entornos virtuales con un nivel de detalle similar al de la realidad, o incluso convertir zonas reales a modelos 3D de manera fidedigna. Principalmente ha habido tres sectores que han intervenido en la mejora de estas tecnologías: el sector de la arquitectura, el sector de la ingeniería y el sector de los videojuegos.

Existen diferentes formatos para el almacenaje de modelos 3D y podemos encontrar formatos cerrados (propietarios), como puede ser el formato .fbx de Autodesk, o abiertos, como el formato .obj de Wavefront. En ambos casos la información almacenada suele ser muy parecida o similar: los vértices del modelo 3D y las caras que forman estos vértices.

El tratamiento de modelos 3D es una operación costosa computacionalmente hablando, a nivel de almacenamiento y tiempo de procesamiento, y se necesitan maneras de trabajar con ellos de forma eficiente.

La voxelización consiste en el procesado de un modelo 3D para su transformación en un modelo formado por cubos, reduciendo la complejidad del modelo y facilitando su tratamiento. Visualmente consiste en “pixelar” un modelo 3D, tal y como se muestra en la Figura 1 y la Figura 2:

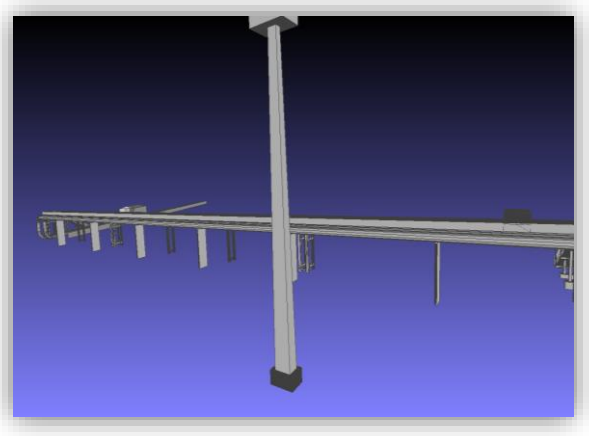

Figura 1: Modelo 3D de un puente

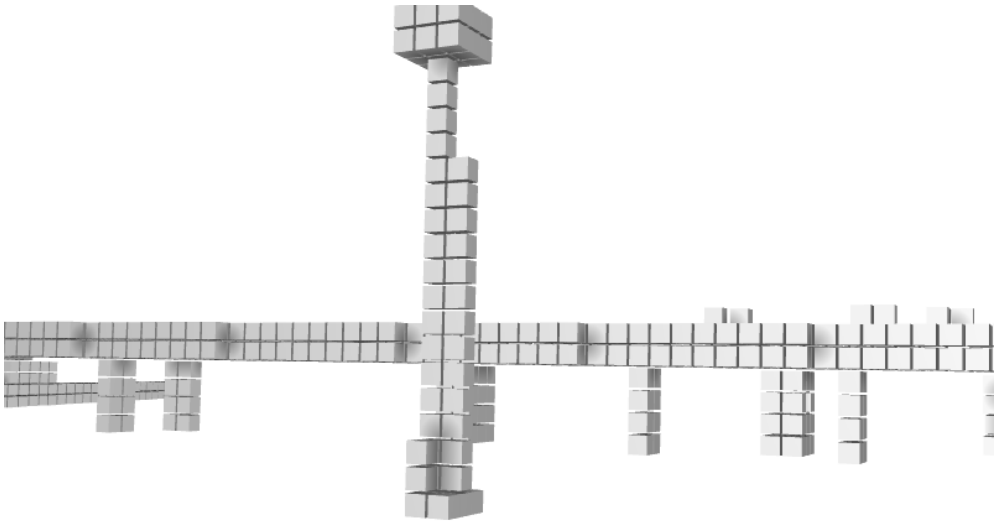

Figura 2: Voxelizado del modelo 3D del puente

A primera vista puede parecer que se añaden vértices y caras innecesarias, pero este tipo de modelo 3D puede ser almacenado de forma más simple que el resto ya que el tamaño de todos los cubos de la matriz es siempre el mismo: bastaría con guardar el tamaño del lado del cubo y qué cubos existen dentro del voxelizado, mediante sus índices. En la Figura 3 se puede observar el voxelizado de un polígono, el cual es extrapolable a las 3 dimensiones:

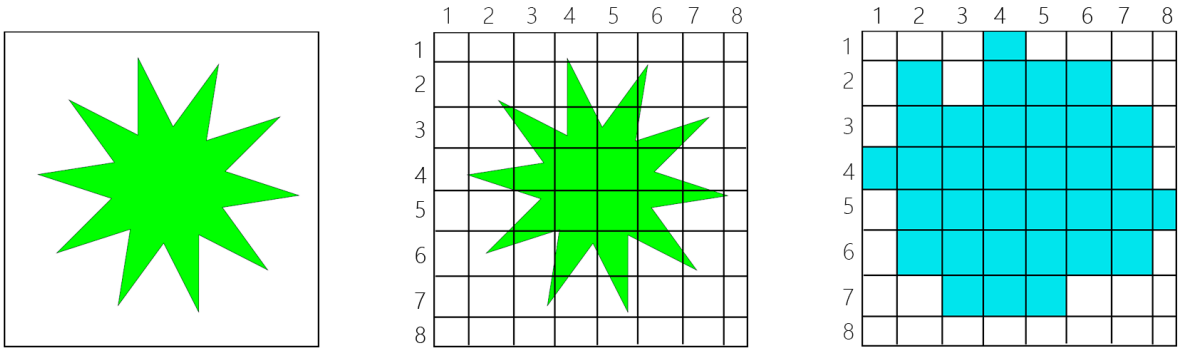

Figura 3: Ejemplo voxelizado 2D.

No es necesario guardar cada vértice de cada cuadrado, solo sus índices: (1, 4), (2, 2), (2, 4), (2, 5), (2, 6) …y el tamaño del lado del cuadrado: 1.

A la hora de voxelizar un modelo 3D se pueden obtener dos resultados distintos: un voxelizado superficial o un voxelizado completo. El voxelizado superficial solo tiene en cuenta el perímetro de un polígono del modelo 3D y no su interior, por lo que puede provocar voxelizados huecos. El voxelizado completo es aquel que sí tiene en cuenta el interior de los modelos 3D y los rellena también de voxels.

En la Figura 4 se puede ver, usando un polígono, la diferencia entre el voxelizado superficial y el voxelizado completo.

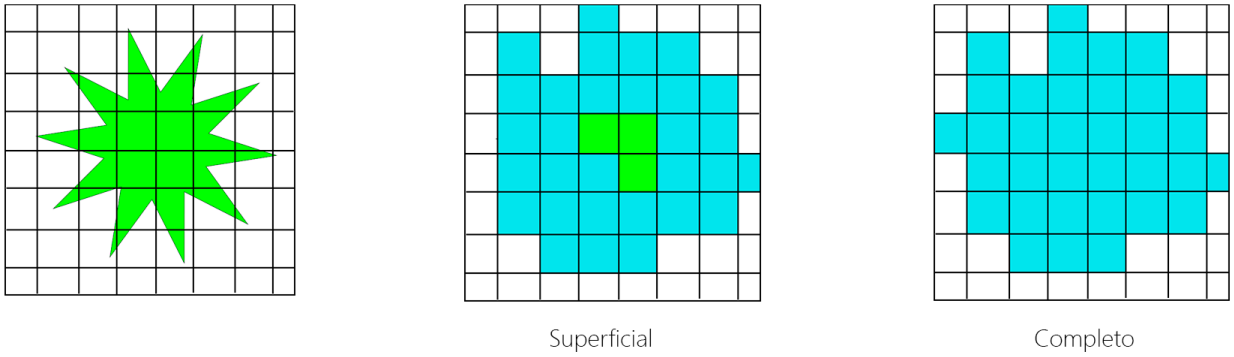

Figura 4: Voxelizado superficial y completo.

En el voxelizado superficial se marcan como voxels aquellas casillas que intersecan con el perímetro del polígono mientras que en el voxelizado completo se marcan todas las casillas que intersecan con el polígono, no solo con su perímetro.

Además de estos tipos de voxelizado, existen métodos para obtener voxelizados que no son exactos. El voxelizado aproximado consiste en obtener un voxelizado donde se añaden voxels sobrantes, o faltan algunos por marcar; el voxelizado conservativo es el que puede añadir voxels sobrantes, pero no falta ninguno por marcar; y el voxelizado exacto es el que marca los necesarios y ninguno más.

El procesamiento de un modelo 3D para obtener su voxelizado se puede realizar en dos sitios diferentes: en la unidad central de procesamiento (CPU) o en la unidad de procesamiento gráfico (GPU).

En base a todo lo que hemos visto hasta ahora, se pueden categorizar los algoritmos de voxelizado dependiendo de dónde se procesen y del resultado obtenido. Existen estudios donde se han probado diferentes métodos y algoritmos, que se pueden categorizar según la Tabla 1:

Técnica	Procesamiento	Tipo de voxelizado	Precisión
Akenine-Möller, 2001 [1]	CPU	Superficial	Exacta
Eisemann & Décoret, 2006	GPU	Superficial	Aproximada
Eisemann & Décoret, 2008 [2]	GPU	Superficial + completo	Aproximada
Dong et al. , 2004a	GPU	Superficial	Aproximada

Tabla 1: Métodos de voxelizado.

1.2. Estado del arte de algoritmos de planificación de trayectorias

El algoritmo encargado de calcular la trayectoria a seguir por una serie de UAVs para realizar la misión definida por el usuario de la mejor forma posible, es uno de los aspectos más críticos de la herramienta presentada en ese artículo.

El problema de la planificación de trayectorias para robots móviles ha sido ampliamente estudiado en el mundo de la robótica, al tratarse de un problema tradicional. Como resultado, se han presentado gran cantidad de estrategias a lo largo de los años, que pretenden aportar soluciones de formas muy diversas. A continuación, se presentan algunas de las técnicas más populares.

Es habitual en las estrategias de planificación realizar algún tipo de simplificación del entorno en el que se desea trabajar. Un ejemplo habitual es la reducción a grafos. Un grafo es un conjunto de nodos, que representan los posibles destinos, conectados a través de arcos, que representan las posibles conexiones y que llevan asociado el coste de cada conexión (a menudo, distancia). La Figura 5 muestra la reducción de un mapa a un grafo muy simple.

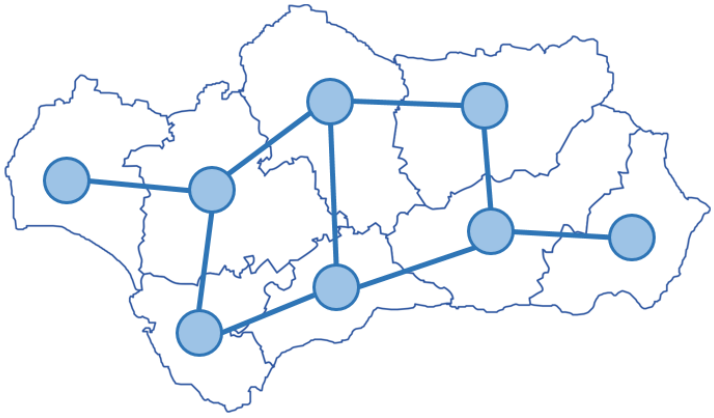

Figura 5: Representación de un grafo, en el que cada nodo representa a una provincia de Andalucía.

Una técnica de planificación muy popular es el algoritmo de Dijkstra, que es un algoritmo de búsqueda en grafos. Mediante este algoritmo es posible calcular la ruta más corta entre un vértice origen y el resto de los vértices de un grafo del que se conocen los pesos en cada arista. La idea en la que se basa este algoritmo es en ir explorando todos los caminos que parten del nodo o vértice origen hacia los demás nodos. En el momento en el que se encuentra el camino más corto que conecta el nodo de origen con el destino, el algoritmo se detiene. El principal inconveniente de esta técnica es que, si el campo de búsqueda es demasiado grande, el tiempo de computación es muy elevado, lo que limita su aplicación en muchas situaciones reales. Por el contrario, su principal ventaja es que, al tratarse de un algoritmo de búsqueda exhaustiva, garantiza que se encuentra la solución óptima en el caso de que ésta exista.

Debido a la gran carga computacional de este algoritmo, surgen otras técnicas que pretenden encontrar la ruta óptima entre dos nodos de un grafo explorando para ello el menor número de nodos posible. En esta línea surgió el algoritmo A* (*A-Star*) [3], que también se trata de una técnica de búsqueda en grafos que pretende encontrar el camino óptimo entre dos nodos, pero realiza para ello una **búsqueda orientada**. Es decir, el algoritmo explora sólo aquellos nodos que encuentra más prometedores para alcanzar al nodo final deseado.

En cada iteración, el algoritmo asocia un coste a cada uno de los nodos vecinos a aquel que está evaluando (empezando por el nodo origen), y los añade a la lista de nodos abiertos, implementada como una cola de prioridad en función del coste asociado a cada elemento de la lista. En la siguiente iteración, se considera como nodo a evaluar aquel nodo de la lista de abiertos que tiene asociado el coste menor, y se repite el procedimiento de la iteración anterior (dejando este nodo de pertenecer a la lista de abiertos y pasando a pertenecer a la de cerrados o evaluados), hasta que alguna de las conexiones acabe en el nodo de destino.

Hasta este punto, el algoritmo es idéntico al de Dijkstra, pero la diferencia fundamental radica en la forma de calcular el coste a asociar a cada nodo para añadirlo a la lista de abiertos, que viene dado por la función $f(n) = C(n) + h'(n)$. Según esta expresión, el coste *f(n)* asociado a un nodo *n* se calcula como el coste acumulado de todos los arcos del grafo desde el nodo inicial hasta el nodo *n*, llamado *C(n)* en la expresión anterior, más un término heurístico *h(n)* que se basa en una estimación del coste necesario para llegar al nodo final desde el nodo *n* que está siendo evaluado. Este término heurístico

es el que permite dirigir la búsqueda hacia el nodo objetivo, asegurando que la solución del algoritmo será óptima siempre que la estimación *h(n)* no sea inferior al coste real necesario para llegar al nodo final desde el nodo *n*. Por otro lado, mientras más pequeño sea el valor del término heurístico, mayor número de nodos visita el algoritmo y por lo tanto mayor tiempo de computación, hasta el caso extremo en el que se toma *h(n)=0*, lo que se corresponde exactamente con el algoritmo de Dijkstra.

Otra técnica popular es la planificación basada en campos potenciales [4], que no se trata de un algoritmo de búsqueda en grafos. Este algoritmo es un método reactivo que consiste en representar al robot como una partícula sometida a diferentes fuerzas de atracción y repulsión, de forma que los obstáculos ejercen fuerzas de repulsión sobre el mismo y el punto objetivo ejerce una fuerza de atracción. Dependiendo de la disposición de cada punto del espacio respecto al punto objetivo y respecto a los obstáculos, se define un valor de potencial para cada punto, siendo la dirección de desplazamiento del robot escogida en función del vector gradiente de este campo potencial, de forma que el movimiento siempre se dirija hacia puntos de menor potencial. Al tratarse de un método reactivo, los campos potenciales no aseguran la obtención del camino óptimo, además de presentar como principal inconveniente la posibilidad de caer en mínimos locales del campo potencial. Por otro lado, su principal ventaja es que se trata de un método muy rápido y que requiere poca capacidad de cálculo.

Otro algoritmo de planificación de caminos popular en determinadas situaciones es el llamado RRT (*Rapidly Exploring Random Trees*) [5]. Este algoritmo se caracteriza por ser puramente aleatorio y encontrar una solución de forma rápida. La desventaja que presenta el RRT es que el camino que devuelve el algoritmo, al basarse en una exploración aleatoria, no tiene por qué ser el óptimo. El algoritmo devuelve como trayectoria la primera solución que encuentra, y debido al carácter aleatorio de la búsqueda, en un mismo escenario y con los mismos puntos de inicio y fin, la solución será distinta cada vez que se ejecute el RRT. Esta técnica por tanto es adecuada para problemas en los que se valora más la velocidad al hallar una solución que la calidad de la misma.

La estrategia más básica del algoritmo RRT pretende explorar un espacio y encontrar las áreas libres de obstáculos. Con este propósito se construye un árbol de configuraciones que crece hacia direcciones aleatorias explorando el mapa desde un punto dado. En cada iteración el algoritmo añade una rama de una longitud determinada en una dirección aleatoria. Para aplicar este algoritmo a la planificación de trayectorias, se crean dos árboles de forma que uno se extienda desde el punto inicial y el otro desde el final, acabando el algoritmo cuando los dos árboles se conectan.

2. ARQUITECTURA DE LA HERRAMIENTA

Este trabajo presenta la plataforma software desarrollada en Fundación Ayesa para la ejecución de las planificaciones y la gestión de la información relevante para la planificación. La plataforma está formada por los siguientes componentes principales:

- Herramienta para el usuario: Este servicio web permitirá a los usuarios definir, planificar y ejecutar las misiones e interactuar con la plataforma. El servicio web está desarrollado siguiendo una arquitectura cliente-servidor con una conexión a través de una API REST segura.
- Herramienta de voxelizado: Esta herramienta se encargará de extraer el mallado 3D asociado al modelo 3D, representativo de la instalación sobre la que se va a realizar la inspección.

- Planificador Inteligente de Misiones Cooperativas: Componente que definirá el plan óptimo para realizar las inspecciones sobre unas instalaciones a partir de datos como el voxelizado, los puntos a visitar o las características físicas de los UAVs disponibles.

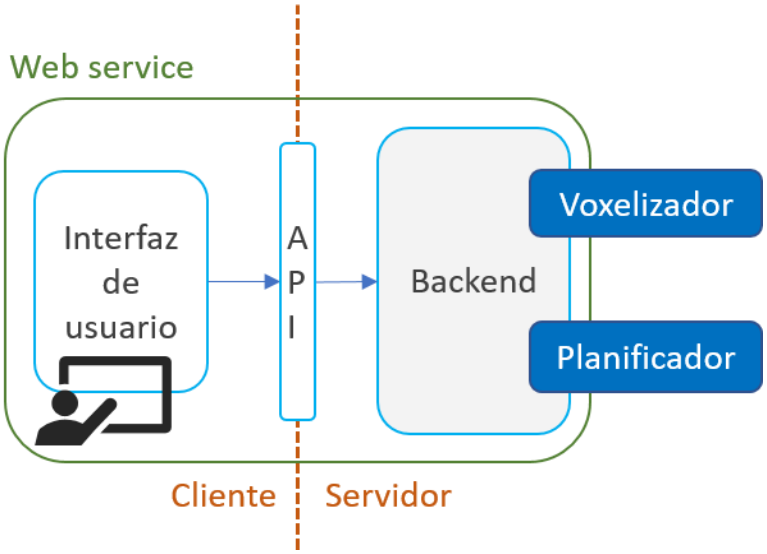

Figura 6: Diagrama de la estructura de la herramienta.

El proceso de ejecución de una misión sigue varios pasos secuenciales. En primer lugar, los usuarios introducen los datos necesarios para la planificación (UAVs disponibles, modelo 3D y puntos de misión) a través del servicio web. Durante una fase de preprocesamiento, el sistema voxeliza el modelo 3D para que los algoritmos reciban un mapa 3D de espacios libres y ocupados. Los algoritmos recibirán el mapa 3D y la parametrización para generar el plan óptimo desde el punto de vista de la autonomía de los UAVs y del número de aeronaves utilizadas. Las planificaciones se pueden ejecutar varias veces con diferentes parámetros de entrada hasta que, finalmente, se obtiene la planificación deseada. Una vez obtenida la planificación final, se lanza la ejecución sobre los UAVs que vayan a inspeccionar las instalaciones. La comunicación con los vehículos se realiza a través de una API por la que reciben los comandos y los puntos de misión para que realicen la misión de manera autónoma.

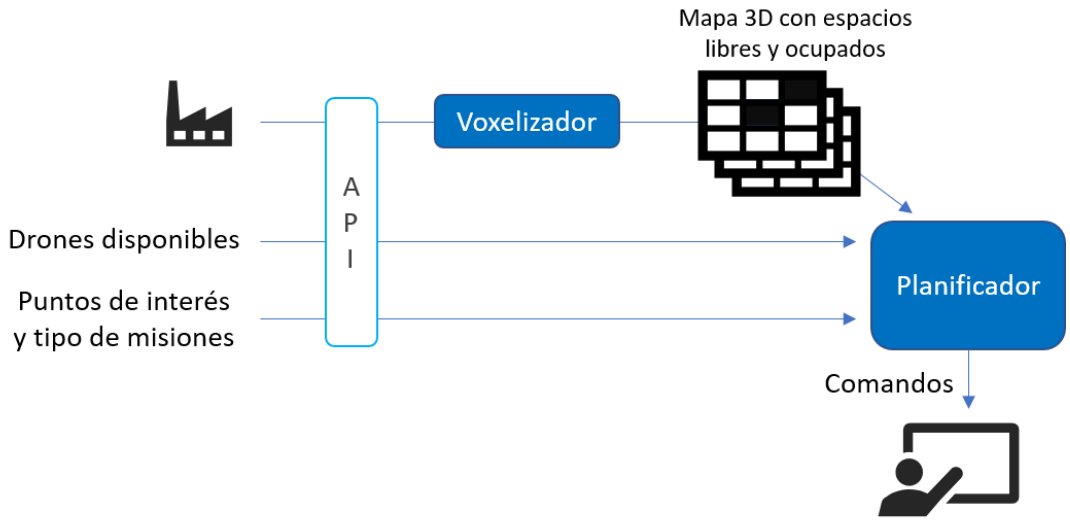

Figura 7: Diagrama de flujo de información del planificador.

3. ALGORITMOS

3.1. PLATAFORMA

Se ha desarrollado un servicio web que permitirá a los usuarios interactuar con el planificador inteligente de misiones cooperativas de manera gráfica. La herramienta se ha desarrollado poniendo especial atención en la experiencia de usuario y siguiendo las guías de diseño estipuladas por *material design*.

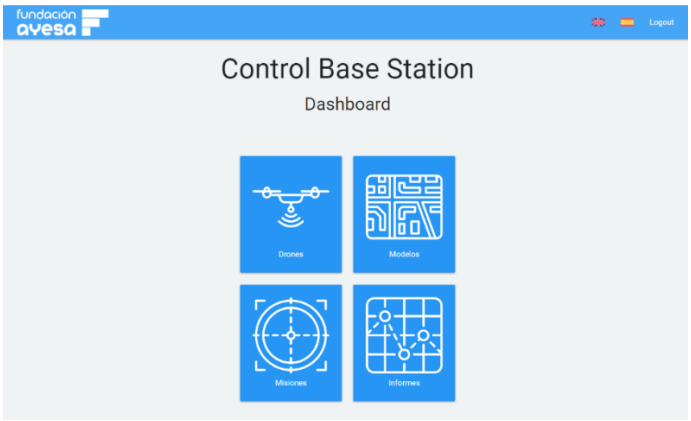

Figura 8:Pantalla de inicio de la plataforma.

La plataforma abstrae a los usuarios de conocimientos adicionales para realizar las planificaciones. Solamente se necesita dar de alta las características físicas de los UAVs y las instalaciones. Los puntos de interés que deben visitar los vehículos serán añadidos visualmente mediante la interfaz gráfica, a través del visor de misión avanzado.

3.1.1. Preparación de los datos

Para realizar una planificación, previamente se deben haber dado de alta los UAVs disponibles y los sensores que puedan tener asociados.

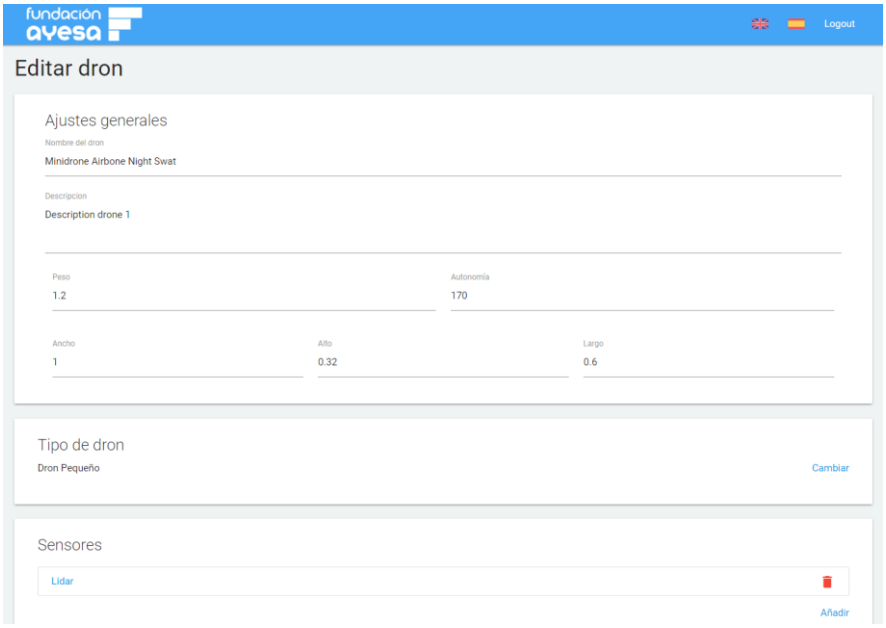

Figura 9: Registro de un UAV en la plataforma.

El sistema recibirá un modelo 3D de la instalación que se va a inspeccionar. Para registrarlo en la plataforma hay que subir al servidor el fichero (fichero de tipo .obj) y la escala con respecto a las coordenadas reales.

El alta de la información sobre las aeronaves y modelos solamente será necesaria una vez en el sistema, ya que se utilizarán a lo largo de todas las planificaciones.

3.1.2. Planificación

El servicio web dispone de una página en la que se planifica la misión en un único y sencillo paso. La creación de una misión se puede hacer desde cero, o también se puede crear una misión desde una plantilla de otra anterior. Todas las misiones son guardadas como plantillas para agilizar el proceso de creación de aquellas misiones que se repitan con frecuencia.

Para crear una misión desde cero se deberán especificar los siguientes parámetros:

- Nombre identificativo de la misión.
- Modelo sobre el que efectuar la planificación.
- Aeronaves disponibles.
- Puntos de interés y zona de salida y aterrizaje.
- (Opcional) Optimizaciones dependiendo de cada escenario.

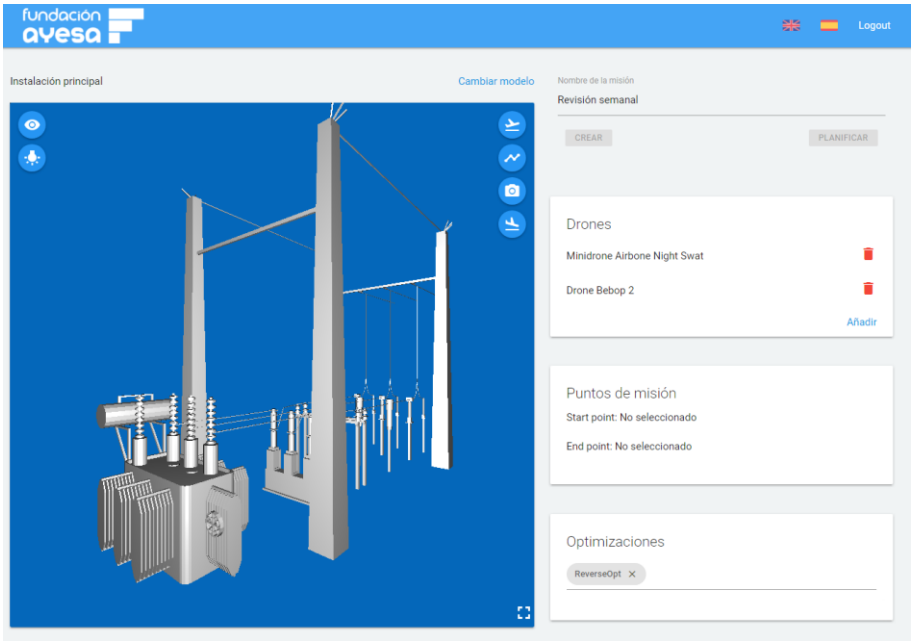

Figura 10: Ventana de planificación de misiones.

Las optimizaciones permiten acelerar el proceso de planificación para entornos controlados en los que los usuarios aseguren que se cumplen ciertas restricciones, dependiendo de la optimización que se desea aplicar.

El visor de misión avanzado presenta una funcionalidad extendida en dos pasos a la hora de añadir un punto de interés que facilita la adición de un punto en un espacio 3D. En primer lugar, se especifica el valor en el plano X-Z, y luego se especifica el valor para el eje Y.

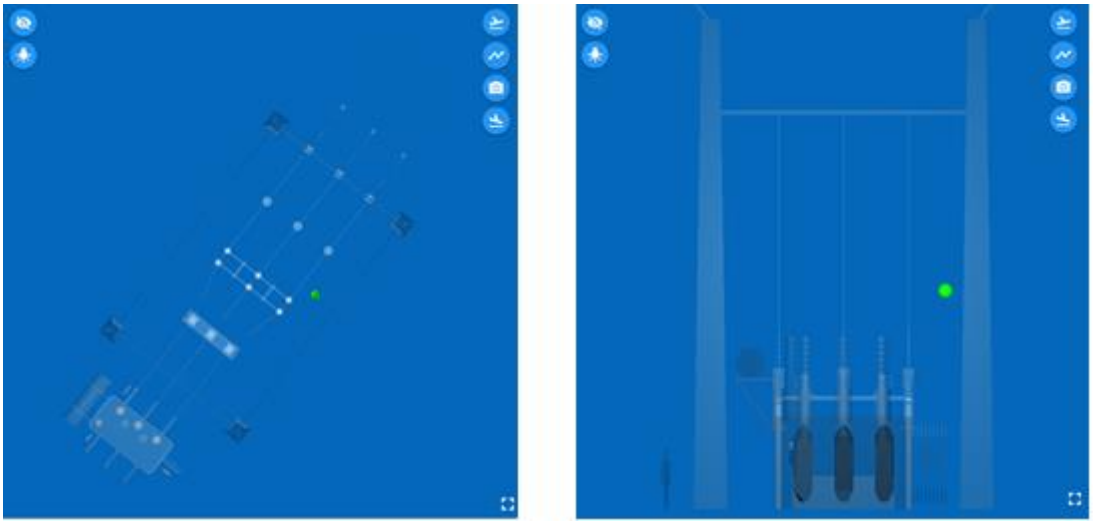

Figura 11: Elección de un punto de interés en el plano X-Z (izquierda) y en el eje Y (derecha).

Además, si el usuario quiere desplazar el punto para cambiarlo de posición, también puede hacerlo activando los controles de traslación al hacer clic sobre el punto a mover.

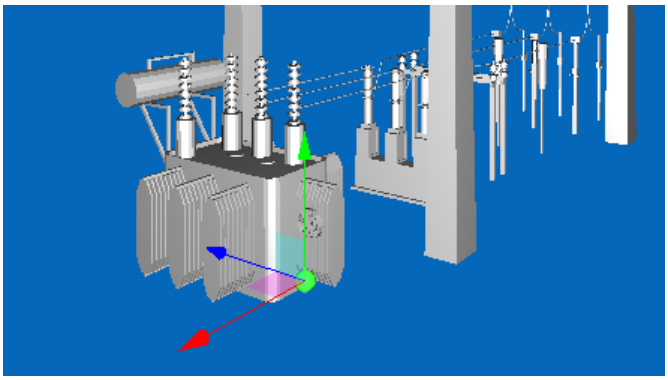

Figura 12: Controles de traslación para un punto de interés en el espacio.

Una vez especificados estos parámetros, se crea la misión en el sistema. El único paso que queda para planificar es ejecutar la acción del botón de "Planificar" y el sistema devolverá la planificación. En el apartado 4 Resultados se pueden observar los resultados de la planificación y su visualización en el visor de misión avanzado. El siguiente paso sería el de ejecutar la misión en un conjunto de vehículos, pasándole a través de una API local a cada UAV los comandos que deben seguir.

3.2. VOXELIZADO

La planificación de rutas en interiores requiere que el modelo 3D inicial sea procesado y almacenado como una matriz de posiciones ocupadas o libres, por lo que se ha procedido a la implementación de un algoritmo de voxelizado.

Se han escogido como base de partida los estudios publicados por Akenine-Möller [1], donde se desarrolla un algoritmo de voxelizado por CPU exacto superficial, debido a que en el proceso de planificación debemos tener un voxelizado lo más exacto posible y, al ser procesado en la CPU, puede ser portado a otros sistemas de manera más sencilla.

La implementación de este método se basa en el teorema de separación de ejes para realizar pruebas de intersección de triángulos-cubos. En las 2 dimensiones, este teorema indica que dos polígonos no intersecan si existe un eje (o recta) que sea capaz de separar los polígonos.

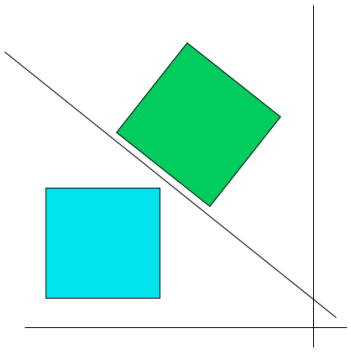

Figura 13: Teorema de separación de ejes.

Este teorema puede ser llevado a 3 dimensiones y poliedros tal como hizo Akenine-Möller [1] en su estudio de intersección de triángulos con cubos.

3.2.1. Implementación del voxelizado

Suponiendo un modelo 3D estándar, es decir, con superficies formadas por uniones de triángulos, y una malla voxel, se usaría el teorema de separación de ejes para comprobar qué voxels intersecan con la superficie del modelo.

Dicho de otro modo, si tenemos un modelo 3D y una malla voxel superpuesta al modelo, podemos ir comprobando si cada voxel interseca con la superficie del modelo 3D. Aquellos voxeles que no intersecan con la superficie del modelo quedan vacíos y aquellos voxeles que sí intersecan quedan rellenos. Tras aplicar esto a todos los voxeles de la matriz, obtenemos un voxelizado superficial del modelo 3D.

Debido a que se puede disponer de UAVs de tamaños diferentes hay que diferenciar caminos en los cuales unas aeronaves puedan pasar y otras no. Para que el algoritmo de planificación pueda tener esto en cuenta es necesario realizar diferentes procesos de voxelizado. Por cada tamaño de UAV disponible para planificar será necesario realizar un voxelizado, estableciendo como tamaño del voxel el tamaño de la aeronave, escalado acorde al modelo 3D. Gracias a estos voxelizados el algoritmo de planificación puede calcular rutas más específicas para vehículos más pequeños dentro de un conjunto de UAVs que tienen diferentes dimensiones, obteniendo rutas óptimas para cada caso.

El voxelizado obtenido en este proceso es un voxelizado superficial, pero es suficiente para las planificaciones de ruta. Aunque existan huecos dentro del voxelizado, se encontrarán encerrados por el voxelizado superficial y el algoritmo de planificación de rutas no será capaz de llegar a esos huecos vacíos. El algoritmo ignora las zonas inalcanzables, por lo que se acelera la voxelización y la planificación haciendo uso de este tipo de voxelizado.

3.3. PLANIFICADOR

Los algoritmos del planificador inteligente son los encargados de asignar las tareas a realizar por cada vehículo, así como de calcular el camino que deben recorrer para ello. El objetivo siempre es minimizar la distancia total recorrida por los UAVs, consiguiendo así la solución más eficiente

posible. Esta optimización se realiza de forma que se satisfagan los requisitos y restricciones definidos por el operario.

El planificador inteligente, en un primer nivel, se encarga de decidir qué vehículo de la flota debe realizar cada acción, así como el orden en el que debe visitarse cada uno de los puntos de interés. Se busca que la solución obtenida consiga realizar la misión definida de forma óptima desde el punto de vista de la eficiencia, teniendo en cuenta para ello los requisitos y restricciones definidas por el usuario.

El orden en el que se deben visitar un conjunto de puntos en una ruta para que la solución sea óptima se formuló por primera vez en 1930 y se conoce como el problema del viajero (TSP, *Travelling Salesman Problem*) [6]. En su formulación original se plantea, dado un conjunto de ciudades y las distancias entre ellas, cuál sería la forma óptima de visitar todas las ciudades una sola vez y de volver a la ciudad inicial, recorriendo para ello la menor distancia posible. Para resolverlo, el problema del viajero puede plantearse como un problema de optimización lineal entera (MILP) formulado de la siguiente forma:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$sujeto\ a:$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad i,j = 1 \dots n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i,j = 1 \dots n$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad i,j = 2, \dots n \quad i \neq j$$

$$0 \leq x_{ij} \leq 1 \quad i,j = 1, \dots n \quad i \neq j$$

$$u_i \geq 0\ y\ entero \quad i = 2, \dots n$$

Donde x_{ij}es la variable que refleja si la solución óptima contiene el tramo que va de la ciudad *i* a la *j*, c_{ij} es el coste de realizar este tramo, *n* el número total de ciudades y *u* es una variable asociada al orden en el que se visita cada una, siendo necesaria la tercera restricción para obligar a que una sola ruta cubra todas las ciudades.

En la herramienta desarrollada en este trabajo se contempla la realización de misiones de forma cooperativa por una flota de UAVs, por lo que se ha formulado un planteamiento más complejo que pretende abarcar el mayor número de situaciones posible a la hora de planificar una inspección cooperativa en un entorno industrial, ofreciendo la solución óptima.

La herramienta es capaz de resolver el problema definido por el usuario contemplando los siguientes aspectos:

- **Tipo de inspección a realizar**: En el mantenimiento de un entorno industrial puede ser necesario realizar una tarea distinta o recoger datos diferentes en cada punto a inspeccionar. Por ello, el algoritmo permite al usuario definir el tipo de misión que se desea realizar en cada punto de inspección.
- **Equipamiento y características de cada UAV**: El planificador contempla la posibilidad de que todas las aeronaves no sean iguales, es decir, que tengan diferente equipamiento (sensores) o que difieran en tamaño, peso, autonomía, etc. La herramienta presentada en este

trabajo es capaz de planificar la inspección asignando a cada UAV aquellas misiones que puedan realizar de forma satisfactoria teniendo en cuenta sus características y los sensores que tengan embarcados.

- **Autonomía de los UAV**: Puede darse el caso en el que se deseen realizar demasiadas inspecciones para la flota de UAVs disponible, dada su autonomía. Por ello, uno de los parámetros que tiene en cuenta el planificador es la autonomía de cada aeronave. Así, en casos en los que no se puedan realizar todas las misiones que el operario desee, el planificador inteligente devuelve como solución la planificación de las misiones que sería posible realizar y se notifica al operario qué puntos han quedado sin inspeccionar.
- **Puntos de recarga**: En la línea de la funcionalidad anterior, se añade al planificador la opción de que existan puntos en los que los UAVs puedan recargar las baterías. De esta forma, la herramienta tiene en cuenta que, si no hay suficiente autonomía para completar la misión, la solución óptima puede pasar por que alguna aeronave pare en algún punto de recarga para después continuar con las tareas asignadas.
- **Misiones parciales**: El planificador contempla que una misión necesite ser realizada por más de una aeronave para que la solución sea óptima. Esto puede deberse bien a falta de autonomía por parte de un UAV para completar la misión asignada o bien a que, por motivos de equipamiento, no se puedan realizar todas las acciones previstas en el punto marcado por el operario.
- **Restricciones de tiempo**: La herramienta contempla la definición de restricciones temporales para la realización de algunas misiones, por ejemplo, en el caso en el que alguna acción deba ser realizada en un determinado intervalo de tiempo.
- **Definición de zonas de riesgo e incertidumbre**: El algoritmo tiene en cuenta la posible existencia de zonas de catalogadas como de riesgo y de zonas de incertidumbre, en las que cabe la posibilidad de que el modelo no esté actualizado.

Para resolver el problema de optimización que decide el orden en el que deben realizarse las misiones, así como el UAV que debe realizar cada una de ellas, es necesario conocer el coste que supone realizar cada uno de los tramos que conectan los distintos puntos de interés. Sin embargo, la obtención de este coste no es trivial, ya que depende del camino escogido para unir ambos puntos. De esta forma, para conseguir que la solución del algoritmo de planificación completo sea óptima, es necesario que estos costes reflejen el correspondiente al camino más corto posible.

Se ha implementado un algoritmo de búsqueda de camino óptimo (ver sección 1.2), adaptado para realizar una búsqueda en un modelo voxelizado del entorno de trabajo (ver sección 3.2).

El algoritmo implementado siempre busca la solución óptima, y por lo tanto el tiempo de computación necesario aumenta considerablemente al aumentar el espacio de búsqueda. A pesar de que el algoritmo ha sido considerablemente optimizado para su aplicación al caso presentado en este artículo, cabe destacar que el paso de 2 a 3 dimensiones aumenta de forma considerable el espacio en el que buscar la solución, y por lo tanto también puede producirse un aumento en el tiempo de cálculo.

Por ello, además de la optimización del algoritmo para que el tiempo de cálculo sea lo menor posible, se han implementado una serie de aproximaciones para estimar los costes de los distintos tramos mediante búsquedas en el plano, en lugar de directamente en el espacio, acelerando aún más el tiempo de cálculo. Se reserva la opción por parte del usuario de descartar esta aproximación al realizar el cálculo, para aquellas situaciones extraordinarias en la que la aproximación afecte a la calidad de la solución. La Tabla 2 ilustra cómo esta aproximación reduce el tiempo de cálculo, especialmente a medida que aumenta el número de puntos de interés. En ella se muestra el tiempo empleado en realizar

la planificación completa, para una situación en la que se desean visitar 5 puntos de interés, y otra en la que son 10 los puntos de interés a visitar, obteniéndose en todos los casos idénticas soluciones con aproximación y sin ella.

	APROXIMACIÓN 2D	CÁLCULO 3D
5 Puntos de interés	2.2 segundos	11.97 segundos
10 Puntos de interés	3.2 segundos	37.47 segundos

Tabla 2: Comparación de tiempos de cálculo con y sin aproximación.

4. RESULTADOS

Se han realizado pruebas con diferentes modelos y para un número de puntos variable. En los casos mostrados a continuación, se han establecido 10 puntos de interés, variando la distancia entre ellos, la autonomía de los UAVs y las características de las instalaciones. Estas pruebas han probado los siguientes escenarios:

- **Uno de los UAVs con autonomía superior al camino que une los 10 puntos de interés:** Una única aeronave realiza la misión completa.
- **Ninguno de los UAVs tiene autonomía suficiente para realizar el camino que une los 10 puntos de interés:** Se seleccionan diferentes vehículos (el menor número posible de ellos) para visitar todos los puntos.
- **Entre todos los UAVs no suman la autonomía suficiente para cubrir todos los puntos en ninguna combinación de caminos y subcaminos:** Algunos puntos de interés no son visitados (el menor número de puntos posibles).

En la Figura 14 se observa un caso más simple, que el planificador ha resuelto con un UAV único, habiendo dos vehículos disponibles.

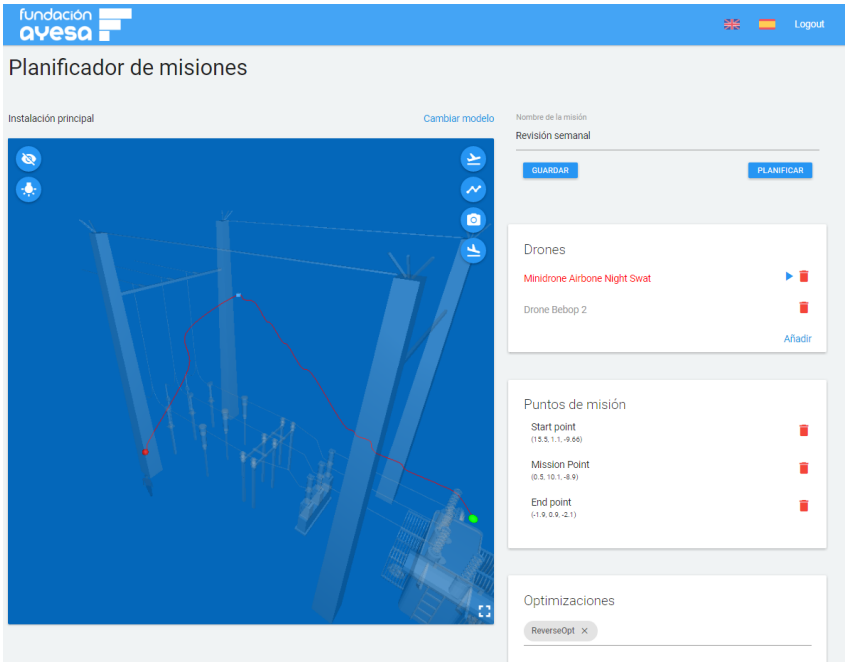

Figura 14: Solución arrojada por el planificador empleando un solo UAV, representada por la línea roja.

En la Figura 15 se muestra un caso similar al anterior, en el que el planificador ha tenido que hacer una planificación con ambos UAVs, ya que la autonomía de ninguno de ellos es suficiente para abarcar todos los puntos de interés de la misión.

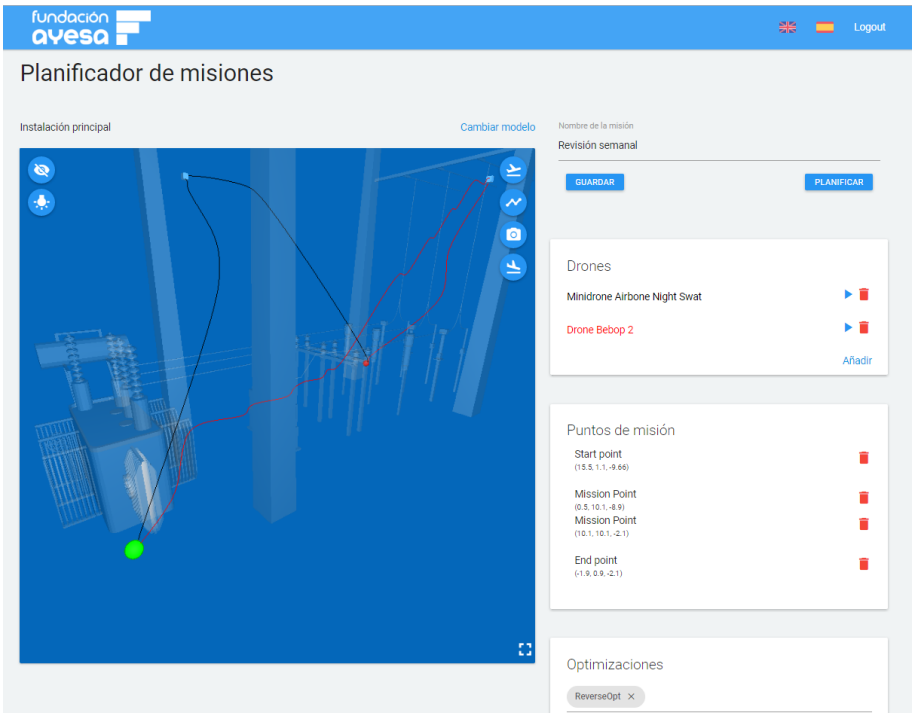

Figura 15: Solución arrojada por el planificador empleando dos UAVs. Las líneas de color negro y rojo indican las trayectorias propuestas para cada UAV.

El tiempo de ejecución varía en función del número de puntos de interés configurados para la misión, así como depende también de la distancia entre los mismos. La computación paralela permitirá acelerar el proceso y reducir la escalabilidad a orden lineal con respecto al número de puntos de interés configurados.

También se ha observado que el tiempo de cálculo varía notablemente dependiendo de las instalaciones. Instalaciones menos diáfanas mejoran los tiempos de búsqueda dado que se encuentran los caminos óptimos de manera más rápida al haber menos bifurcaciones.

Se está estudiando la implementación de más optimizaciones, además de las incluidas, que permitan acelerar el proceso en entornos controlados. Si se cumplen ciertas condiciones, se puede acelerar la ejecución de los algoritmos ahorrando tiempo y coste computacional aplicando heurísticas admisibles para la resolución de la solución óptima.

5. CONCLUSIONES

En este trabajo se ha presentado un planificador de misiones avanzado, basado en el tratamiento de modelos 3D y la búsqueda de soluciones óptimas para la planificación de inspecciones en entornos industriales que pueden ser muy diversos, empleando para ello una serie de algoritmos inteligentes desarrollados para esta aplicación.

La interfaz desarrollada permite la definición de las inspecciones a realizar en la planta industrial, así como las restricciones y requisitos asociadas a las mismas, por parte del operario, de una forma muy sencilla e intuitiva. De la misma manera, el operario puede detallar el número de vehículos disponibles para realizar las inspecciones, así como las características y sensores de cada uno de ellos,

para que el algoritmo calcule la forma más eficiente de realizar la misión. Por otro lado, la forma en la que se ha implementado el manejo de los modelos 3D permite facilitar aún más la definición de las misiones por parte del operario.

Una ventaja fundamental de la herramienta presentada es la gran flexibilidad que permiten los algoritmos implementados. La forma en la que se han desarrollado permite encontrar la solución óptima en una gran diversidad de casos, independientemente del número de UAVs, las características de cada uno de ellos, el número de acciones a realizar y la naturaleza de las mismas, las posibles restricciones temporales definidas por el usuario, etc.

El tiempo de cálculo necesario para realizar la planificación depende principalmente del número de puntos de interés que conforman la misión, y de las dimensiones y complejidad del entorno. Debido a que los entornos industriales pueden alcanzar gran complejidad, se han optimizado los algoritmos de planificación para reducir los tiempos de cálculo a unos límites satisfactorios, sin perder por ello calidad en la solución buscada.

En cuanto a las posibles líneas de trabajo futuro, se plantea la posibilidad de aprovechar la potencia de la herramienta presentada, así como la flexibilidad de los algoritmos, para su aplicación a flotas de robots móviles de distintas tipologías, así como su ampliación a otros tipos de aplicaciones complejas en las que sea necesario realizar una planificación de alto nivel.

6. AGRADECIMIENTOS

Proyecto SENSE2DRONE cofinanciado, en el programa RETOS-COLABORACIÓN, por el Ministerio de Economía, Industria y Competitividad y la Unión Europea, a través del Fondo Europeo de Desarrollo Regional.

7. BIBLIOGRAFÍA

[1] T. Akenine-Möller, «Fast 30 Triangle-Box Overlap Testing,» Department of Computer Engineering, Chalmers University of Technology, 2001.

[2] E. Eisemann y X. Décoret, «Single-pass GPU solid voxelization for real-time applications.,» *GI '08 Proceedings of Graphics Interface ,* 2008.

[3] A. Patel, 2009. [En línea]. Available: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[4] M. A. Padilla Castañeda, J. Savage, A. Hernández y F. Arámbula Cosío , «Local Autonomous Robot Navigation using Potential Fields,» 2008.

[5] J. Nasir, F. Islam, . U. Malik, Y. Ayaz, . O. Hasan, . M. Khan y . M. S. Muhammad, «RRT*-SMART: A Rapid Convergence Implementation of RRT*,» 2013.

[6] K. L. Hoffman, M. Padberg y G. Rinaldi, «Traveling salesman problem».

[7] C. Tripiana, «Master thesis: GPU Voxelization.,» 2009.