FlightCommander - Planificador de vuelos

Gonzalo Martínez Martínez
Universidad Politécnica de Madrid
Modelos de Razonamiento
Madrid, España

Àlex García Montané
Universidad Politécnica de Madrid
Modelos de Razonamiento
Madrid, España

Martín Molina González
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
Madrid, España

***Abstract*—Este proyecto propone un planificador de vuelos para drones haciendo hincapié en la capacidad de razonamiento de los sistemas autónomos. Para la implementación de este proyecto se ha hecho uso de algoritmos como A* y D* los cuales no solo implican la búsqueda eficiente de caminos, sino también la toma de decisiones basada en un razonamiento lógico para así, poder enfrentarse a obstáculos y calcular múltiples rutas. Para ello se debe analizar y evaluar las distintas opciones, utilizando la información proporcionada por el mapeo del entorno realizado a través de una imagen en blanco y negro de nuestra escena donde no se tienen en cuenta los objetos, tan solo las paredes. Durante el vuelo, el dron debe adaptar su ruta en función de la complejidad del entorno y la presencia de obstáculos imprevistos, mostrando una capacidad de toma de decisiones autónoma dependiendo del factor de riesgo que el usuario permita al dron. Además, el dron durante todo el recorrido priorizará la seguridad del mismo por lo que siempre intentará tener batería suficiente para realizar el trayecto.**

***Index Terms*—Planificador de vuelos, razonamiento, A*, D*, analizar, complejidad.**

I. Introducción

Este documento presenta una visión integral de nuestro proyecto centrado en la planificación de vuelos de drones en un entorno simulado utilizando CoppeliaSim. El objetivo principal es guiar a un dron a través de un entorno doméstico, enfrentándose a obstáculos en el camino. El entorno de simulación está equipado con varios caminos que llevan a una meta definida, y nuestro proyecto utiliza los algoritmos A* y D* para determinar la ruta óptima.

El entorno simulado refleja un espacio doméstico, completo con obstáculos que el dron debe sortear, presentando desafíos del mundo real. Para mejorar las capacidades de toma de decisiones del dron, este cuenta con un grafo extraído a partir de la imagen de nuestra escena sin obstáculos, teniendo en cuenta tan solo las paredes. Este proceso de mapeo es necesario para que el dron pueda planificar eficientemente su ruta de vuelo al proporcionar una representación detallada del entorno.

La utilización de algoritmos como D* destaca nuestro compromiso de aprovechar técnicas avanzadas de búsqueda de caminos. Este algoritmo desempeña un papel crucial al ser capaz de adaptar la ruta seguida por el dron, teniendo en cuenta factores como la evasión de obstáculos o la necesidad de ir a la estación de carga más cercana.

A lo largo de este documento, exploramos los detalles de nuestro proyecto, examinando la configuración de la simulación, las implementaciones algorítmicas y la integración del mapeo. Los resultados y las percepciones obtenidas de este proyecto contribuyen al campo de la navegación de drones y sistemas autónomos, demostrando la aplicación práctica de algoritmos de búsqueda de caminos en escenarios del mundo real.

II. Estado del Arte

La planificación de vuelos para drones ha experimentado notables avances, respaldados por el crecimiento acelerado de la tecnología de drones y sus aplicaciones en diversas industrias. La búsqueda de rutas eficientes y seguras se ha convertido en un área de investigación clave. A continuación, se presenta un panorama general del estado del arte en la planificación de vuelos para drones, destacando ejemplos significativos:

A. Algoritmos de Búsqueda de Rutas

Los algoritmos de búsqueda de rutas son utilizados para encontrar la ruta más corta o más eficiente entre dos puntos. Estos algoritmos se basan en la teoría de grafos y pueden ser utilizados para planificar rutas de drones. Ejemplos de algoritmos de búsqueda de rutas incluyen el algoritmo de Dijkstra y el algoritmo A*. Por ejemplo, Amazon Prime Air utiliza algoritmos de búsqueda de rutas para planificar la entrega de paquetes a los clientes.

B. Optimización de Trayectorias

La optimización de trayectorias se refiere a la planificación de rutas que minimizan el tiempo de vuelo y el consumo de energía. Los algoritmos de optimización de trayectorias pueden ser utilizados para planificar rutas de drones en entornos urbanos y rurales. Ejemplos de algoritmos de optimización de trayectorias incluyen el algoritmo de optimización de trayectorias basado en el método de los campos potenciales y el algoritmo de optimización de trayectorias basado en el método de la programación dinámica. La empresa PrecisionHawk utiliza algoritmos de optimización de trayectorias para planificar rutas de drones que permiten la recolección de datos precisos sobre los cultivos.

C. Integración de Sistemas de Evitación de Obstáculos

La integración de sistemas de evitación de obstáculos se refiere a la planificación de rutas que evitan obstáculos en tiempo real. Los sistemas de evitación de obstáculos pueden ser

utilizados para planificar rutas de drones en entornos urbanos y rurales. Ejemplos de sistemas de evitación de obstáculos incluyen el sistema de evitación de obstáculos basado en la visión y el sistema de evitación de obstáculos basado en el LiDAR. Actualmente, la empresa DJI utiliza sistemas de evitación de obstáculos basados en la visión para planificar rutas de drones que permiten la inspección de infraestructuras como puentes y torres de energía.

D. Aplicaciones Específicas

Las aplicaciones específicas se refieren a la planificación de rutas de drones para aplicaciones específicas, como la agricultura de precisión, la inspección de infraestructuras y la entrega de paquetes. Ejemplos de aplicaciones específicas incluyen la planificación de rutas de drones para la inspección de líneas eléctricas y la planificación de rutas de drones para la entrega de paquetes. Por ejemplo, la empresa Aerones utiliza drones para la limpieza de turbinas eólicas, lo que permite la realización de tareas de mantenimiento de manera más segura y eficiente.

E. Simulación y Validación

La simulación y validación se refiere a la planificación de rutas de drones utilizando simulaciones y validaciones en tiempo real. La simulación y validación pueden ser utilizadas para planificar rutas de drones en entornos urbanos y rurales. Ejemplos de simulación y validación incluyen la simulación y validación de rutas de drones para la entrega de paquetes y la simulación y validación de rutas de drones para la inspección de infraestructuras.

III. Metodología

La metodología empleada en este proyecto se fundamenta en la integración de algoritmos avanzados y la simulación detallada en CoppeliaSim para el desarrollo y evaluación del planificador de vuelos para drones. A continuación, se presenta una descripción detallada de los pasos seguidos en la metodología:

A. Simulación en CoppeliaSim

La simulación en CoppeliaSim proporciona un entorno virtual meticulosamente diseñado que recrea con realismo un entorno doméstico con obstáculos. Este entorno no solo imita situaciones del mundo real, sino que también facilita la experimentación y validación de los algoritmos de planificación de vuelo. Al diseñar la escena, se consideraron varios factores, incluyendo el número de caminos disponibles para llegar a la meta, la distribución de obstáculos en cada ruta, y la ubicación de puntos de carga.

El objetivo fue crear un escenario diverso que presentara al dron diversas posibilidades de rutas, cada una con su propio conjunto de desafíos. Estos desafíos se gestionan a través de factores como el número de obstáculos en el camino y la ubicación de puntos de carga, que influyen en la toma de decisiones del dron. Este planteamiento permite que el dron, basándose en un factor de riesgo predeterminado, seleccione la ruta más apropiada, demostrando así su capacidad de adaptación y toma de decisiones autónoma en tiempo real.

Además, con el objetivo de añadir un nivel adicional de realismo y evitar que el dron aprenda a seguir siempre la misma ruta, decidimos diseñar múltiples escenarios. En estos escenarios, se mantendrán más o menos obstáculos de los que hay creados en el mapa, simulando variaciones en el entorno. Por ejemplo, en días lluviosos, se introdujeron más obstáculos y más presencia humana en la casa, creando un entorno más complejo y desafiante. Por otro lado, en días laborables, la disposición del entorno se mantuvo más organizada, con menos obstáculos y una menor presencia de personas.

Más detalladamente, en nuestro problema creamos 3 posibles caminos, cada uno diseñado para representar diferentes niveles de complejidad. El camino corto, dispone de 16 obstáculos y 1 punto de carga, además este camino recorre 442 nodos del grafo. El camino medio presenta una menor complejidad con 13 obstáculos y 2 puntos de carga estratégicamente ubicados. El camino medio recorre 466 nodos. Por último, el camino largo, el más simple, cuenta con una cantidad aún menor de obstáculos, específicamente 10 obstáculos, y 3 puntos de carga distribuidos estratégicamente a lo largo de la ruta. El camino largo pasa por 567 nodos del grafo. Estas distintas opciones obligan al dron a adaptarse a diversas condiciones y desafíos presentes en el entorno, permitiendo una planificación de vuelo más versátil y realista.

Fig. 1. Representación de los tres posibles caminos, siendo el verde el trayecto más corto, el amarillo una opción de longitud intermedia y el rojo el recorrido más extenso.

B. Elección de Algoritmos

Se ha implementado un conjunto de algoritmos clave para la planificación de vuelos, centrándonos en A*, y D*. La elección de utilizar el algoritmo de A* en este proyecto se

fundamentó en la necesidad de determinar el camino más corto para la planificación de vuelos del dron. Se implementó con el objetivo de realizar una búsqueda exhaustiva y sistemática de todos los caminos posibles desde el punto de inicio hasta el destino.

Se eligió el algoritmo A* sobre Dijkstra, debido a su capacidad para optimizar la eficiencia de búsqueda al considerar heurísticas, lo cual es fundamental para la planificación de rutas en entornos complejos.

Sin embargo, con el objetivo de añadir un nivel adicional de robustez al sistema, se decidió incorporar D* en la implementación ya que ofrece una adaptabilidad excepcional en tiempo real, ajustándose dinámicamente a cambios en el entorno. Esta decisión se fundamentó en la idea de que, durante el vuelo del dron, podrían surgir obstáculos imprevistos en el camino y mientras que con A* el dron debería volver a calcular la ruta entera de nuevo, D* nos permite recalcular la ruta de manera eficiente en respuesta a estos obstáculos, creando un mecanismo mucho más flexible y adaptable al sistema.

C. Eliminación del Robot Terrestre

Durante la presentación se propuso la inserción de un robot terrestre para mapear nuestro entorno. Sin embargo, durante la creación de los algoritmos se observo que existía una forma más sencilla y eficaz de conseguir dicho mapa que es mediante una imagen bidimensional como bien se detalla en la subsección "Generate map.py".

IV. Implementación del código

A continuación, se detallará que archivos constituyen nuestro proyecto y explicaremos las principales características y funcionalidades del mismo.

A. Generate map.py

En esencia, el código genera una representación visual en forma de imagen bidimensional con dos canales donde cada píxel de la imagen actúa como un nodo en un grafo, y los canales de la imagen cumplen funciones específicas. Un canal, representado en negro, delinea las áreas ocupadas por paredes en el entorno, proporcionando una clara identificación visual de los límites físicos. Por otro lado, el segundo canal, en blanco, denota las áreas libres de obstáculos, creando un contraste que resalta los espacios accesibles. Esta estructura de dos canales refleja la topología del entorno en un formato fácilmente interpretable. Además, esta representación gráfica en 2D es esencial para la implementación de algoritmos de planificación de ruta, ya que facilita la identificación de nodos y aristas que permiten la navegación eficiente a través del espacio. Este enfoque visual proporciona una herramienta valiosa para comprender la disposición del entorno y optimizar la planificación de movimientos en entornos tridimensionales.

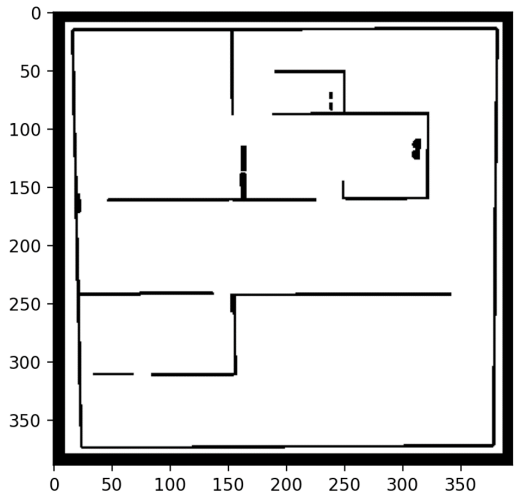

Fig. 2. Representación visual en forma de imagen bidimensional muestra un entorno con dos canales distintos

B. Generate path.py

Este código en Python constituye una herramienta integral para la generación de trayectorias y la planificación de rutas en entornos bidimensionales para robots terrestres. Su funcionalidad principal se centra en el procesamiento de imágenes, interpretando los píxeles negros como obstáculos y los píxeles blancos como áreas transitables. A través de la creación de un grafo que modela la conectividad espacial, se establecen nodos en las zonas transitables y se generan conexiones entre nodos adyacentes. Además, para simular el tamaño del dron, se consideran obstáculos todos aquellos nodos que estén a una distancia menor a 11 de otro obstáculo.

La versatilidad de este código radica en la variedad de algoritmos de búsqueda y planificación que ofrece. Desde búsqueda en profundidad y en anchura hasta Dijkstra, A* y algoritmos probabilísticos como el árbol de expansión rápida y el muestreo probabilístico de carreteras. La elección del algoritmo y, en algunos casos, de heurísticas como la distancia de Manhattan o euclidiana, permite adaptarse a diversas situaciones y requisitos específicos.

A medida que se ejecutan los algoritmos, el código proporciona información detallada sobre las rutas generadas, el número de nodos visitados y, en ciertos casos, la distancia estimada de la ruta. La capacidad de visualizar gráficamente tanto la ruta como los nodos visitados en la imagen original facilita la comprensión y evaluación de los resultados.

Aunque en dicho código aparecen una cantidad notable de algoritmos de búsqueda y planificación, al final se decidió usar A* con la heurística euclidiana, ya que es un algoritmo que siempre permite encontrar el mejor camino entre dos puntos y para hacerlo no tarda tanto tiempo como el resto de los algoritmos presentes en esta hoja de código.

C. Robotica.py

Este código en Python, llamado robotica.py, se enfoca en facilitar la comunicación entre una aplicación externa escrita

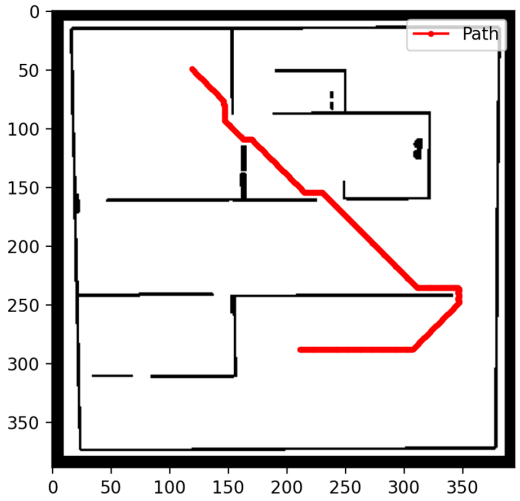

Fig. 3. Generación de la trayectoria en el mapa bidimensional generado

en Python y el simulador robótico CoppeliaSim.

La primera parte del código establece la conexión con CoppeliaSim a través de la API remota ZeroMQ. La clase Coppelia se encarga de manejar la comunicación y controlar el inicio y detención de la simulación. Además, ajusta parámetros para optimizar la simulación.

La segunda parte del código presenta la clase QuadCopter, que representa un dron en el entorno simulado de CoppeliaSim. Esta clase incluye funciones para obtener y establecer posiciones del cuadricóptero y su objetivo, así como para manipular objetos en el entorno, volviéndolos invisibles y no colisionables en función al escenario en el que nos encontremos (día lluvioso, día festivo...). Finalmente, también obtiene las distancias calculadas por el LiDAR y estima los ángulos en los que el LiDAR está detectando los objetos.

En resumen, el código proporciona una interfaz en Python para interactuar con un simulador robótico, enfocado para uso de drones. Facilita la gestión de la simulación, el control de posiciones y la manipulación de objetos en el entorno de CoppeliaSim.

D. Drone.py

Este código de Python constituye todos los métodos que se deben ejecutar cada vez que queramos realizar una simulación, es decir, consta de una etapa inicial donde se inicializan las variables, se llama a Robotica.py para conectarse con CoppeliaSim y se selecciona el camino inicial; y consta de una etapa reiterativa, la cual se repetirá hasta que el robot llegue a la meta o se quede sin batería. Esta parte reiterativa, como posteriormente se especificará en más detalle, consta de una parte de detección de obstáculos cercanos, una parte donde se determina de forma lógica si se debe seguir yendo hacia la meta o si se debe ir a cargar a la estación más cercana, y una última parte donde se reajusta la ruta de ser necesario. Finalmente, también se determina donde se debe mover el dron y se actualiza el estado de la batería.

Inicialización de variables: Primero de todo, se inician todas las variables que el dron necesita para determinar que camino debe seguir. Debido a esto, en esta primera sección, se define el número de obstáculos y de estaciones de carga en cada camino, se carga el grafo y los tres posibles caminos calculados en Generate path.py, y se selecciona de forma aleatoria si es festivo, si es fin de semana y si está lloviendo. En función a esto, se eliminan el 30% de todos los objetos de nuestra escena por cada una de las anteriores afirmaciones que no se cumplan, y un 3% por las que sí se cumplan. Es decir, si está lloviendo, es festivo y es fin de semana se seleccionarán de forma aleatoria y se eliminarán el 10 % de objetos de toda la escena y si no llueve, no es festivo y no es fin de semana, se eliminarán un 91 % de los obstáculos de toda la escena. Al no solo seleccionarse de forma aleatoria cuántos objetos se deben eliminar, sino que también qué objetos se eliminarán de todos los que se encuentran en la escena, se crean cientos de escenarios diferentes en los cuales el dron no sabe qué objetos están en la escena ni mucho menos cuántos obstáculos tiene cada camino, teniendo que estimarlos en función al riesgo que quiera asumir.

Otro apartado clave es que tenemos dos sistemas de coordenadas distintos, uno para CoppeliaSim y otro para nuestro grafo. Debido a esto, tendremos que ir adaptando las posiciones observadas en CoppeliaSim para usarlas en nuestros grafos y lo mismo a la inversa. Esto lo podemos hacer gracias a que conocemos las coordenadas exactas en ambos sistemas, de la posición inicial de nuestro dron y de la meta.

Selección del camino inicial: Una vez tenemos todas las variables inicializadas, se le pide al usuario qué riesgo quiere asumir (alto, medio o bajo) y en función a este, al número de objetos eliminados, a las estaciones de carga presentes en cada camino, y a la longitud de cada camino se determina qué camino seguir inicialmente. En más detalle, para un riesgo alto se asume que no habrán muchos objetos en el camino seleccionado, que no se tardará demasiado en rodearlos y que no hará falta cargar el dron demasiadas veces. Así pues, se buscaría seleccionar un camino más corto, aunque tenga más obstáculos y menos estaciones de carga. Sin embargo, para un riesgo bajo se asume que habrá muchos objetos, que se tardará en rodearlos y que se tendrá que cargar el dron varias veces, así que se buscaría seleccionar un camino más largo pero seguro.

Determinar si seguir hacia la meta o ir hasta la estación de carga más cercana: En esta sección del código se busca determinar si ir hacia la estación de carga más cercana o si seguir yendo hacia la meta. Para esto de nuevo se usa el riesgo, el número de objetos eliminados y la longitud del camino seleccionado que queda hasta llegar a la meta. Usando esto, se estima con cuánta batería llegaría el dron a la meta, teniendo en cuenta que como la batería se va reduciendo de forma aleatoria entre 0 y 1, un riesgo alto asume que este

número será más cercano a 0 y un riesgo bajo que será más cercano a 1. Una vez está estimado con cuánta batería se llegaría a la meta, si está batería es más alta que un número (el cual es bajo si es el riesgo es alto, y alto si el riesgo es bajo) se seguirá yendo hacia la meta y sino se irá hacia la estación de carga más cercana.

Además, cuando se determina cambiar la meta para ir a la estación de carga más cercana, tras cada paso se mira si ya está en dicha estación y si lo está, desciende hasta tocar el suelo, espera 3 segundos para simular que se está cargando y vuelve a subir a su altura habitual con la batería al 100% y el objetivo siendo de nuevo la meta.

Actualizar el punto intermedio del camino: Cabe recalcar, que como a veces es necesario recalcular la ruta (si se decide ir a cargar el dron o se detectan obstáculos que no están en el grafo), es posible que si se recalcula al principio del camino intermedio o lento, se decida seguir el camino rápido, ya que al final D* busca siempre el camino óptimo en función a la distancia entre el punto inicial y la meta. Sin embargo, para evitar que esto ocurra, para el camino intermedio y el lento se va primero a un punto intermedio que está en el camino correspondiente y de allí sigue yendo hacia la meta.

Detectar obstáculos: En esta parte se utiliza el LiDAR para determinar donde hay objetos. Básicamente se convierten las coordenadas polares devueltas por el LiDAR (ángulo y distancia) en coordenadas cartesianas (x,y) y se combinan con la posición y orientación del dron para tener en cuenta donde se encuentra el dron. Además, se filtran las medidas del LiDAR para tan solo tener en cuenta los objetos cercanos al dron y finalmente se eliminan los nodos del grafo que corresponden a dichos objetos y sus vecinos de una distancia menor a 8 para el camino intermedio y 11 para el camino largo y corto, para simular de nuevo el tamaño del dron. No se usa 11 para el camino intermedio, aunque funciona mejor, debido a que como sus puertas son más estrechas, si se usa 11, D* piensa que el dron no caberá.

Recalcular la ruta: Posteriormente, se usa el algoritmo D*, que como bien se ha dicho con anterioridad es útil para recalcular de forma eficiente una ruta anterior, es decir sin tener que recalcularla desde 0. Esto se hace solo si o bien se ha determinado ir a una estación de carga cercana o bien se ha detectado un objeto que previamente no estaba en el grafo, es decir si se detecta una pared o un objeto ya detectado con anterioridad no hace falta recalcular la ruta.

Asignaciones finales: Finalmente, se mira si se ha llegado ya a la meta, y de ser así se devuelve el tiempo que se ha tardado en llegar a la meta y se termina la simulación. De no ser así, se actualiza la posición del dron dos píxeles más allá y se comprueba si nos vamos a quedar sin batería. De ser así, el dron aterriza para minimizar daños, se le comunica al usuario que el dron no ha llegado y termina la simulación. Y de no ser así se le resta a la batería un número determinado de forma aleatoria entre 0 y 1. Finalmente, si la simulación ha terminado se vuelven a hacer todos los objetos visibles y tangibles (hay que hacerlos tangibles desde CoppeliaSim).

PID: Para actualizar la posición del dron, desde Python, lo único que hacemos es actualizar la posición de un objeto llamado target que le comunica al dron donde tiene que ir como si se tratara de un control remoto. Sin embargo, para ir hasta allí, lo que el dron usa es el control proporcional, integral y derivativo (PID), el cual está ejecutado dentro de la clase QuadCopter en CoppeliaSim. PID es un proceso que, mediante tres parámetros, corrige la distancia entre una variable deseada y la variable actual, en nuestro caso una posición. Este mecanismo es muy usado en el mundo de los drones ya que determina de forma idónea como debe ir el dron de una posición a otra sin perder el control. Esta función se ejecuta en bucle de forma automática.

V. Experimentación

Finalmente, se llevaron a cabo una serie de experimentos con el objetivo de evaluar la robustez y la adaptabilidad del sistema en diferentes situaciones y entornos simulados. Las pruebas se enfocaron en aspectos como la toma de decisiones del cuadricóptero basada en el factor de riesgo y la variabilidad del entorno y en comparar las veces que el dron se debe cargar, el tiempo y la probabilidad de llegar a la meta antes de quedarse sin batería.

A. Selección del camino inicial

Para evaluar la capacidad del sistema para tomar decisiones informadas, se realizaron pruebas variando el nivel de riesgo en el entorno simulado, y el número de objetos activos.

- **Número de objetos:** No lluvia, no festivo y no fin de semana:
  - **Riesgo:** high (h):
    En estas condiciones, tan solo hay un 9% de los objetos presentes en la escena. Debido a esto, para un riesgo alto con pocos obstáculos, se elige el camino rápido y arriesgado.
  - **Riesgo:** medium (m):
    Como tan solo hay un 9% de los objetos presentes en la escena, para un riesgo medio, se vuelve a elegir el camino rápido y arriesgado, ya que es un riesgo asumible debido a los pocos obstáculos presentes.
  - **Riesgo:** low (l):
    De nuevo, al haber tan solo un 9% de los objetos presentes, se elige el camino intermedio, ya que es un riesgo asumible por la baja cantidad de obstáculos en el mapa.
- **Número de objetos:** lluvia, no festivo y no fin de semana:
  Para el ejemplo, se consideró lluvia cierto y festivo y fin de semana falso, para da igual cuales sean ciertos y cuales no, ya que cada opción cierta añade el mismo porcentaje

de obstáculos independientemente de si se refiere a lluvia a festivo o a fin de semana.

  - **Riesgo:** high (h):
    En estas condiciones, un 36% de los objetos de la escena están activos. Debido a esto, para un riesgo alto y una cantidad equilibrada de obstáculos, se elige el camino rápido y arriesgado.
  - **Riesgo:** medium (m):
    Para un riesgo medio, al haber más objetos que en el caso anterior (antes había un 10%), se selecciona el camino intermedio.
  - **Riesgo:** low (l):
    Finalmente, para un riesgo bajo, se selecciona el camino lento, pero sin riesgo.

- **Número de objetos:** lluvia, festivo y no fin de semana:
  De nuevo, da igual que situaciones sean ciertas, siempre y cuando dos lo sean ya que todas añaden el mismo número de objetos.
  - **Riesgo:** high (h):
    En estas condiciones, un 63% de los objetos de la escena están activos. Debido a esto, para un riesgo alto y una cantidad equilibrada de obstáculos, se elige el camino rápido y arriesgado.
  - **Riesgo:** medium (m):
    Para un riesgo medio, de nuevo encontramos la situación de equilibrio, por lo que se vuelve a seleccionar el camino intermedio.
  - **Riesgo:** low (l):
    Finalmente, para un riesgo bajo, se selecciona el camino lento, pero sin riesgo.

- **Número de objetos:** lluvia, festivo y fin de semana:
  - **Riesgo:** high (h):
    En estas condiciones, un 90% de los objetos de la escena están activos. Debido a esto, para un riesgo alto y una cantidad alta de obstáculos, se considera muy arriesgado elegir el camino rápido, y por eso se selecciona el intermedio.
  - **Riesgo:** medium (m):
    Lo mismo pasa aquí ya que se considera muy arriesgado elegir el camino intermedio y por ello se selecciona el corto.
  - **Riesgo:** low (l):
    Finalmente, como nos podíamos imaginar, para un riesgo bajo, se selecciona el camino lento, pero sin riesgo.

B. Número de veces que el dron se carga, tiempo medio y probabilidad de llegar a la meta

Finalmente, se analizó la relación entre el riesgo asumido y el número de objetos activos con el número de veces que un dron necesita cargarse en un recorrido, las veces que consigue llegar a la meta y el tiempo medio en el que lo consigue. Para evaluar esto, se ejecutó cada experimento 5 veces para tratar de ver el máximo de casos posibles.

- **Número de objetos:** No lluvia, no festivo y no fin de semana:
  - **Riesgo:** high (h):
    Como bien sabemos, en estas condiciones tan solo hay un 9% de los objetos activos y se ha elegido el camino rápido. Debido a esto, se logra llevar a la meta en tan solo 70.84 segundos de media. Además, a pesar de haber seleccionado el camino largo, como hay muy pocos objetos, no es necesario cargar el dron ni parece haber posibilidades de que este se quede sin batería.
  - **Riesgo:** medium (m):
    En este caso se seleccionó de nuevo el camino rápido. Debido a esto, si el dron no necesita cargarse, consigue un tiempo medio similar para llegar a la meta que en el caso anterior (70.73 -es un poco diferente debido a la aleatoriedad de obstáculos-). Sin embargo, al tratarse de un riesgo medio, algunas veces, dependiendo de los obstáculos activos y lo rápido que se pierde la batería, necesita cargarse. Esto ocurre un 20% de las veces y en ese caso tarda un tiempo medio de 86.29 segundos. Además, el dron no tiene posibilidad de quedarse sin batería, ya que antes de que esto ocurra decide cargarse.
  - **Riesgo:** low (l):
    En este caso se seleccionó el camino intermedio. Debido a esto, el dron necesita cargarse una vez. De media, llega a la meta en 84.15 segundos, es decir un poco más de lo que se tarda en el camino corto sin cargarse, pero algo menos que en el camino corto cuando el dron necesita cargarse. Esto ocurre ya que la estación de carga está mejor colocada en el camino medio que en el corto. La posibilidad de quedarse sin batería es del 0%.

- **Número de objetos:** lluvia, no festivo y no fin de semana:
  - **Riesgo:** high (h):
    En estas condiciones, un 36% de los objetos de la escena están activos y se ha elegido el camino rápido. Ahora, debido a que hay más objetos, el dron tarda algo más de media en llegar a la meta. Para ser concretos unos 73.4 segundos. Además, debido a este tiempo extra que se toma con estos objetos tiene un 20% de probabilidades de quedarse sin batería. Sin embargo, al usar un riesgo alto, nunca se va a cargar.
  - **Riesgo:** medium (m):
    En este caso se seleccionó el camino intermedio. Debido a que ahora hay más objetos, el dron tarda 85.93 segundos en llegar a la meta. Además, se carga una vez. Debido a que se ha cargado y a que tampoco hay tantos objetos no hay posibilidad de que no llegue a la meta.
  - **Riesgo:** low (l):
    Finalmente, para un riesgo bajo se seleccionó el

camino lento. Debido a esto, como el camino lento es bastante largo, el dron debe cargarse una o dos veces para llegar a la meta (solo un 20% de las veces se carga dos veces). Debido a esto tarda de media 100.55 segundos si se carga una vez y 163.47 si se carga dos. La diferencia entre cargarse una o dos veces es bastante grande debido a que tiene que dar bastante vuelta para llegar a la segunda estación de carga. Sin embargo, consigue no tener riesgo de quedarse sin batería.

- **Número de objetos:** lluvia, festivo y no fin de semana:
  - **Riesgo:** high (h):
    En estas condiciones, un 63% de los objetos están activos y se selecciona el camino rápido. Debido a esto, como ahora hay más obstáculos en la escena, el tiempo medio sube a 75.2 segundos. Además, ahora hay más probabilidades de que el dron se quede sin batería, siendo está del 40%. Sin embargo, debido a que asume mucho riesgo sigue sin irse a cargar ninguna vez.
  - **Riesgo:** medium (m):
    Para un riesgo medio, se selecciona de nuevo el camino intermedio. Debido a esto y a que hay más obstáculos, de nuevo se tarda de media un poco más de tiempo en llegar a la meta, más concretamente 86.4 segundos. Sin embargo, debido a lo bien colocada que está la estación de carga en este camino, el dron sigue cargándose tan solo una vez y llegando a la meta siempre.
  - **Riesgo:** low (l):
    Finalmente, para un riesgo bajo, se selecciona el camino lento. Debido a esto, el dron se sigue cargando entre 1 y 2 veces (ahora el 60% de las veces se carga dos veces) pero siempre llega a la meta. El tiempo medio de nuevo sube un poquito debido a que hay más objetos, siendo este de 105.61 segundos cuando se carga una vez y 167.84 segundos cuando se carga dos.
- **Número de objetos:** lluvia, festivo y fin de semana:
  - **Riesgo:** high (h):
    Finalmente, en estas condiciones, un 90% de los objetos de la escena están activos y se selecciona el camino intermedio. Debido a que se ha seleccionado un camino algo más lento, pero con menos objetos, el dron sigue asumiendo mucho riesgo y sigue sin cargarse. Sin embargo, llega a la meta el 80% de las veces. Además, consigue un tiempo de 80.2 segundos de media, el cual es menor que todos los anteriores tiempos en este camino, ya que no ha necesitado cargarse.
  - **Riesgo:** medium (m):
    En este caso, se decide seleccionar el camino largo. Debido a lo largo que es este camino y la cantidad de objetos que hay, el dron necesita cargarse dos veces, pero tiene un 100% de probabilidades de llegar a la meta. Sin embargo, tarda un poco más de tiempo en hacerlo debido a que hay más objetos, siendo su tiempo medio de 169.33 segundos.
  - **Riesgo:** low (l):
    Finalmente, para un riesgo bajo, se selecciona el camino largo. Debido de nuevo a lo largo que es y al número de obstáculos que hay, el dron se debe cargar entre 2 y 3 veces. Sin embargo, siempre llega a la meta. De nuevo, tarda un poco más en llegar a la meta debido a que hay más obstáculos. Más concretamente tarda 168.72 segundos cuando se carga dos veces y 187.55 segundos cuando se carga tres veces.

Es decir, el objetivo era usar la lógica para dar prioridad al tiempo para un riesgo alto y seguridad para un tiempo bajo, y parece que lo hemos conseguido debido a que el riesgo alto siempre consigue el menor tiempo para llegar a la meta y los riesgos medios y bajos tienen una seguridad del 100% de llegar a la meta.

Finalmente recalcar que en las referencias hemos añadido un link a un video de youtube donde se ve cómo funciona el dron y un link a un repositorio de GitHub con nuestro código.

VI. Conclusiones

En conclusión, el proyecto ha logrado desarrollar un sistema robótico basado en el entorno de simulación CoppeliaSim y su comunicación remota mediante la API ZeroMQ. La implementación de este sistema ha permitido la interacción efectiva entre un dron cuadricóptero y su entorno simulado, abriendo posibilidades para la experimentación y evaluación de diferentes escenarios.

La capacidad del sistema para iniciar y detener la simulación, así como controlar las posiciones del dron y su objetivo, ha demostrado ser robusta y eficiente. La funcionalidad de activar y desactivar objetos en el entorno, especialmente aquellos relacionados con obstáculos, ha ampliado las capacidades del sistema para simular situaciones realistas.

La integración de un sensor LiDAR en el dron ha agregado una dimensión crucial al proyecto, permitiendo la adquisición de datos ambientales que influyen directamente en la toma de decisiones del dron. La interpretación de estos datos, junto con la capacidad de activar o desactivar objetos en el entorno, ha demostrado ser vital para la navegación autónoma del dron.

Durante las pruebas experimentales, se observó que el sistema respondió de manera eficaz a diferentes condiciones, adaptándose a cambios en el entorno simulado de forma lógica. Se llevaron a cabo pruebas específicas para evaluar el comportamiento del dron en días laborables, festivos, y durante condiciones meteorológicas adversas como la lluvia. Estas pruebas proporcionaron insights valiosos sobre la capacidad del sistema para gestionar situaciones variadas y su resistencia a condiciones imprevistas.

Aunque los resultados son prometedores, es importante señalar que el sistema actual presenta ciertas limitaciones. La

precisión del modelo de física en la simulación y la representación realista de las condiciones meteorológicas podrían mejorarse en futuras iteraciones.

VII. Trabajo futuro

El proyecto actual sienta las bases para futuras investigaciones y mejoras. Para un trabajo futuro, se considera la posibilidad de implementar algoritmos de aprendizaje automático que permitan al dron adaptarse y aprender de manera autónoma en entornos dinámicos. La integración de técnicas de visión por computador podría mejorar la percepción del dron, permitiéndole tomar decisiones más informadas y precisas.

También, se podría explorar la posibilidad de crear el mapa en 3 dimensiones y usar un lidar en 3 dimensiones que es algo que existe en la vida real pero no en CoppeliaSim, y por ese motivo no hemos podido explorar esta posibilidad. Esto ayudaría a que el robot también sea capaz de evitar obstáculos por arriba o por abajo y no solo por los lados.

Además, se podría explorar la aplicación de estrategias de planificación de trayectorias más avanzadas para optimizar la eficiencia y seguridad de los movimientos del dron. La inclusión de sensores adicionales, como cámaras estéreo o sensores de proximidad, podría proporcionar información adicional para una navegación más precisa y evitar colisiones de manera más efectiva.

En términos de experimentación, un área interesante para futuras pruebas podría ser la evaluación del rendimiento del dron en entornos simulados más complejos y realistas, lo que permitiría una validación más completa de su capacidad para enfrentar situaciones del mundo real. Este enfoque proporcionaría una comprensión más profunda de las limitaciones y fortalezas del sistema en escenarios más desafiantes.

References

[1] Rocío Fernández Sánchez, https://www.eagledron.es/los-10-usos-principales-de-los-drones/ , Ago 13, 2022.

[2] Anónimo, https://www.iberdrola.com/innovacion/drones-usos-tipos, 2022.

[3] Anónimo, https://easydrones.es/blog/aplicaciones-drones/ , 2023.

[4] Anónimo, https://www.pimealdia.org/es/15-aplicacions-dels-drons-a-la-nostra-societat/, May 29, 2016.

[5] Dibyendu Biswas, https://dibyendu-biswas.medium.com/d-d-lite-lpa-e7483779a7ca, Jun 27, 2021.

[6] Àlex García Montané, Gonzalo Martínez Martínez Biswas, https://github.com/gonmarmar5/FlightCommander, Jan 15, 2024.

[7] Àlex García Montané, Gonzalo Martínez Martínez Biswas, https://www.youtube.com/watch?v=4wjA8943gA8, Jan 15, 2024.