UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE INGENIERÍA ELECTRÓNICA

"Desarrollo e Implementación de un Sistema de Control Para la Planificación de Trayectorias de Vehículos Aéreos no Tripulados en Ambientes Interiores Controlados"

Trabajo de graduación previo a la obtención del título de:

INGENIERO ELECTRÓNICO

Autor:

ADRIÁN SAÚL REIBÁN GARNICA

Director:

Ing. ANDRÉS PATRICIO CABRERA FLOR (MSc)

CUENCA, ECUADOR

2021

Desarrollo e Implementación de un Sistema de Control Para la Planificación de Trayectorias de Vehículos Aéreos no Tripulados en Ambientes Interiores Controlados

RESUMEN

Este trabajo propone una alternativa a los sistemas de planificación de trayectorias para drones de uso interior que navegan en ambientes estáticos sin elementos de posicionamiento externo. Su enfoque está orientado al manejo de cuatro vehículos. El sistema planteado se fundamentó en técnicas del *Path Planning* discreto, y su diseño fue condicionado por restricciones de movimiento en los drones, obteniéndose una estrategia algorítmica y un software para la generación de rutas automáticas de vuelo. En la demostración de las capacidades del sistema, se utilizaron los Tello EDU (mini drones) y sus funciones de comando propietarias.

Palabras clave: *Path Planning*, Vehículo aéreo no tripulado, Restricciones de ruta, Voxelización, Generación de rutas, Sistema de comando y supervisión.

X 

Ing. Daniel Iturralde Piedra, Ph.D
Coordinador de Carrera

X 

Ing. Andrés Cabrera Flor, MSc
Director de Trabajo de Titulación

X 

Adrián Saúl Reibán Garnica
Autor

Development and Implementation of a Trajectory Planning Control System for Unmanned Aerial Vehicles in Controlled Indoor Environments

ABSTRACT

This work proposes an alternative to trajectory planning systems for indoor drones navigating in static environments without external positioning devices. The approach is geared towards the control of four aerial vehicles. The system was based on discrete path planning techniques, and its design was conditioned by the dynamic constraints of drones. This led to obtain an algorithmic strategy and software to generate automatic flight paths. System capabilities were tested using Tello EDU mini drones and their proprietary command functions.

Keywords: Path Planning, Unmanned aerial vehicle, Path constraints, Voxelization, Path Generation, Command and supervision system.

X

Eng. Daniel Iturralde Piedra, Ph.D
Faculty School Director

X

Eng. Andrés Cabrera Flor, MSc
Thesis Director

X

Adrián Saúl Reibán Garnica
Author

Translated by

Eng. Andrés Cabrera Flor, MSc

Desarrollo e Implementación de un Sistema de Control Para la Planificación de Trayectorias de Vehículos Aéreos no Tripulados en Ambientes Interiores Controlados

Adrián Saúl Reibán Garnica
Escuela de Ingeniería Electrónica
Universidad del Azuay
Cuenca, Ecuador
areiban@es.uazuay.edu.ec

***Resumen*—Este trabajo propone una alternativa a los sistemas de planificación de trayectorias para drones de uso interior que navegan en ambientes estáticos sin elementos de posicionamiento externo. Su enfoque está orientado al manejo de cuatro vehículos. El sistema planteado se fundamentó en técnicas del *Path Planning* discreto, y su diseño fue condicionado por restricciones de movimiento en los drones, obteniéndose una estrategia algorítmica y un software para la generación de rutas automáticas de vuelo. En la demostración de las capacidades del sistema, se utilizaron los Tello EDU (mini drones) y sus funciones de comando propietarias.**

Palabras clave*—*Path Planning, Vehículo aéreo no tripulado, Restricciones de ruta, Voxelización, Generación de rutas, Sistema de comando y supervisión.

I. INTRODUCCIÓN

La planificación de rutas y trayectorias para el movimiento de vehículos aéreos no tripulados (UAVs - *Unmanned Aerial Vehicles*) o drones en ambientes confinados, ha sido causa de interés a lo largo de los años debido al sin número de aplicaciones derivadas de ellas, tales como: exploración de entornos de riesgo; medición y adquisición de datos en edificios; movimiento coordinado enfocado a eventos artísticos, etc. Lo que ha impulsado el estudio, la aplicación y el desarrollo de diversas estrategias y recursos para el control de drones en ambientes interiores. Estos ambientes, al ser espacios cerrados, vuelven imprácticos a los sistemas de ubicación y posicionamiento GPS, por tanto, se requieren de nuevos elementos que los reemplacen.

Con referencia a lo anterior, investigaciones realizadas por Honig et al. [1], Kushleyev et al. [2], y Lucia et al. [3], plantean soluciones ante la carencia de sistemas GPS, mediante el uso de cámaras de visión (Vicon) para el rastreo y la determinación de la posición u orientación de los vehículos dentro de una región delimitada. Esto permite diseñar planes para la maniobrabilidad y guía de los drones, enfocados a la ejecución de determinadas tareas o acciones.

Publicaciones como [4], [5] entregan perspectivas diferentes en cuanto a la solución de problemas del movimiento de drones, involucrando sensores de rango tipo láser para la detección de obstáculos del entorno, con el fin de realizar la planificación en tiempo real de rutas que garanticen la navegación segura.

Las ideas inherentes en este conjunto de soluciones resultan demasiado costosas para ser replicadas y analizadas por la mayoría de estudiantes o centros de investigación con bajo presupuesto. Ante ello, trabajos efectuados por [6], [7] exponen soluciones y sistemas rentables para el tratamiento y estudio de los problemas de planificación del movimiento en vehículos aéreos no tripulados.

Según [6], el guiado y control de un UAV se puede trabajar mediante el uso de medidores y sistemas de visión embebida al vehículo, como la unidad de medición inercial (IMU – *Inertial Measurement Unit*) y la cámara. La IMU, en la determinación de la velocidad u orientación en el espacio. La cámara, en el reconocimiento de objetos o patrones del entorno (marcadores de posición) que conducen al vehículo por un camino libre de colisiones hacia su destino.

La propuesta de [7], al contrario de [6], precisa una solución factible al uso de cámaras Vicon u OptiTrack para el posicionamiento y control de un dron: la incorporación de un sistema de visión externo basado en el Microsoft Kinect del Xbox 360, cuyo procesamiento y tratamiento de datos está dado por el uso de librerías de MATLAB.

Acorde a lo expuesto, las soluciones presentadas enfatizan de una u otra manera el desarrollo de rutas dinámicas. Sin embargo, cuando el entorno se mantiene estático o los vehículos tienden a moverse por caminos repetitivos, es necesario replanificar el problema o buscar planeamientos que mejoren el desenvolvimiento de los drones en el entorno.

Por las consideraciones anteriores, se han seleccionado dos publicaciones que resumen la metodología a seguir al momento de enfrentarse con tales situaciones. Trabajos en [8], [9] empiezan analizando el problema por medio de la simplificación y la reestructuración del entorno de navegación, discretizando en regiones cuadriculares o de volumen los elementos u objetos que ponen en riesgo el movimiento de los drones. Esto les conlleva a aplicar algoritmos de búsqueda discreta, que les ayudan a determinar caminos factibles por cuales desplazar a los vehículos.

Al analizar las publicaciones basadas en el empleo de sistemas externos o internos para el control del movimiento de drones, y de algoritmos de búsqueda discreta en la obtención de rutas de navegación, se opta por las alternativas propuestas en [6], [8], [9] y se aplican al diseño de un sistema de planificación de rutas, cuyo objetivo es la conducción automática de cuatro vehículos en un entorno confinado. En este sistema de control se descarta la cámara de visión incorporada en los drones como la utilizada en [6], con la intención de no comprometer el rendimiento de un ordenador convencional al usar librerías de reconocimiento de imágenes como OpenCv o el *Image Processing Toolbox* de MATLAB empleado en [7].

La metodología en la realización de este trabajo se fundamenta en los principios del *Path Planning* bidimensional, que incluyen las técnicas de búsqueda discreta como Grassfire,

Dijkstra y A*, para adaptarlas progresivamente al contexto tridimensional de los vehículos aéreos. Tal adaptación se efectúa a partir de un replanteamiento del entorno, y por el análisis de las capacidades y limitaciones físicas de los drones.

Este documento se encuentra organizado en las siguientes secciones: Sección II, presenta la teoría y algoritmos del *Path Planning* bidimensional. Sección III, expone los conceptos y generalidades de un sistema de planificación de trayectorias para vehículos aéreos. Sección IV, describe las características del dron y los recursos que se usarán para su control. Sección V, estudia las limitaciones y restricciones del dron. Sección VI, adapta los algoritmos de búsqueda para su uso en el ámbito tridimensional. Sección VII, detalla el diseño y desarrollo del sistema de planificación de rutas. Sección VIII, plantea el proceso de comunicación, comando y supervisión de los drones, e indica su implementación. Sección IX, analiza los resultados del desenvolvimiento de los drones al ejecutar un conjunto de trayectorias. Sección X, exhibe las conclusiones.

En el desarrollo de este proyecto de tesis se emplea el lenguaje Python, y herramientas de software de código abierto que se describen continuación: NumPy para el cálculo numérico; Matplotlib [10] y vtkplotter [11] en la presentación de las gráficas; Panda3D en la construcción del entorno tridimensional y la representación de trayectorias; PyQtGraph junto a Pandas para la visualización en tiempo real y guardado de la odometría de los drones.

II. Fundamentos Del Path Planning

Path Planning es conocido en el ámbito de la robótica móvil, como un conjunto de estrategias que determinan las rutas que comunican dos puntos A y B de un entorno. Si estos puntos se sitúan en lugares donde exista presencia de obstáculos, la respuesta detrás del análisis será encontrar, en lo posible, caminos libres de colisiones que conecten dichos puntos [12]–[14].
Sea cual fuere la situación, cualquier estrategia que se elija estará condicionada bajo los siguientes aspectos [13]:

- Factibilidad, enfocada en la obtención de un resultado sin importar la eficiencia.
- Optimalidad, en garantizar la eficiencia del resultado.

En el análisis algorítmico existe un detalle importante al momento de evaluar una estrategia: determinar si su metodología de solución es completa [13, pp. 185-186]. Esto significa, que el algoritmo debe ser capaz de conducir a un resultado satisfactorio o un fallo, dentro de un intervalo finito de tiempo, con el objetivo de evitar bucles infinitos en el proceso de solución.

De acuerdo a lo manifestado, idear un plan en primera instancia puede resultar complejo si no se tiene noción abstracta de cómo abordar el problema. Sin embargo, existen algoritmos capaces de resolver problemas comunes del *Path Planning* que satisfacen los criterios de factibilidad, optimalidad o completitud [15]. Entre ellos se encuentran los algoritmos de planificación basados en descomposiciones geométricas del entorno.

A. *Algoritmos de planificación referenciados en descomposiciones geométricas del entorno*

Se caracterizan por trabajar de forma *offline* sin el requerimiento de sensores para la exploración y solución del entorno. Son veloces y eficientes al momento de encontrar una ruta libre de colisiones para el desplazamiento del robot. Estos algoritmos parten de una conceptualización básica del entorno, como es el área, los obstáculos y el espacio libre. La Fig. 1 expone una forma simple de distinguir estos elementos en un entorno continuo, y los define de la siguiente manera [15, p. 4]:

- Área o espacio de trabajo, W.
- Obstáculos, O_1, O_2.
- Espacio libre, la región resultante entre el espacio de trabajo y los obstáculos. Se expresa matemáticamente en (1) [14, p. 14]:

$$Espacio\ libre = W \setminus \bigcup_i O_i \tag{1}$$

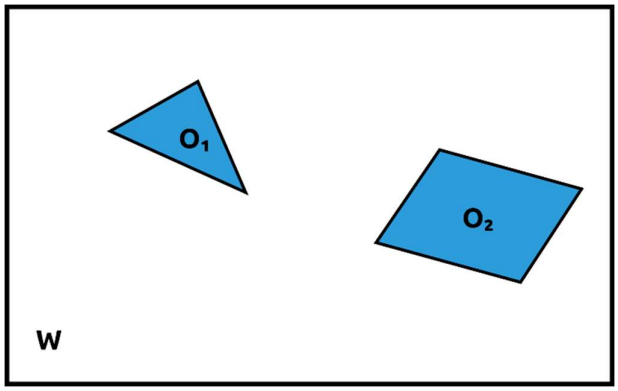


Fig. 1.Conceptualización continua del entorno en elementos de análisis [15].

Dicha conceptualización del entorno establece las bases para la representación discreta, donde el espacio libre (1) y los obstáculos se descomponen en porciones de área, tal como se muestra en la Fig. 2 y Fig. 3.

La Fig. 2 indica un espacio continuo de obstáculos poligonales segmentado en regiones convexas, cuya unión es equivalente al área de espacio libre definida en (1).

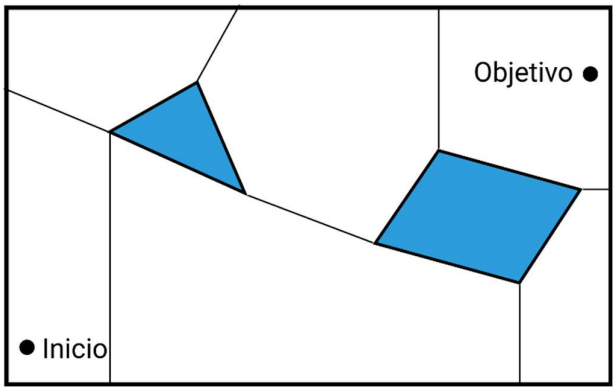


Fig. 2. Representación del entorno en regiones convexas [15].

La Fig. 3 presenta un criterio diferente al anterior, describe un entorno como un mapa de cuadrícula, donde el espacio libre corresponde al conjunto de casillas no ocupadas.

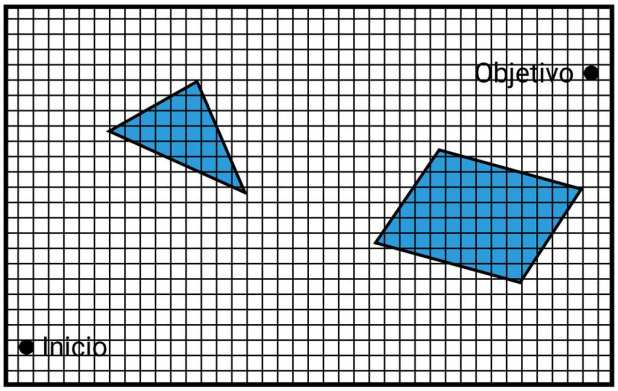


Fig. 3. Representación cuadriculada del entorno [15].

Según [12], [16], la técnica de cuadriculación es la más utilizada en el ámbito de la robótica móvil por la facilidad que implica la representación de un entorno cuadricular.

En términos de solución, los procedimientos para tratar con las dos representaciones del entorno (poligonal y cuadricular) radican en seleccionar los espacios libres y desplazarse consecuentemente hasta localizar la región o casilla que contenga el objetivo.

A continuación, se presentan las estrategias de solución que se aplican a entornos discretos.

1) Método de descomposición vertical y roadmap: Comprende la ejecución de dos procedimientos para el tratamiento del problema:

El primero, conocido como algoritmo de trapezoidación (Algoritmo 1). Se caracteriza por redefinir un entorno de polígonos irregulares en un entorno convexo con áreas de espacio libre (trapezoides). Para ello emplea segmentos verticales de recta (Fig. 4) que se construyen en función de los vértices de los obstáculos del entorno [14, pp. 165-168], [15, pp. 29-32], [17, pp. 45-46].

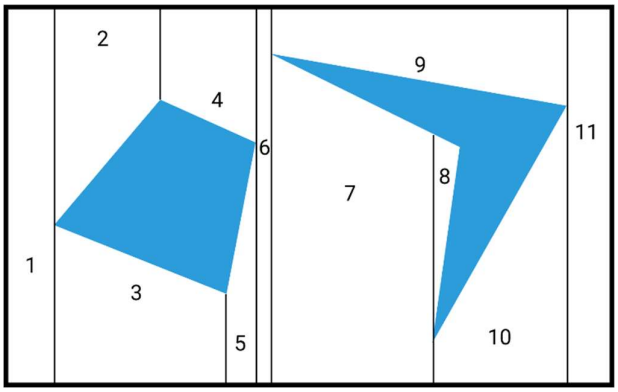


Fig. 4. Trapezoidación del entorno. Las numeraciones representan las regiones libres [14], [15], [17].

Algoritmo 1: Algoritmo de Trapezoidación

Entrada: Entorno poligonal
Salida: Lista de nodos trapezoidales cuya unión forma la región libre del entorno

1: crear una lista vacía T;
2: ordenar de izquierda a derecha los vértices de los obstáculos y del espacio de trabajo W;
3: **para** *cada vértice seleccionado en un barrido de izquierda a derecha* **hacer**
4: extender segmentos verticales hacia arriba y hacia abajo del vértice, hasta que intercepten un obstáculo o un límite del espacio de trabajo W;
5: Agregar a T todos los trapecios que se hayan formado por la fragmentación del entorno;
6: **fin**
7: **devolver** T

El segundo, denominado roadmap (Algoritmo 2) [15, pp. 33-34], marca los puntos centrales de cada región libre y sus límites para enlazarlas y formar uno o varios caminos (Fig. 5), de los cuales se elegirán solo aquel o aquellos que comuniquen con el objetivo.

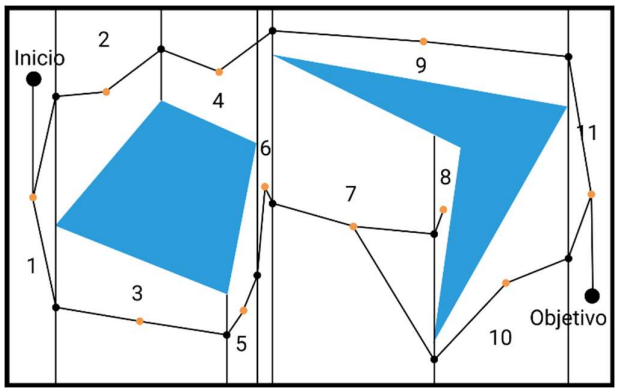


Fig. 5. Técnica roadmap [14], [15].

Algoritmo 2: Roadmap

Entrada: Lista con nodos trapezoidales
Salida: Roadmap

1: marcar los centros de cada trapezoide y sus fronteras;
2: **para** *cada trapezoide* **hacer**
3: conectar su centro con los puntos medios de sus fronteras;
4: **fin**
5: **devolver** *caminos formados por la unión de los puntos*

La búsqueda del objetivo implica el análisis de los caminos que enlazan cada región libre o celda trapezoidal (numeradas en la Fig. 5), tratándolas como aristas y nodos de un grafo interconectado, que al recorrerlo estratégicamente desde un nodo inicial, se llegue hasta el objetivo final determinando una ruta entre dichos puntos [14, pp. 163-164].

2) Método de descomposición por cuadrícula fija: Transforma los elementos del entorno en una combinación de cuadrículas elementales de iguales dimensiones [16, pp. 288-289], donde la solución del problema se traduce en una búsqueda por sectores, siendo válidas solamente aquellas casillas que no se encuentren ocupadas.

La Fig. 6 y Fig. 7 indican el proceso de descomposición de un entorno continuo definido por obstáculos poligonales.

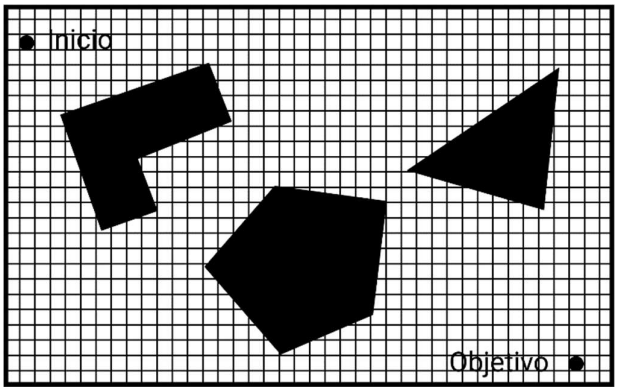


Fig. 6. Cuadriculación fija de un entorno de obstáculos poligonales [16].

El proceso de cuadriculación es el primer paso de una nueva representación del entorno, donde sus elementos se simplifican acomodándolos a las regiones cuadriculares como se detalla en la Fig. 7.

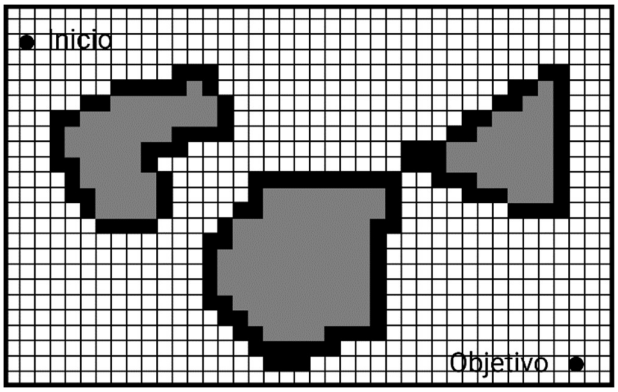


Fig. 7. Elementos del entorno ajustados a las regiones cuadriculares [16].

La Fig. 7 demuestra que los sectores libres constituyen en sí un grafo conectado, con un mayor número de nodos que el que se obtendría en un proceso de trapezoidación. Sin embargo, la cuadriculación ofrece la ventaja de constituir al entorno como un *puzzle*, donde se juntan elementos diferenciales para definirlo completamente.

Respecto a lo anterior, se recomienda tener cuidado al seleccionar un tamaño para las cuadrículas; una elección inadecuada ubicaría a las coordenadas del punto de inicio y el objetivo en las inmediaciones de la grilla, lo que impediría a cualquier algoritmo de búsqueda resolver el entorno, obteniéndose un mensaje de fallo como respuesta [16, p. 288], [18, p. 379].

Como las técnicas de trapezoidación y cuadriculación generan durante su ejecución grafos conectados, existen diversos algoritmos de búsqueda discreta que se pueden usar con el fin de determinar el camino más corto entre dos puntos del entorno [16, pp. 376-385]. Tales algoritmos son:

- Grassfire
- Dijkstra
- A* (A-Star)

a) Algoritmo Grassfire: Basa su estrategia de operación en función de una búsqueda por anchura [15, p. 39], ejecutando su recorrido a partir del nodo objetivo (nodo meta), y estableciendo como valor al nodo visitado la distancia Manhattan de su posición actual respecto al punto de partida [16, p. 381].

La distancia Manhattan para dos puntos $\boldsymbol{x} = (x_1, x_2, x_3, \dots, x_n)$ y $\boldsymbol{y} = (y_1, y_2, y_3, \dots, y_n)$ de un espacio n dimensional se define en (2) [19] como:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{2}$$

Desde la Fig. 8 a la Fig. 10 se indica el proceso de exploración y solución del algoritmo para un entorno cuadriculado. Las celdas libres representan los nodos de un grafo interconectado.

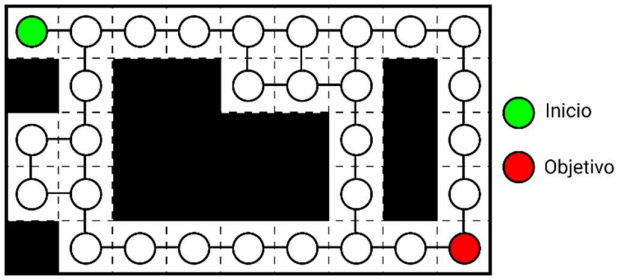


Fig. 8. Grafo interconectado del entorno. Los círculos en cada celda representan los nodos, y las líneas sólidas que los enlazan, las aristas del grafo [15].

En la Fig. 8, la ponderación de cada nodo en el grafo se desarrolla de acuerdo a (2), considerando que el valor de distancia entre celdas adyacentes es igual a la unidad, tal como se detalla en la Fig. 9.

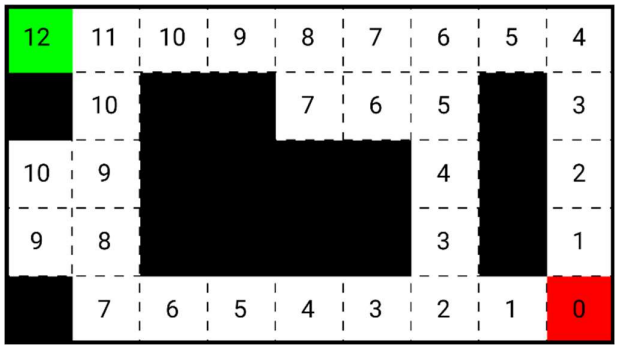


Fig. 9. Ponderación de los nodos del entorno a través de un análisis discreto de la distancia Manhattan. Cada celda está considerada como la unidad patrón de medida (en este ejemplo se toma un valor de 1) [15], [16].

A partir del análisis presentado en la Fig. 9, el proceso de solución implica recorrer desde el nodo inicial las celdas ponderadas del entorno, seleccionando como valores válidos únicamente aquellos inferiores en una unidad al valor previo. Véase la Fig. 10.

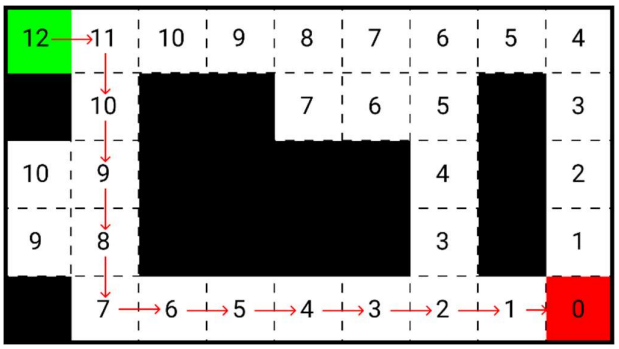


Fig. 10. Solución del entorno. Las flechas indican el camino más corto entre los puntos de inicio y destino (objetivo) [15], [16].

Según la solución de la Fig. 10, la distancia Manhattan del camino resultante es 12 unidades. Esto significa, que la celda inicial en la que se encontraría un robot móvil está a 12 casillas de la celda objetivo.

La metodología estratégica del algoritmo Grassfire, expuesta en los ejemplos, se detalla a continuación en el Algoritmo 3.

Algoritmo 3: Algoritmo Grassfire

Entrada: Grafo G del entorno, nodo n_{inicio} y nodo $n_{objetivo}$

Salida: Mensaje de fallo o una ruta entre n_{inicio} y $n_{objetivo}$

```
1:  crear una cola vacía Q;
2:  para cada nodo n en G hacer
3:      distancia(n):= ∞;
4:  fin
5:  distancia(n_objetivo):= 0;
6:  agregar n_objetivo a Q;
7:  mientras Q no esté vacía hacer
8:      n_actual := obtener(Q);
9:      para cada n vecino a n_actual hacer
10:         si distancia(n) = ∞ entonces
11:             distancia(n) = distancia(n_actual) + 1;
12:             agregar n a Q;
13:         fin
14:         si n = n_inicio entonces
15:             crear lista L := [n_inicio];
16:             n_aux := n_inicio;
17:             mientras n_aux ≠ n_objetivo hacer
18:                 para cada n vecino a n_aux hacer
19:                     si distancia(n) ≠ ∞ entonces
20:                         si distancia(n) = distancia(n_aux) - 1
                            entonces
21:                             agregar n a L;
22:                             n_aux := n;
23:                             break;
24:                         fin
25:                     fin
26:                 fin
27:             fin
28:             devolver L
29:         fin
30:     fin
31: fin
32: devolver mensaje de fallo
```

Cabe mencionar que, el proceso de búsqueda en que se sustenta el Algoritmo 3 causa que su tiempo de ejecución sea como se define en (3), según los criterios de [13, p. 35], [15, pp. 42-43]. Estos tiempos de ejecución o esfuerzo computacional se expresan comúnmente en notación $\mathcal{O}$. Su explicación se detalla en [20, pp. 10-16], [21, pp. 24-27], [22, pp. 39-41].

$$tiempo_{ejecución} = \mathcal{O}(n + m) \tag{3}$$

donde:

- n es el número de nodos
- m es el número de aristas del grafo

b) Algoritmo de Dijkstra: Se construye de un análisis distinto al desarrollado en el método Grassfire o la búsqueda por anchura. Su estrategia se basa en encontrar, reordenar y almacenar, durante el proceso de exploración, la ruta de mínimo coste que comunica a dos nodos de un grafo [15, pp. 105-107], [16, pp. 382-383]. Para ello, el algoritmo realiza ponderaciones a los nodos en función de los pesos de las aristas (segmentos que unen dos nodos), obsérvese la Fig. 11 y Fig. 12, pudiendo cambiar sus valores y ordenarlos durante la ejecución, si encuentra valuaciones que produzcan la menor ponderación al tomar los costos de los caminos que comuniquen con el nodo de partida.

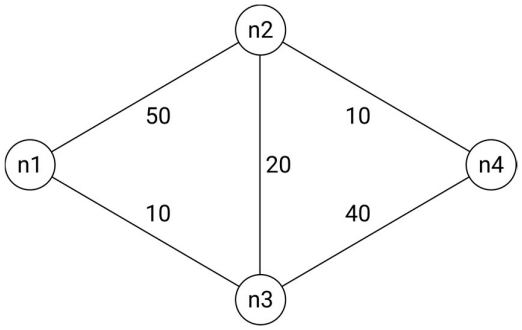


Fig. 11. Grafo ponderado. Los nodos están numerados de n1 a n4, y los pesos de las aristas por los valores numéricos descritos sobre ellas [15].

El resultado del algoritmo, a partir del análisis de Dijkstra, consiste en obtener del árbol ordenado los nodos del grafo con la menor ponderación, que involucren una ruta de conexión entre el nodo de partida y el objetivo, tal como se indica en la Fig. 12.

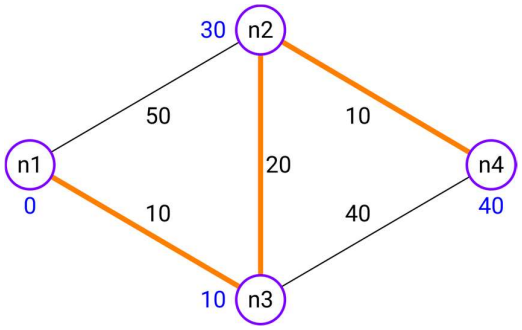


Fig. 12. Solución del grafo de la Fig. 11. Los nodos de inicio y destino se especifican como n1 y n4, respectivamente [15].

El Algoritmo 4 presenta el pseudocódigo que detalla toda la estrategia desarrollada por el algoritmo de Dijkstra.

Algoritmo 4: Algoritmo de Dijkstra

Entrada: Grafo G del entorno, nodo n_{inicio} y nodo $n_{objetivo}$

Salida: Mensaje de fallo o una ruta entre n_{inicio} y $n_{objetivo}$

```
crear una lista vacía L;
para cada nodo n en G hacer
    distancia(n):= ∞;
    padre(n):= Ninguno;
fin
distancia(n_inicio):= 0;
agregar n_objetivo a L;
padre(n_inicio):= n_inicio;
mientras L no esté vacía hacer
    n_actual := buscar n con la menor distancia en L;
    remover n_actual de L;
    si n_actual = n_objetivo entonces
        crear una lista R := [n_objetivo];
        n_aux := n_objetivo;
        mientras padre(n_aux) ≠ n_aux hacer
            n_aux := padre(n_aux);
            insertar n_aux al comienzo de R;
        fin
        devolver R
    fin
    para cada n vecino a n_actual hacer
        si distancia(n) > distancia( n_actual ) + peso(n_actual, n) entonces
            distancia(n) := distancia( n_actual ) + peso(n_actual, n);
            padre(n) := n_actual;
            agregar n a L si no está en ella;
        fin
    fin
fin
devolver mensaje de fallo
```

La complejidad computacional de este algoritmo depende del tipo de estructura de datos que se use para ordenar y almacenar las distancias de cada nodo explorado, obteniéndose diferentes tiempos de ejecución según lo señalan [13, p. 37], [15, pp. 107-108]:

- Tiempo de ejecución (4) por el uso de un arreglo o lista como estructura de datos, donde la selección de la menor distancia se obtiene por comparación de todos sus valores.

$$tiempo_{ejecución} = \mathcal{O}(n^2) \tag{4}$$

- Tiempo de ejecución (5) al usar una cola de prioridad en forma de montículo binario (*binary heap*) [23, Cap. 6].

$$tiempo_{ejecución} = \mathcal{O}\big((n+m)\log n\big) \tag{5}$$

- Tiempo de ejecución (6) por el uso del montículo de Fibonacci (*Fibonacci heap*) [23, Cap. 19].

$$tiempo_{ejecución} = \mathcal{O}(n\log n + m) \tag{6}$$

c) Algoritmo A:* Este tipo de algoritmo actúa de forma similar al propuesto por Dijkstra, con la ventaja que su método de búsqueda utiliza una función heurística que indica al algoritmo a explorar solamente aquellos nodos con la ponderación más baja, y que conduzcan de forma directa al objetivo [16, pp. 383-385], [24, pp. 80-81]. Dicha ponderación se calcula de acuerdo con su función de coste (7) [12, p. 605], [14, p. 531]:

$$f(n) = g(n) + h(n) \tag{7}$$

donde:

- $g(n)$ es el costo de la ruta formada entre el nodo de inicio y el nodo actual (n).
- $h(n)$ es la función heurística que especifica el valor de la distancia mínima entre el nodo actual y el objetivo.

Para la función heurística se puede emplear la distancia Manhattan (2) o la distancia euclidiana (8) [25, p. 349]. La selección de una u otra dependerá del entorno en el que se trabaje [14, pp. 527-528], [24, p. 80].

Una "correcta" elección de la heurística causará que la búsqueda sea eficiente, tome menos tiempo y devuelva como respuesta una solución óptima al problema. En el caso de un entorno convexo, como el que se expuso en la Fig. 4 y Fig. 5, la heurística más adaptable a sus condiciones es la distancia euclidiana.

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{8}$$

El proceso algorítmico A* se resume en el Algoritmo 5.

Algoritmo 5: Algoritmo A*

Entrada: Grafo G del entorno, nodo n_{inicio} y nodo $n_{objetivo}$

Salida: Mensaje de fallo o una ruta entre n_{inicio} y $n_{objetivo}$

```
crear una lista vacía L;
para cada nodo n en G hacer
    f(n) := ∞;
    g(n) := ∞;
    padre(n) := Ninguno;
fin
g(n_inicio) := 0;
f(n_inicio) := h(n_inicio);
agregar n_inicio a L;
padre(n_inicio) := n_inicio;
mientras L no esté vacía hacer
    n_actual := buscar n con la menor distancia en L;
    remover n_actual de L;
    verificar nodos (Función 1);
    para cada n vecino a n_actual hacer
        si g(n) > g(n_actual) + peso(n_actual, n) entonces
            g(n) := g(n_actual) + peso(n_actual, n);
            padre(n) := n_actual;
            agregar n a L si no está en ella;
        fin
    fin
fin
devolver mensaje de fallo
```

Función 1: Verificación de nodos y devolución de ruta

```
si n_actual = n_objetivo entonces
    crear una lista R := [n_objetivo];
    n_aux := n_objetivo;
    mientras padre(n_aux) ≠ n_aux hacer
        n_aux := padre(n_aux);
        insertar n_aux al comienzo de R;
    fin
    devolver R
fin
```

III. Path Planning y Planificación de Trayectorias Orientado a Vehículos Aéreos No Tripulados

Los principios o fundamentos del *Path Planning*, descritos en la sección II, se construyen desde una conceptualización bidimensional del entorno. Sus resultados son útiles al trabajar en la solución de problemas de movimiento para el contexto de los vehículos terrestres. Al tratar con entornos tridimensionales en el que navegan vehículos aéreos (drones o UAVs), no resultan del todo prácticos o por lo menos sus soluciones no podrían usarse de forma directa. De ahí que es necesario realizar abstracciones [26, Cap. 1] en el proceso de solución de tales estrategias, para adaptarlas a los problemas del movimiento tridimensional (Fig. 13). En este caso, el estado o configuración del dron en un punto de inicio P_s y en el objetivo (destino) P_f se caracteriza de acuerdo a su posición espacial (x, y, z), y su orientación (ϕ, θ, ψ) con respecto al entorno.

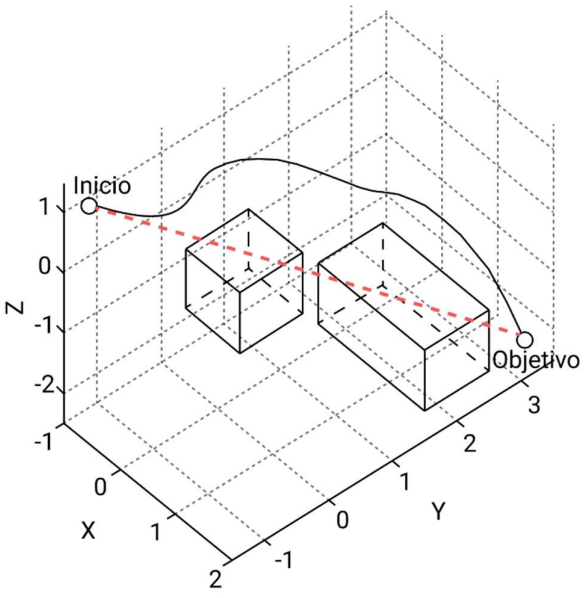


Fig. 13. Problema (línea discontinua) y resultado (línea curvilínea continua) de un algoritmo de planificación de rutas en tres dimensiones, enfocado en el movimiento de un vehículo aéreo que navega por el interior de un entorno confinado [27].

La ecuación (9) resume el propósito del *Path Planning* en tres dimensiones [26]:

$$P_s(x_s, y_s, z_s, \phi_s, \theta_s, \psi_s) \xrightarrow{r(q)} P_f(x_f, y_f, z_f, \phi_f, \theta_f, \psi_f) \quad (9)$$

siendo:

- ϕ, θ y ψ los ángulos de rotación longitudinal (*roll*), transversal (*pitch*) y vertical (*yaw*) del UAV (Fig. 14).
- $r(q)$ la ruta que comunica el punto de inicio con el objetivo.
- q representa las condiciones o restricciones geométricas de la ruta.

Fig. 14. Descripción de ejes y ángulos en un cuadricóptero [28].

De acuerdo con [26], las estrategias algorítmicas de planificación del movimiento deberán contemplar no solo la búsqueda y generación del camino (ruta de vuelo) libre de colisiones que conlleve hacia el destino como lo indica (9) y la Fig. 13; sino también las restricciones geométricas de la ruta, los estados y las limitaciones cinemáticas y dinámicas del vehículo que navegará a través del entorno. El conjunto de todas las restricciones transformará una ruta de comunicación $r(q)$ en una trayectoria de navegación.

Trabajos como [29]–[31], referentes al análisis y generación de trayectorias para drones autónomos, sugieren que, a partir de un camino formado por un conjunto de nodos o puntos del entorno, resultado de un proceso de búsqueda algorítmica o por asignación, la ruta podrá transformarse en una trayectoria al utilizar técnicas de interpolación polinomial. Estas técnicas, desarrolladas en [32, pp. 252-263], [33, Cap. 4], especifican que la trayectoria debe construirse como una función parametrizada en el tiempo, tomando como base los diferentes estados del robot, y estableciendo las restricciones de velocidad y aceleración a lo largo de la ruta sobre la que actuará, o en puntos clave de ella (waypoints).

La definición de la trayectoria polinomial se indica en (10); donde $P(t)$ representa el estado del dron para diferentes instancias de tiempo; n el grado del polinomio, establecido de acuerdo al número de restricciones cinemáticas impuestas (posición, velocidad, aceleración, etc.) para cada punto de la ruta; y a_n los coeficientes independientes, cuya solución satisfará las restricciones de la trayectoria.

$$P(t) = a_0 + a_1 t + \cdots + a_n t^n \quad (10)$$

Para el número de restricciones derivadas de (10), su solución procederá de la construcción de un sistema de ecuaciones lineales de orden $n \times n$.

Un aspecto a destacar, es que el estado $P(t)$ del dron puede construirse como lo describe [31], de acuerdo al orden del

sistema que representa el comportamiento dinámico del UAV para evitar discontinuidades en las soluciones de (10).

Pese a las limitaciones y restricciones estipuladas hasta el momento en el caso de un solo dron, existen consideraciones a tomar cuando se involucra el movimiento conjunto de varios vehículos de forma simultánea. La complejidad en la búsqueda de la solución se incrementa, porque el algoritmo deberá construir una ruta individual para cada UAV que evite la colisión entre rutas, permitiendo que el resultado final garantice la seguridad en el vuelo (Fig. 15).

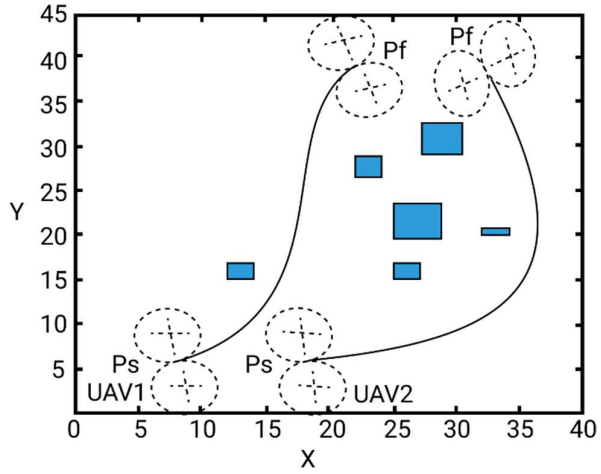


Fig. 15. Rutas de vuelo seguras observadas desde un enfoque 2D. La vista de los UAVs se encuentran amplificadas [26].

El análisis restrictivo del movimiento convierte a una estrategia de solución sencilla (derivada de un algoritmo de búsqueda), en un sistema completo de planificación de rutas que genera soluciones seguras para el movimiento de los UAVs en el espacio [26].

IV. Descripción Del Dron, Software Y Recursos

El Tello EDU (Fig. 16) es un mini dron de uso interior, que pertenece al grupo de drones denominados centimétricos. Se distingue por ofrecer características importantes como la formación de enjambres (*swarm*), el reconocimiento de patrones, y una controlabilidad decente de sus movimientos gracias a su kit de desarrollo de software (SDK – *Software Development Kit*) [34].

Fig. 16. Tello EDU [35].

El SDK está adaptado originalmente al lenguaje de programación Python, pero existen opciones y ejemplos de código no oficiales desarrollados en los lenguajes C++, C# o JavaScript. A esto se añade una alternativa sólida como el framework Gobot (lenguaje Go) [36], que dispone de un driver para el control del dron y cubre hasta un 70% las funciones del SDK.

A. *Dimensiones del dron*

Las medidas aproximadas del dron, incluido los protectores de hélices, son: 175mm x 170mm x 50mm. En la Fig. 17 se indica las medidas de su ancho y su largo.

Fig. 17. Dimensiones del Tello EDU [35].

B. *Componentes del dron*

El Tello EDU está constituido por 4 componentes importantes:

- Unidad de medición inercial (IMU).
- Cámara frontal de 5 megapíxeles, que incluye la grabación de vídeo a 720p.
- Cámara inferior, que trabaja a una frecuencia máxima de 20Hz y una frecuencia mínima de 10Hz en el reconocimiento de patrones [37].
- Sensor de proximidad infrarrojo, cuyo conjunto con la cámara inferior forman un sistema de posicionamiento automático y detector de colisiones. Véase la Fig. 18.

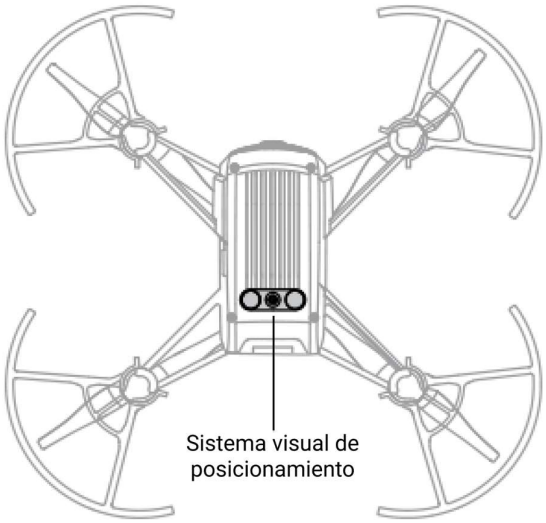


Fig. 18. Sistema visual de posicionamiento [38].

C. *Elementos de posicionamiento externo*

Constituyen un grupo de 8 marcadores denominados Mission Pads (Fig. 19). Actúan de forma similar a los ArUco Markers [39] en la estimación de la localización u orientación del dron, y poseen atributos (solamente distinguibles por el Tello EDU) que ayudan a su estabilidad en el despegue y aterrizaje, así como la corrección de su posición durante la ejecución de movimientos cartesianos en el plano. Véase la Fig. 20.

Fig. 19. Mission Pads numerados del 1 al 8 [35].

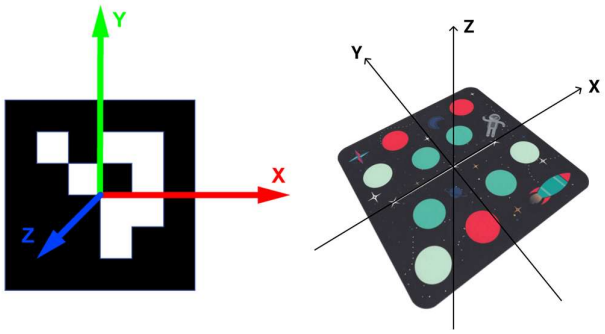


Fig. 20. Comparativa entre un ArUco Marker y un Mission Pad [37], [40].

Pese a las ventajas que aportan los Mission Pads en el movimiento de los drones, existe un problema latente: la limitación de su rango de acción. Según el manual de uso [37], la región efectiva de un Mission Pad está comprendida entre 0.3m y 1.2m de altura, así como de una desviación admisible de hasta 1m en el plano horizontal.

Tal limitación será la razón por la que no se utilizarán de forma funcional estos elementos de posicionamiento. Solamente se incluirán en el desarrollo de este documento y en los experimentos como puntos de referencia para el usuario.

D. *Interfaz de calibración*

Forma parte de la aplicación de control manual del dron. Se incluye como un recurso para la calibración de la IMU y el centro de gravedad del vehículo. Véase la Fig. 21.

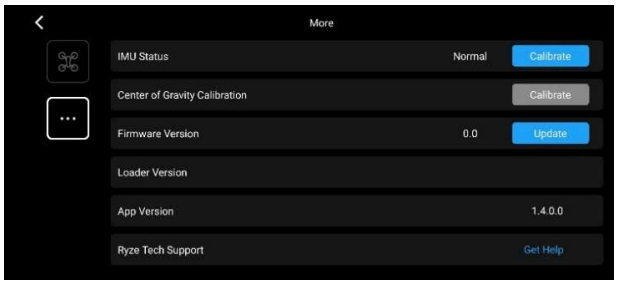


Fig. 21. Interfaz de calibración (captura de pantalla).

V. Limitaciones, Restricciones Físicas Del Dron, Y Márgenes De Seguridad Entre Rutas

Previo al proceso de abstracción de las técnicas de *Path Planning* [26] para su empleo en ambientes 3D, se seleccionarán las limitaciones físicas del dron. Esto permitirá el desarrollo de algoritmos que estén en concordancia con sus capacidades, y que los resultados se conviertan en rutas factibles a ser ejecutadas.

A. *Limitaciones del dron*

El Tello EDU posee varias restricciones inherentes, cuyo conjunto impide ejecutar cierto tipo de acciones o desplazamientos en el entorno, las cuales son [34], [38]:

- Altura mínima y máxima de vuelo: Limitada a una altura mínima de 0.50m y una altura máxima de 6m. Los valores dentro de este rango garantizan el adecuado funcionamiento del sistema de posicionamiento embebido (Fig. 22).
- Altura de despegue: Aproximadamente 0.80m.
- Nivel de iluminación del entorno: Condicionado por el sistema de posicionamiento del dron. Se recomienda un valor cercano a la media del rango: 10lx < nivel < 100 000lx.
- Tiempo de vuelo: Aproximadamente 13 minutos en ambientes interiores.
- Velocidad mínima y máxima: Restringida al intervalo de 10 a 100cm/s respectivamente.
- Distancia de comunicación: Depende de las capacidades de la interfaz de comunicación inalámbrica (Wifi). De acuerdo con el fabricante del dron, esta distancia podría llegar hasta los 100m.

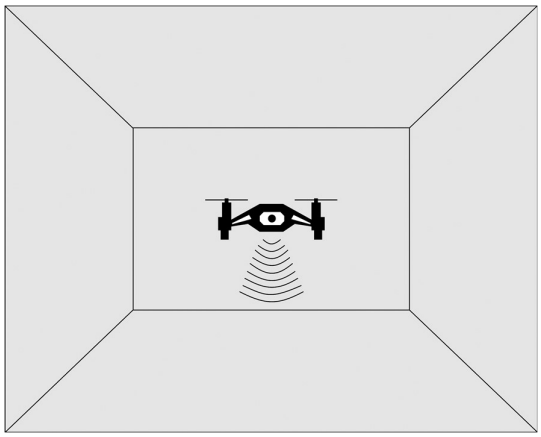


Fig. 22. Sistema visual de posicionamiento: mantiene la estabilidad del dron [38].

B. *Restricciones físicas y geométricas impuestas para el planificador de rutas*

Son las restricciones que confieren las condiciones que se entregarán al sistema de planificación del movimiento en la generación de rutas y trayectorias satisfactorias. Las restricciones abarcan las limitaciones señaladas por el fabricante y las que se impone el diseñador de algoritmos, para el diseño de un sistema de *Path Planning* o de planificación de trayectorias. La serie de imposiciones en que se fundamentará el desarrollo del sistema de planificación se resume en lo siguiente:

1) Restricciones físicas del dron: Abarca la altura mínima y máxima del dron (apartado A de esta sección), y un límite de velocidad propuesto en 50cm/s para que el vehículo pueda ejecutar la ruta sin mayor inconveniente.

2) Restricciones geométricas de la ruta (q): Constituyen las limitaciones de posicionamiento del dron y los márgenes mínimos de seguridad entre rutas para el vuelo seguro. Estas condiciones involucran: altura mínima y máxima del dron; desplazamiento mínimo lateral y vertical que puede ejecutar el dron, de acuerdo a las limitaciones de su controlador interno; distancia de separación vertical y lateral entre las diferentes rutas a generar.

a) Limitaciones en el desplazamiento lateral y vertical: La Fig. 23 indica las restricciones de distancia mínima para el desplazamiento lateral del dron. Según la descripción del SDK esta distancia no será menor a los 20cm.

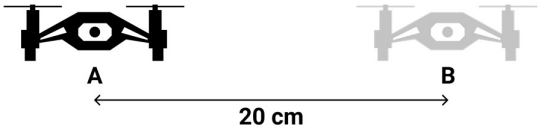


Fig. 23. Desplazamiento mínimo entre dos puntos del entorno [34], [38].

Para el desplazamiento vertical, la distancia permitida, al igual que en el movimiento lateral es de 20cm, con la consideración que el desplazamiento debe realizarse cuando el dron haya despegado (0.80m) o se encuentre en un nivel de altura igual o mayor al mínimo recomendado.

Los desplazamientos pueden ser de ascenso o descenso, teniendo presente que en el descenso la distancia de desplazamiento no provoque llegar al límite mínimo de altura sobre el nivel del suelo: 0.50m (Fig. 24). Caso contrario, el sistema de posicionamiento, a través de su controlador, no podrá mantener al dron en una posición fija, causando movimientos erráticos de la aeronave y posibles colisiones con otros drones u objetos del entorno.

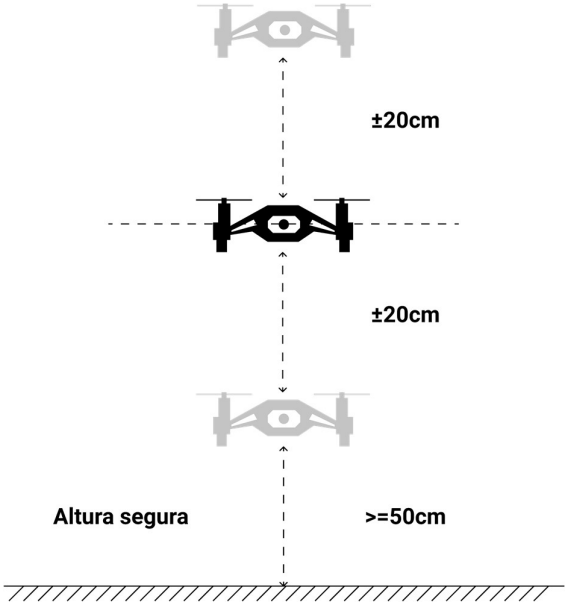


Fig. 24. Desplazamientos verticales mínimos entre dos puntos del entorno [34], [38].

b) Márgenes de separación entre rutas: Los márgenes de separación entre rutas, conforme al número de vehículos involucrados en la navegación, se determinan en función de su tamaño y acorde al límite de separación vertical (altura segura) entre el dron o cualquier objeto del entorno.

Estas distancias de separación se resuelven al considerar la geometría del dron como una caja, donde sus fronteras se extienden hasta alcanzar un volumen que garantice protección contra colisiones, y cuyo centro geométrico aloja una partícula representando la masa total del vehículo. Véase la Fig. 25.

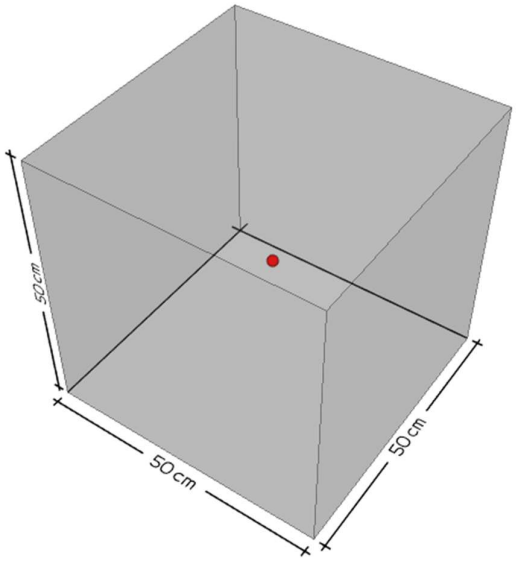


Fig. 25. Representación de las fronteras del dron asignadas a un valor de 50cm por lado.

El cubo en la Fig. 25, delimita la región espacial de seguridad del dron en un valor escogido empíricamente a partir de sus dimensiones (Fig. 17), y con adición de valores de tolerancia. Los valores podrían ser superiores a los propuestos, dependiendo del margen de seguridad que se pretenda entregar. No se recomiendan valores menores a los indicados, porque se causaría colisiones entre drones.

Al sumar la restricción volumétrica y la altura de seguridad del dron, medida a partir de un límite de su frontera, da como resultado las distancias de separación mínimas que se exponen en la Tabla I.

TABLA I. DISTANCIAS DE SEPARACIÓN ENTRE RUTAS

Separación entre rutas	
Separación vertical	50cm
Separación horizontal	50cm

Los valores de separación vertical y horizontal entre rutas se detallan en la Fig. 26. La distancia mínima de separación horizontal y vertical está dada en función de los centros geométricos de dos vehículos contiguos.

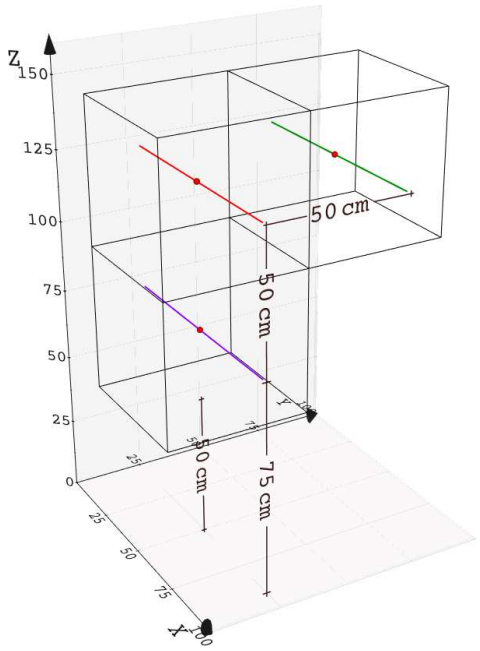


Fig. 26. Separación mínima entre rutas y altura segura. Las rutas se especifican por las rectas de colores.

VI. Abstracción De Los Algoritmos De Búsqueda Para Su Uso en Ambientes 3D

La abstracción de los algoritmos de búsqueda conlleva a una reestructuración de su lógica, de forma que los resultados sean factibles acorde al entorno y conforme a las restricciones de movimiento de los vehículos aéreos.

Al tratarse de una planificación de rutas en entornos 3D controlados, en los que se conoce la ubicación y el tamaño de sus elementos, resulta factible realizar una simplificación de sus formas. Esta simplificación convertirá los obstáculos del entorno y el espacio completo en un conjunto de objetos poligonales, cuadriculares, o vóxeles como en [9], de manera que se puedan aplicar como base las estrategias de solución involucradas en los métodos de descomposición de la sección II.

Todo el proceso de abstracción algorítmico se realizará en tres etapas:

A. *Aplicación directa de los métodos de descomposición en entornos 3D*

Se caracterizan por tratar a los obstáculos más representativos del entorno, definidos en formas poligonales o cuadriculares (sección II), como proyecciones perpendiculares de altura uniforme, basadas en aspectos generales de su forma o en el área que ocupan en el plano.

Considerar a los objetos de altura uniforme como proyecciones en el espacio, permitirá al algoritmo realizar una búsqueda en el plano (X, Y), cuyo resultado, en caso de ser satisfactorio, transformarlo a una ruta 3D al añadir a cada uno de los puntos un valor Z de magnitud constante.

1) Método de descomposición vertical: La aplicación de este método parte de una elección y simplificación previa de un entorno acotado, en el que los objetos relevantes (paredes, pilares, muebles, etc.) se resumen a formas poligonales irregulares (Fig. 27) con el propósito de obtener un mapa del entorno factible para analizar y encontrar una solución.

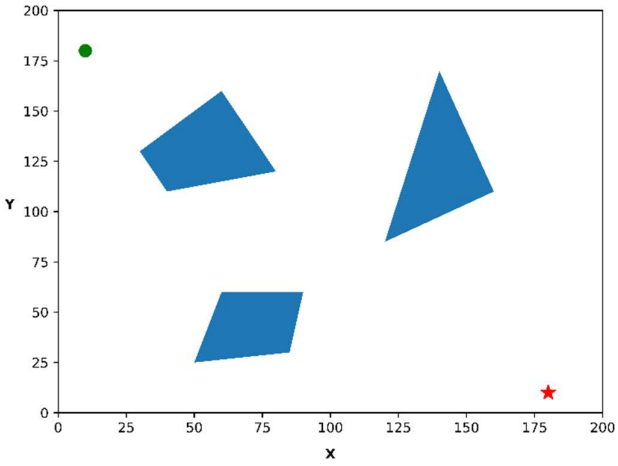


Fig. 27. Mapa del entorno de 200cm de lado. Las marcas de color verde y roja representan las coordenadas de inicio (10, 180) y destino (180, 10).

Los obstáculos del entorno en la Fig. 27 están dispuestos de forma que exista una separación óptima entre ellos para que el dron (Fig. 17) pueda navegar sin riesgo de colisión.

El proceso de solución del entorno propuesto consiste de dos etapas: aplicar el algoritmo de trapezoidación; emplear la técnica roadmap para la obtención de la ruta que comunica el punto de inicio con el destino u objetivo.

a) Aplicación del algoritmo de trapezoidación: La trapezoidación o discretización del entorno en regiones convexas se realiza acorde al Algoritmo 1 de la sección II. Su resultado se indica en la Fig. 28.

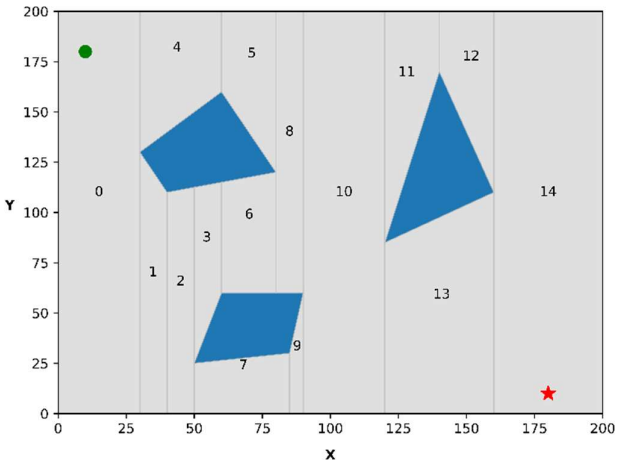


Fig. 28. Discretización del entorno. Las regiones libres representan los nodos de un grafo, y están numeradas acorde al proceso de descomposición vertical (izquierda a derecha).

b) Aplicación de la técnica roadmap: De la segmentación en la Fig. 28, el método roadmap (Algoritmo 2) marca los puntos centrales de las regiones y sus límites con el fin de evaluar las posibles rutas que se puedan formar mediante la unión de sus puntos (Fig. 29).

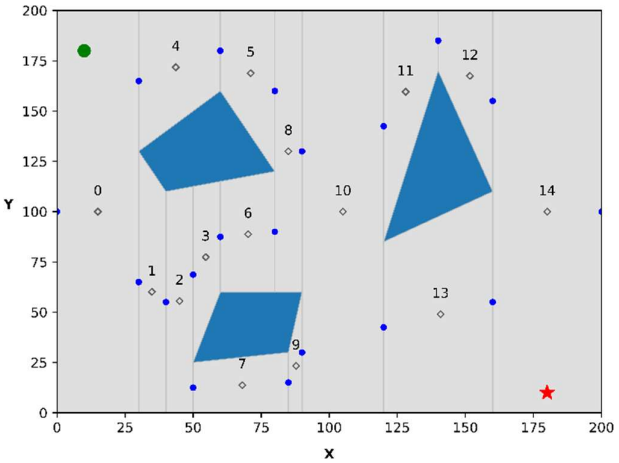


Fig. 29. Técnica roadmap.

La obtención de la ruta (Fig. 30) se da por aplicación del Algoritmo 4 y una estructura de datos que produce el rendimiento (4). Los costes de los nodos están caracterizados por el valor de la distancia euclidiana (8).

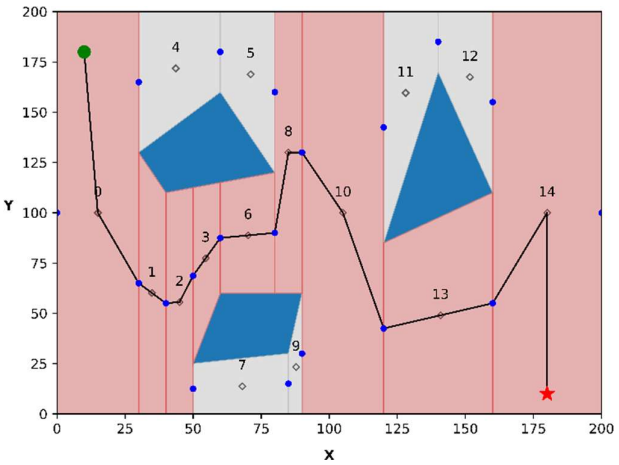


Fig. 30. Ruta generada como resultado del proceso de búsqueda de Dijkstra.

A partir de esta construcción de la ruta, y con el propósito de aplicarla como referente para los desplazamientos en el entorno 3D, la solución consistirá en lo siguiente: realizar las extrusiones de los obstáculos del entorno en el eje perpendicular al plano (eje Z); y establecer un nivel de altura de la ruta en un valor mayor o igual al mínimo recomendado para la navegación segura. Véase la Fig. 31.

Aunque la solución propuesta parezca razonable, tiene dos debilidades que la vuelven impráctica. La primera: limita los movimientos del dron al plano horizontal (X, Y). La segunda: la ruta generada tiene características no compatibles con el sistema dinámico del dron, es decir, existen cambios bruscos de dirección en algunos de sus puntos. Una posible solución a esta incompatibilidad dinámica, consiste en convertir la ruta generada en una trayectoria empleando las técnicas de interpolación polinomial como lo sugieren [29]–[31]. Pero dado que se pretende usar el sistema de control y posicionamiento automático del dron, no se optará por dicha estrategia.

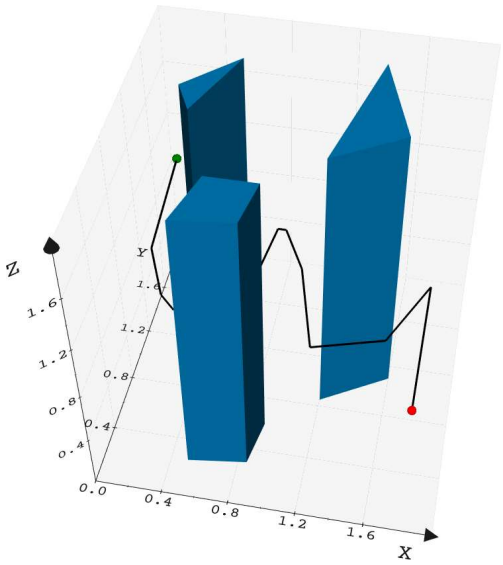


Fig. 31. Ruta bidimensional transportada a 1m de altura.

2) Método de descomposición por cuadrícula fija: El análisis de este método, que involucra a los algoritmos Grassfire, Dijkstra y A*, se efectuará acorde a un mapa discretizado del entorno, como se indica en la Fig. 32. Los puntos de inicio (1, 1) y destino (10, 10) se ubican de forma similar a como se lo hace en una matriz de datos.

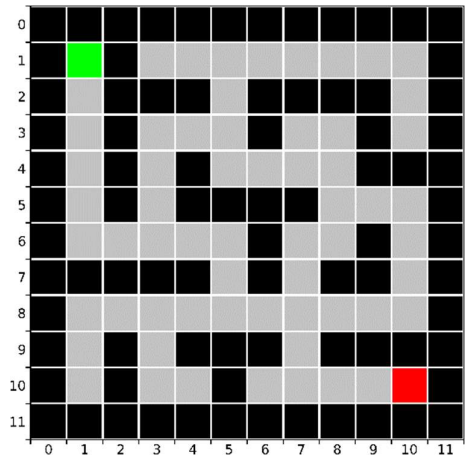


Fig. 32. Entorno cuadriculado.

La representación discreta permite establecer una medida uniforme para todas las celdas, de modo que el entorno definido por un número de filas y columnas podrá representar un entorno de varios metros cuadrados. Sin embargo, el valor más pequeño de la celda no deberá ser menor a la región de seguridad del dron, y la distancia entre celdas vecinas ser igual o mayor al desplazamiento mínimo lateral recomendado por el fabricante.

La estructura de datos que se usará en la ejecución de los algoritmos será una cola de prioridad que produzca el rendimiento en (5).

a) Algoritmo Grassfire: La Fig. 33 detalla la estrategia de búsqueda del algoritmo Grassfire (Algoritmo 3) para el entorno de la Fig. 32. Cada celda del entorno está ponderada según la distancia Manhattan (2) respecto al nodo objetivo.

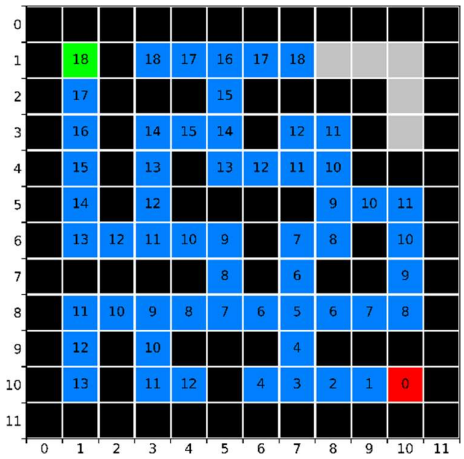


Fig. 33. Proceso de exploración y búsqueda del objetivo.

Encontrar la ruta entre las celdas de inicio y destino, valuadas como 18 y 0 respectivamente, conlleva desplazarse por las celdas de menor ponderación hasta alcanzar la celda objetivo de valor cero (Fig. 34).

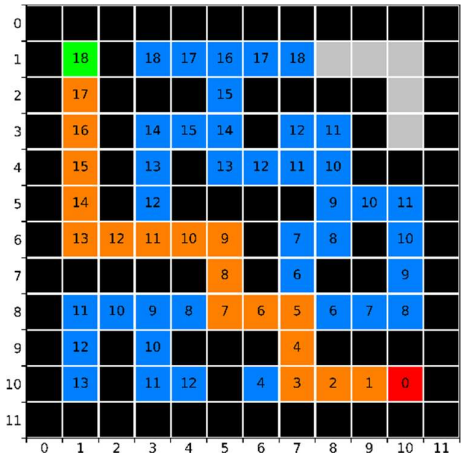


Fig. 34. Obtención de la ruta acorde a las valuaciones de las celdas.

b) Algoritmo de Dijkstra: La aplicación de este método procede según el pseudocódigo del Algoritmo 4 para el entorno de la Fig. 32. Su proceso de búsqueda se expone en la Fig. 35.

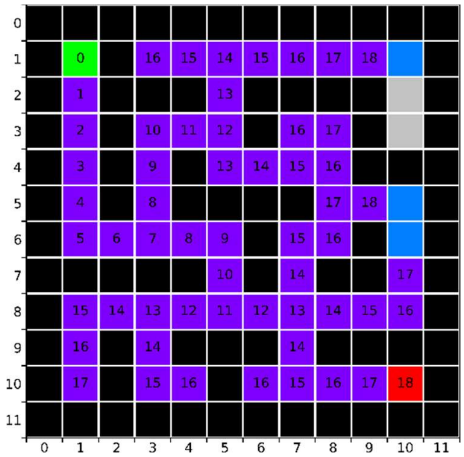


Fig. 35. Búsqueda del objetivo. Los nodos visitados corresponden a las celdas numeradas, y los nodos explorados están señalados en color azul.

De acuerdo con la Fig. 35, el algoritmo de Dijkstra se diferencia del Grassfire en cuanto a la forma de expandir los nodos y establecer sus ponderaciones. El algoritmo empieza realizando una exploración de los nodos vecinos (celdas azules) a un nodo actual, para luego ponderarlos en función de la distancia de mínimo coste respecto al nodo de partida.

Según esta prueba particular, el número de nodos explorados es casi similar al método Grassfire, deduciéndose que, en el peor de los casos, el algoritmo de Dijkstra se comportará de forma similar a una búsqueda por anchura.

En cuanto a la obtención de la ruta, se procede desde la celda destino (objetivo), seleccionando los nodos con la ponderación más baja hasta llegar a la celda de inicio, tal como se muestra en la Fig. 36.

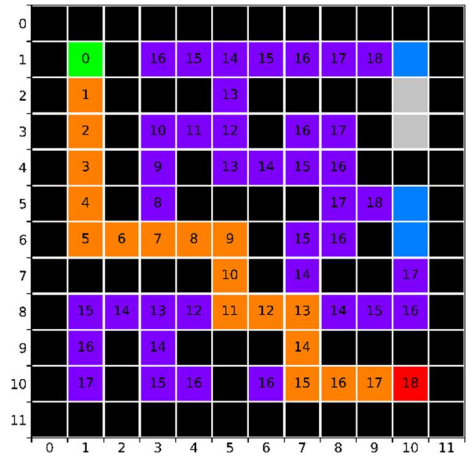


Fig. 36. Generación de la ruta entre el nodo objetivo y el nodo inicial.

c) Algoritmo A:* Al tratarse de la solución de un entorno discreto (Fig. 32), la función heurística a emplear para la ejecución del algoritmo A* (Algoritmo 5) será la distancia Manhattan (2). Su desarrollo de búsqueda se indica en la Fig. 37.

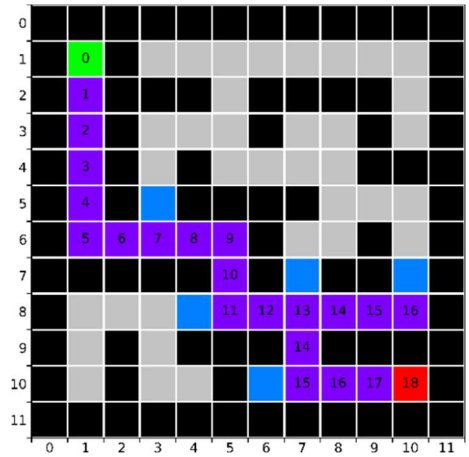


Fig. 37. Proceso de búsqueda de la celda objetivo. Los nodos visitados corresponden a las celdas numeradas, y los nodos explorados están señalados en color azul.

La Fig. 37 demuestra la eficiencia del algoritmo A*. El uso de una función heurística en la determinación de la ruta de mínimo coste hacia el nodo objetivo, cambia por completo la exploración del entorno. Según se aprecia, el algoritmo tiende a dirigirse muy rápidamente hacia el objetivo, disminuyendo

así la cantidad de nodos visitados en comparación con el algoritmo de Dijkstra y Grassfire.

La ruta generada para esta solución es exactamente igual a la aplicada en el algoritmo de Dijkstra, como se lo indica en la Fig. 38.

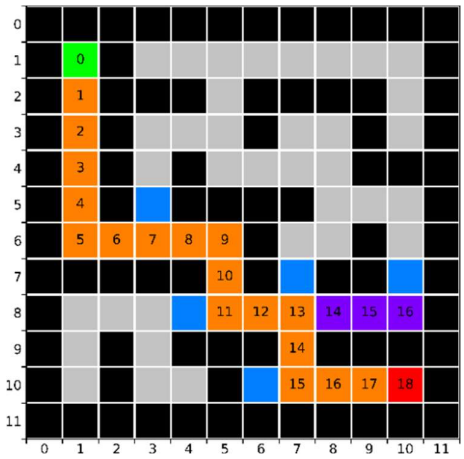


Fig. 38. Generación de la ruta entre el nodo objetivo y el nodo inicial.

Analizadas las tres estrategias de solución, se elegirá el algoritmo A* como base de búsqueda por ser el más eficiente. Servirá perfectamente para la exploración de entornos con un alto número de nodos, como los que se tratará posteriormente en los entornos 3D.

De cualquiera de las soluciones indicadas en la Fig. 34, Fig. 36 y Fig. 38, por ser iguales, se puede tomar una de ellas y convertir la ruta generada en un camino de tres dimensiones (Fig. 39) aplicando el mismo criterio que en el caso del algoritmo de descomposición vertical.

El camino resultante es compatible con la dinámica del dron (Tello EDU). Los ángulos de giro en algunos puntos de la ruta siempre son de 90 grados, por lo que no existirán "cambios bruscos" en su movimiento. Los desplazamientos entre celdas están acorde a la distancia mínima lateral exigida por el fabricante.

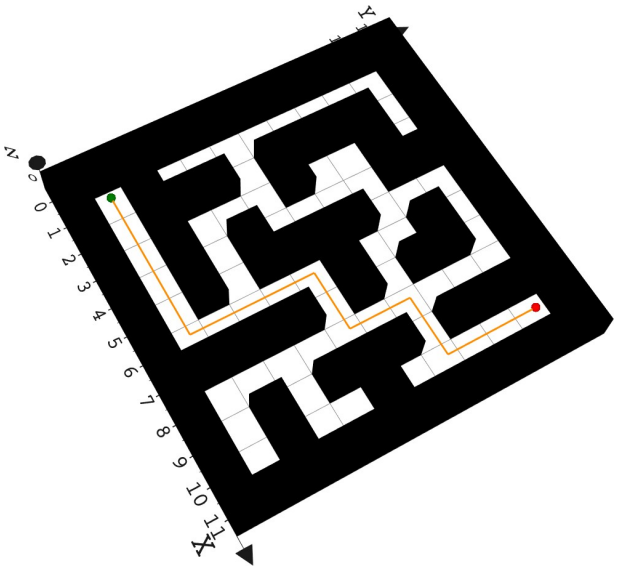


Fig. 39. Ruta 3D de un entorno discreto. Las filas están consideradas en la dirección del eje X y las columnas en el eje Y.

B. Conversión del método de cuadriculación fija para su aplicación en entornos 3D

Esta estrategia de conversión partirá de la conceptualización del entorno 3D como una agrupación de unidades elementales de volumen o vóxeles, similar al proceso de cuadriculación fija, pero desde un contexto tridimensional (Fig. 40). Los tamaños de las unidades volumétricas serán homogéneos, y estarán definidos por las dimensiones de la región de seguridad del dron, establecidas para el ejemplo particular a 50cm de lado (Fig. 25). La resolución del entorno y el número de nodos en el espacio libre son dependientes de las dimensiones de los vóxeles.

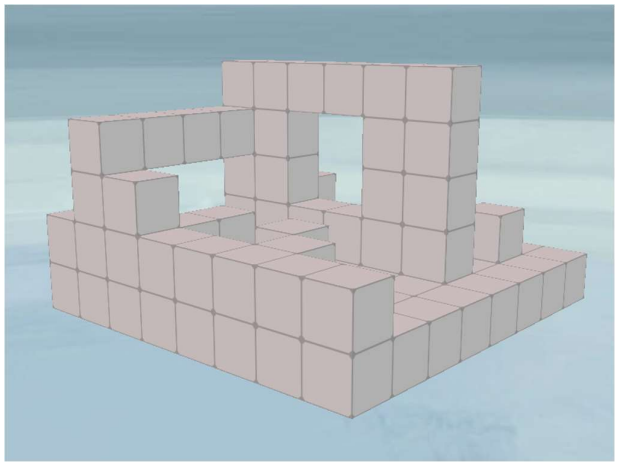


Fig. 40. Entorno 3D construido por elementos de volumen de tamaño homogéneo.

Solucionar el entorno voxelizado, con un alto número de nodos, conlleva adaptar el método de búsqueda del algoritmo A* a la exploración de nodos en el espacio. Para ello, se propone que la selección de los nodos vecinos a un nodo particular del espacio libre se realice según la estrategia que se expone en la Fig. 41.

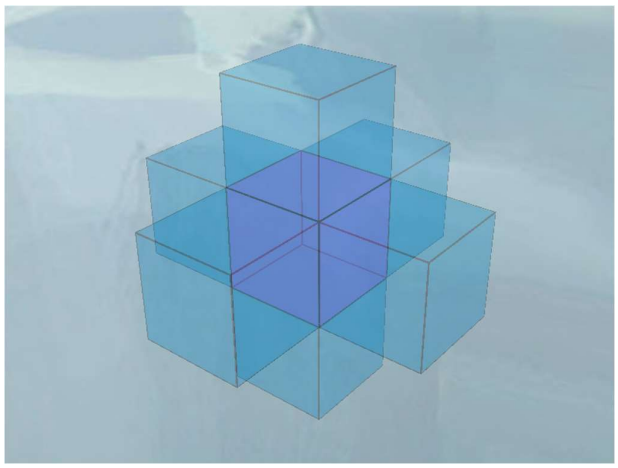


Fig. 41. Selección de los vecinos a un nodo actual (color púrpura). Se ha amplificado la vista para una mejor visualización.

Las ponderaciones de los nodos se realizan según la magnitud de distancia (peso de la arista) entre los puntos centrales de los vóxeles, y de su costo evaluado con respecto al nodo de partida o nodo inicio.

En cuanto a la heurística del algoritmo, se usará la distancia Manhattan (2) como función de evaluación, para excluir desplazamientos diagonales en la generación de la ruta.

La Fig. 42 presenta la solución del entorno 3D (Fig. 40) al modificar la búsqueda del algoritmo A*, según el criterio propuesto en la Fig. 41.

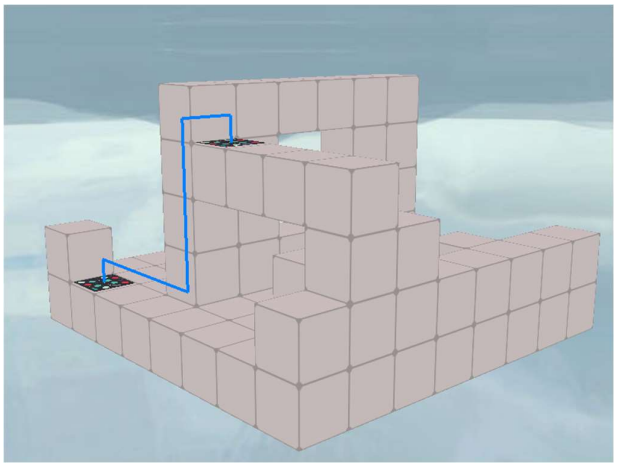
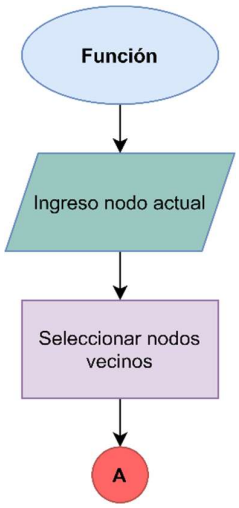


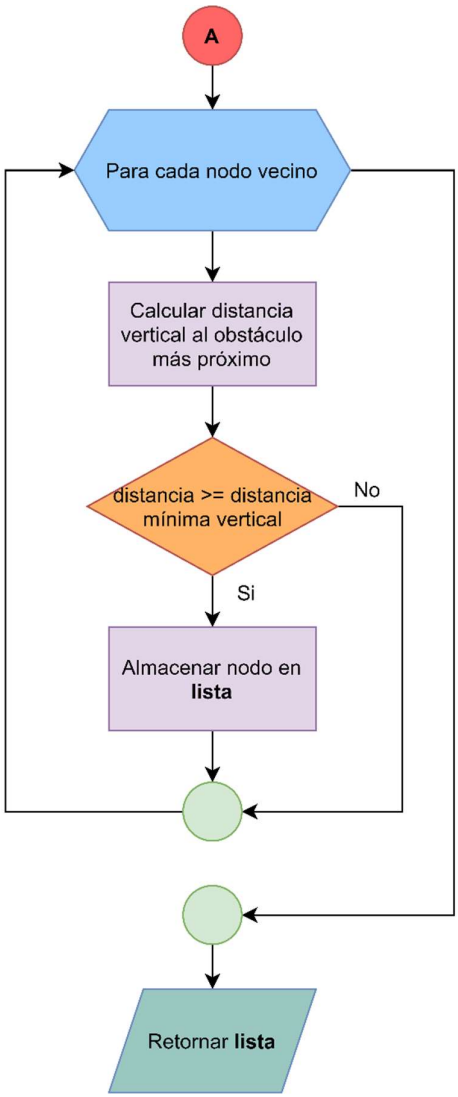
Fig. 42. Solución del entorno 3D. Los puntos de inicio y destino están marcados de izquierda a derecha y corresponden a las coordenadas (1, 0, 0) y (4, 1, 3), respectivamente.

El resultado de la Fig. 42 no satisface la condición de separación de ruta mínima entre el dron y el suelo, tampoco considera la altura inicial de despegue del dron, definida en la sección V apartado A; por tanto, se abordará el problema bajo los siguientes criterios:

- Redefinir la selección de los nodos en el Algoritmo 5 según el diagrama de flujo de la Fig. 43.
- Realizar la búsqueda del objetivo a partir del nodo ubicado a una distancia libre con respecto al piso, igual o superior a la altura de despegue del dron. Este valor se acomodará según la resolución del entorno.



(a)



(b)

Fig. 43. Función de selección de nodos vecinos, partes (a) y (b).

Finalmente, el proceso de exploración basado en las modificaciones previas del Algoritmo 5 produce una ruta factible a ser ejecutada (Fig. 44).

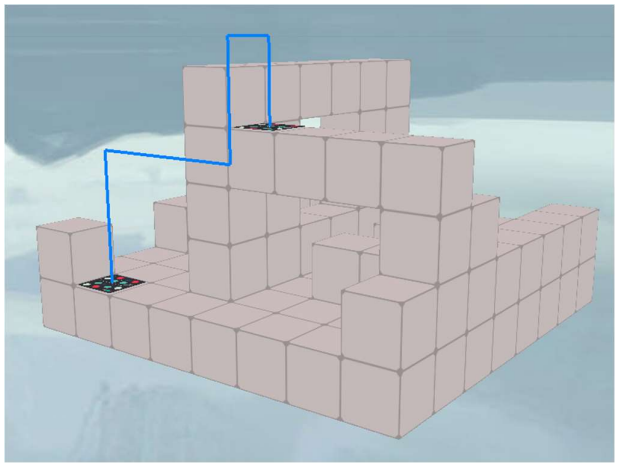


Fig. 44. Ruta generada en función de las condiciones de mínima separación vertical y altura de despegue.

C. *Ampliación del algoritmo de búsqueda 3D para el manejo de múltiples UAVs conforme al algoritmo A* modificado*

La ampliación de la estrategia de búsqueda 3D, analizada en el apartado B, se concreta al considerar cada ruta generada, según las coordenadas de inicio y destino, como obstáculos del entorno; de manera que el algoritmo omita la exploración de todos esos nodos, y sus resultados finales garanticen rutas de vuelo seguras para cada uno de los drones involucrados.

La Fig. 45 y Fig. 46 presentan las rutas generadas para cuatro drones que navegarán por el entorno, cuyas dimensiones de vóxel son de 50cm por lado.

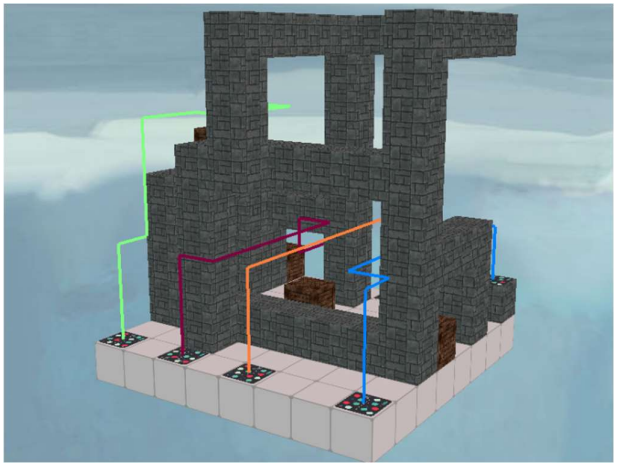


Fig. 45. Rutas de vuelo seguras. Sección de los puntos de origen.

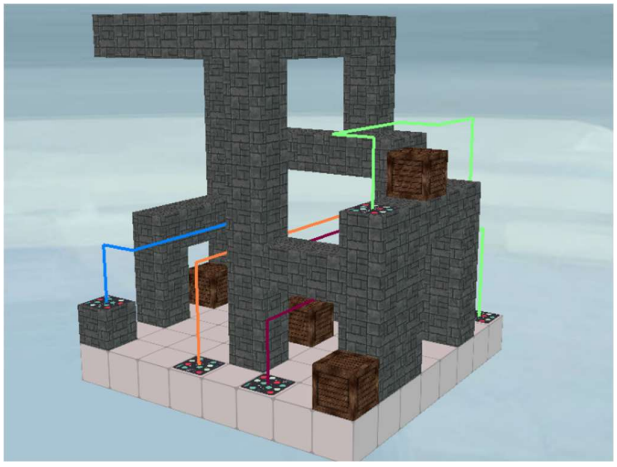


Fig. 46. Rutas de vuelo seguras. Sección de los puntos de destino.

VII. Diseño y Presentación del Sistema de Planificación de Rutas y Trayectorias

El desarrollo del sistema de planificación está divido en dos apartados: la parte conceptual, que desarrolla los criterios básicos de su funcionamiento; y la presentación final, que muestra cada una de las interfaces que ayudarán a construir el entorno, y generar las trayectorias de referencia para el desplazamiento de los drones al momento de su navegación.

A. *Diseño conceptual del sistema de planificación de rutas y trayectorias*

Un sistema de planificación genérico toma instrucciones de alto nivel y las interpreta durante su proceso de ejecución, para entregar en un tiempo finito (completitud algorítmica) uno o varios resultados, pudiendo ser satisfactorios o no, dependiendo de las condiciones de entrada (Fig. 47). Tales condiciones son: el mapa del entorno; las coordenadas de partida (x_s, y_s, z_s) y destino (x_f, y_f, z_f) de cada dron; las limitaciones y restricciones físicas; así como las condiciones de ruta (q) estudiadas en el apartado B de la sección V.

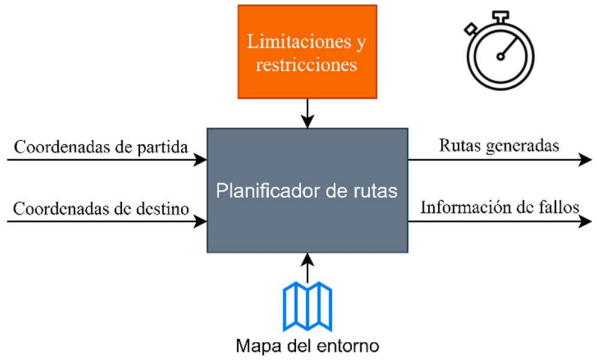


Fig. 47. Sistema de planificación de rutas.

Respecto a las respuestas del sistema, es probable que, por el tamaño y la disposición de los obstáculos del entorno, ciertas rutas no puedan efectuarse; razón por la que el sistema debe informar el fallo e indicar qué tipo de ruta, para las condiciones de origen y destino de un dron determinado, no le es posible generar.

Dado que las condiciones impuestas por el fabricante, en cuanto a las restricciones de movimiento, están íntimamente ligadas con los parámetros de control, la cinemática y la dinámica del dron, los resultados generados por el planificador de rutas se consideran "trayectorias". Con ellas se puede trabajar directamente sin necesidad de readaptarlas a través de técnicas de interpolación polinomial u optimizarlas como lo sugieren [29]–[31]. Claro está, el contexto es diferente, pues en el desarrollo de esta publicación se usará solamente el controlador y el sistema de posicionamiento automático integrado en el vehículo; no así, el uso de elementos externos de posicionamiento y control, a través de cámaras de visión como las empleadas por [31].

B. *Presentación del sistema de planificación de trayectorias*

El sistema de planificación de trayectorias parte del concepto indicado en la Fig. 47, posibilitando la construcción e introducción del mapa del entorno; el ingreso de coordenadas (x, y, z) de los drones soportados en la navegación; y un informador de fallos para las rutas que no fueron posibles generarlas dadas las condiciones.

Los parámetros de separación entre rutas y la altura de vuelo, están definidas como constantes en el software. Este sistema se alojó en GitHub para su uso y verificación: https://github.com/Areiban/Sistema-Planificacion-Tello-EDU.git

1) Interfaz de construcción del mapa del entorno: Entrega los recursos para el diseño básico de un entorno voxelizado con bloques de tamaño fijo. La Fig. 48 indica la ventana de construcción.

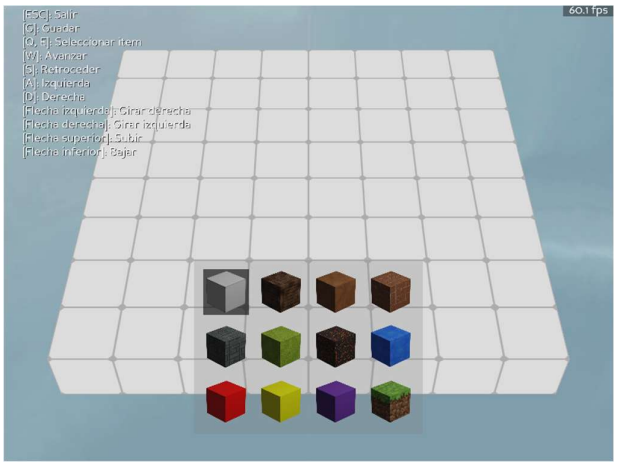


Fig. 48. Ventana de construcción del entorno.

2) Ingreso de coordenadas: Se compone de un cuadro de diálogo que acepta el ingreso de las coordenadas de inicio y destino de los drones, así como de una función que las valida y las estructura en un formato adecuado para el procesamiento del planificador de rutas. Véase la Fig. 49.

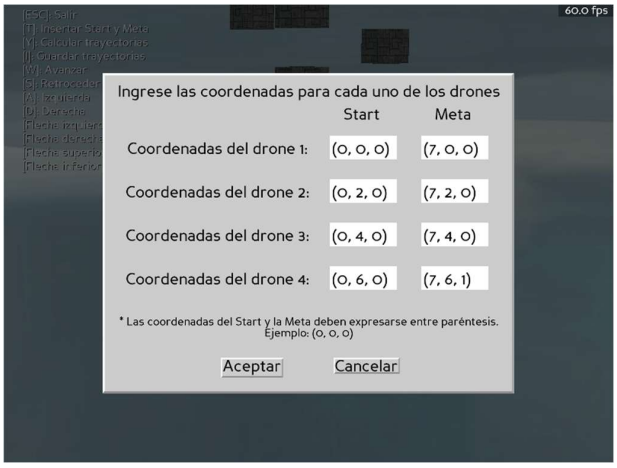


Fig. 49. Ventana de ingreso de coordenadas.

3) Reporte de fallos: En caso de no encontrar una o varias rutas libres de colisiones, para uno o varios drones en particular; ya sea por limitación del tamaño del entorno, o por la ubicación de las coordenadas de inicio y destino en puntos no accesibles, el sistema de planificación informará las rutas que no pudieron construirse, especificando el número de dron al que pertenecen. Véase la Fig. 50.

Fig. 50. Diálogo con el informe de fallo.

VIII. Comunicación, Comando y Supervisión De Los Drones

En esta sección se discuten los recursos empleados en la comunicación de los Tello EDU, seguido de la presentación de las funciones del SDK que se usan en la interacción y la designación de tareas. Finalmente se expone la estrategia implicada en el comando y la supervisión de los drones.

A. Comunicación

Los Tello EDU ofrecen dos formas de comunicación inalámbrica:

- A través de una conexión Wifi punto a punto con el ordenador, donde el dron actúa como servidor y administrador del enlace.
- Con el uso de un punto de acceso (router), que asignará direcciones IP estáticas según el número de drones a controlar, véase la Fig. 51.

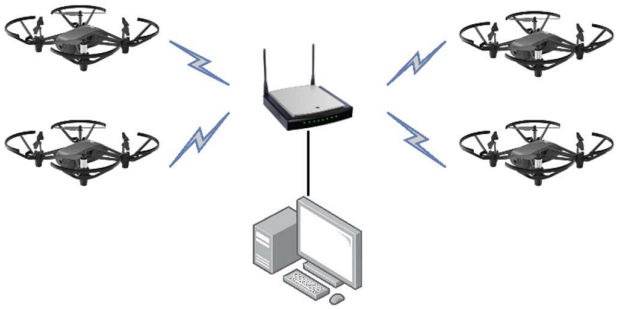


Fig. 51. Red de comunicación.

El protocolo de comunicación que se empleará en la transmisión de la información conforme al SDK, es el UDP (*User Datagram Protocol o Protocolo de Datagramas de Usuario*).

Para permitir la interacción con los drones, el ordenador central trabajará como cliente y servidor en diferentes instancias de tiempo (*multithreading*), enviando comandos través del puerto 8889 y recibiendo las informaciones de estado de los drones por el 8890.

La Tabla II detalla el registro de direcciones IP que se configurarán en el router según la dirección MAC de cada dron.

TABLA II. Registro De Direcciones En El Router

Direcciones MAC e IP	
Dirección MAC del dron	**Dirección IP asignada**
60:60:1f:fc:fb:e0	192.168.100.11
60:60:1f:fc:fb:62	192.168.100.12
60:60:1f:fc:f5:49	192.168.100.13
60:60:1f:fc:f4:fa	192.168.100.14

B. Funciones de interacción y designación de tareas

El SDK [34] provee un conjunto de funciones de control para la realización de diferentes acciones, caracterizadas por la devolución de mensajes tipo “ok” o “error” una vez finalizada la tarea. Estas funciones inhiben por completo al dron durante el proceso de ejecución, y no existe forma de cambiar su proceder hasta que no se haya recibido el mensaje de “ok” o “error” de la función particular.

En la Tabla III se listan las funciones que se usarán en el sistema de control de drones. Algunas de las funciones están referidas a la configuración del dron que se detalla en la Fig. 52.

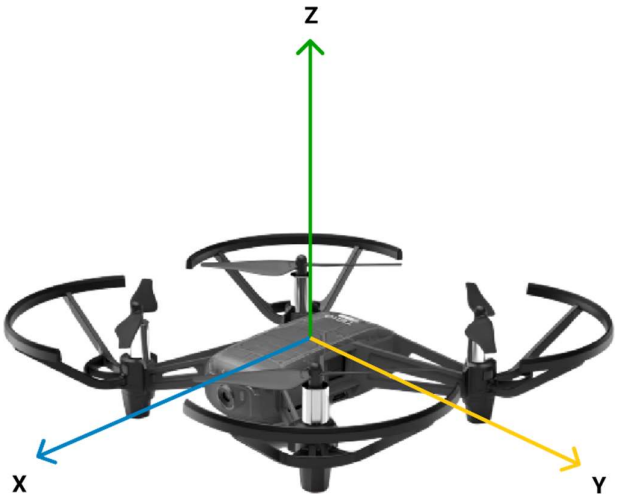


Fig. 52. Ejes del Tello EDU [35].

TABLA III. FUNCIONES DE CONTROL

Funciones del Tello EDU	
Funciones	**Acción**
command	Abre el canal de comunicación para la recepción y transmisión de comandos.
takeoff	Ejecuta el despegue del dron, elevándolo a una altura de 0.80m sobre el nivel del suelo u objeto que se encuentre por debajo del sensor de posicionamiento (Fig. 18).
up x	Realiza el ascenso del dron en un valor de x centímetros.
down x	Desciende al dron según el valor asignado en x.
forward x	Avanza al dron una distancia de x centímetros, en dirección de su cámara frontal (eje X). Véase la Fig. 52.
cw x	Gira el dron alrededor de su eje vertical Z (Fig. 52), en sentido horario, acorde al parámetro x asignado en grados.
ccw x	Gira el dron alrededor de su eje vertical Z (Fig. 52), en sentido antihorario, acorde al parámetro x asignado en grados.

C. *Comando y supervisión*

a) Comando: El comando de los drones implica transformar las rutas de vuelo obtenidas del planificador, y representarlas en una secuencia de funciones de control basadas en la definición de (9). Estas funciones serán administradas a través de una máquina de estados.

Dada la complejidad del sistema, cada vehículo a controlar tendrá su propio traductor de instrucciones y su máquina de estados. La máquina de estados será dependiente del mensaje emitido por el dron, acorde a las funciones de comando que se le suministren.

La Fig. 53 indica el diagrama de bloques que describe la estructura del sistema de comando.

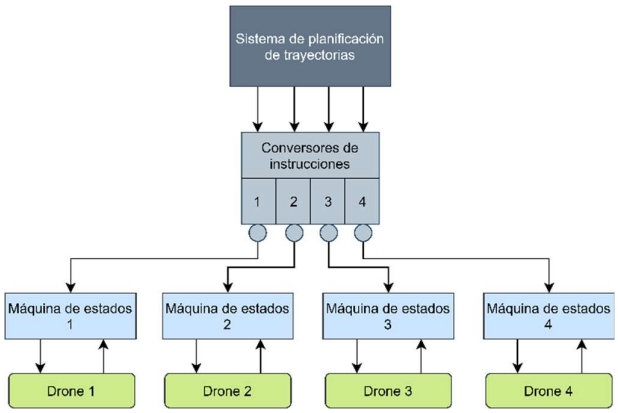


Fig. 53. Diagrama del sistema de control para la ejecución de trayectorias.

b) Supervisión: En lo referente a la supervisión de los drones, el procedimiento consistirá en la recopilación de los datos y su posterior visualización. Para ello se realizará el tratamiento de la información odométrica capturada por el puerto de comunicación 8890, a través de un receptor de estados, quien clasificará la trama de datos según las direcciones IP de cada dron.

Todos los datos odométricos procesados se enviarán a los supervisores asignados a cada dron, quienes formatearán los datos para su representación gráfica en la pantalla del ordenador.

La Fig. 54 presenta el diagrama de bloques que constituirá el proceso de captura y visualización de los datos odométricos.

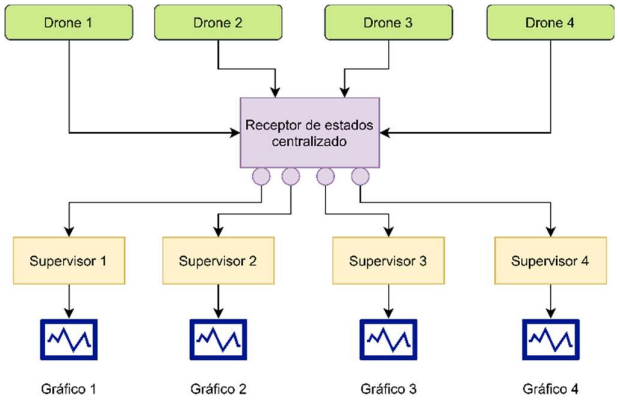


Fig. 54. Diagrama de bloques del sistema de supervisión.

Tanto el sistema de comando o control y el de supervisión, forman parte de un solo programa que se distribuye en dos procesos diferentes. Este programa principal recibe los resultados del sistema de planificación de rutas o un arreglo de coordenadas fijadas manualmente por el usuario (no recomendado), y las envía hacia el sistema de control para la asignación de tareas. Durante la ejecución del programa se coordinará todos los procesos internos incluyendo la supervisión de los drones.

En la Fig. 55 se indica la ventana de salida del sistema de supervisión compuesto por cinco graficadores: cuatro para la visualización de alturas de los drones (cm vs s); y uno para la observación de las trayectorias ejecutadas. También se incluye un botón para el guardado de la información odométrica en dos tipos de archivo: CSV y XLSX.

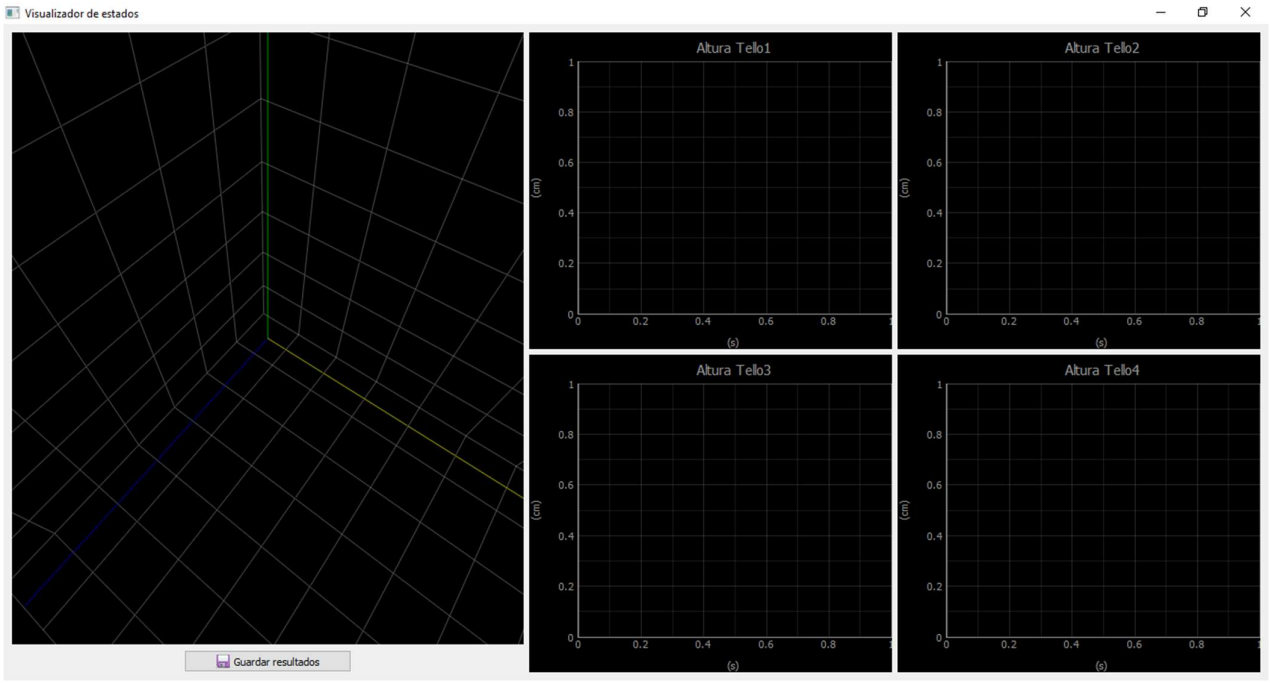


Fig. 55. Visualizador de estados de los drones (Tello EDU).

IX. Experimentos y Resultados

En esta sección se presenta y analiza el comportamiento de los drones durante la ejecución de las trayectorias de navegación.

El análisis está dado por el cálculo de la raíz cuadrada del error cuadrático medio (*RMSE – Root Mean Squared Error*) (11), expresión que informa el valor eficaz de las desviaciones de los puntos de las trayectorias ejecutadas respecto a los valores estipulados por el sistema de planificación.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(valor\ real_i - estimado_i)^2}{N}} \qquad (11)$$

La navegación se realiza en un ambiente confinado de obstáculos ficticios (Fig. 56), representado por tres configuraciones diferentes del entorno que se detallan en los apartados A, B y C de esta sección. Sus dimensiones se restringen a los 3m de largo, por 3m de ancho, por 2m de altura. El tamaño de los vóxeles y las regiones de seguridad de los drones se definen en 50cm por lado, tal como se explicó en la sección VI apartado B.

Fig. 56. Navegación de los drones en el entorno.

Los experimentos están realizados posterior a la actualización de firmware (versión 01.04.91.01) y calibración completa de las aeronaves, por empleo de la aplicación presentada en la Fig. 21.

A. Comportamiento en el entorno 1

La Fig. 57 detalla la configuración del entorno y las rutas de navegación para las coordenadas de inicio y destino que se indican en la Tabla IV.

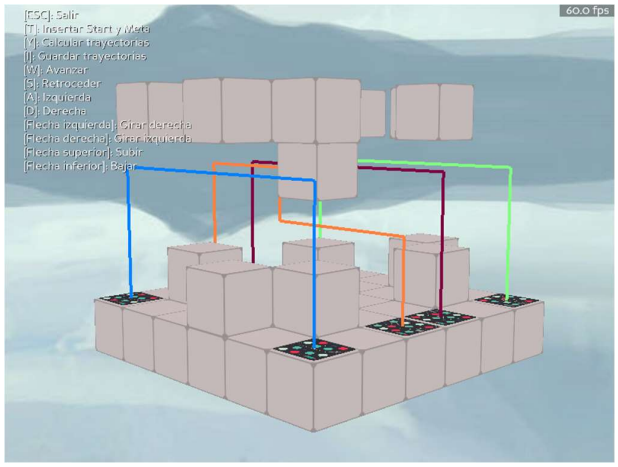


Fig. 57. Entorno 1.

TABLA IV. Coordenadas De Inicio Y Destino Del Entorno 1

Coordenadas de inicio	Coordenadas de destino
(0, 0, 0)	(5, 0, 0)
(0, 2, 0)	(5, 2, 0)
(0, 3, 0)	(5, 3, 0)
(0, 5, 0)	(5, 5, 0)

1) Longitudes de las rutas: En la Tabla V se presenta las longitudes de las rutas de navegación, cuyas magnitudes están dadas en metros. El orden de presentación se establece según un barrido de izquierda a derecha en la Fig. 57.

TABLA V. LONGITUDES DE LAS RUTAS DE VUELO DEL ENTORNO 1

Rutas de vuelo referenciales	Longitud
Ruta 1	5m
Ruta 2	5m
Ruta 3	5m
Ruta 4	5m

2) Rutas ejecutadas: El resumen de la información adquirida por el sistema de supervisión en la Fig. 69 del Anexo A se detalla en la Fig. 58. Los trazos de color, conforme a la Fig. 57, corresponden a las trayectorias desarrolladas por los drones durante la navegación.

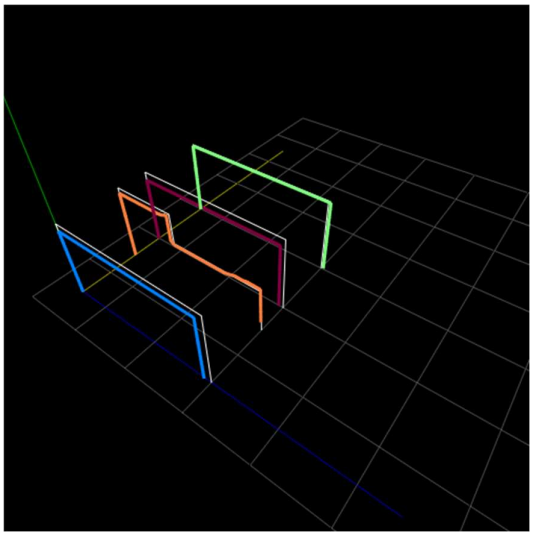


Fig. 58. Trayectorias ejecutadas por los drones en el entorno 1. Las rutas de referencia se especifican en color blanco.

3) Tiempos de ejecución de las rutas de navegación: Se expone en la Fig. 59. Los colores se corresponden con las trayectorias en la Fig. 58, y las numeraciones sobre las barras, valores de tiempo expresados en segundos.

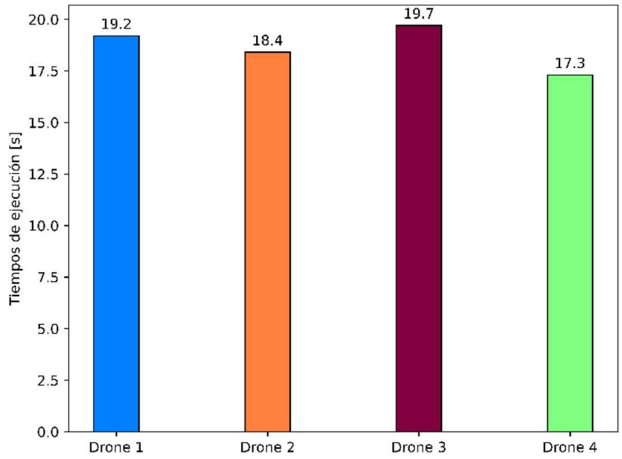


Fig. 59. Tiempos de ejecución de las rutas de navegación del entorno 1.

4) Errores en la ejecución de las rutas: La Fig. 60 presenta los errores RMSE expresados en centímetros para las componentes (x, y, z) de las trayectorias trazadas por los drones.

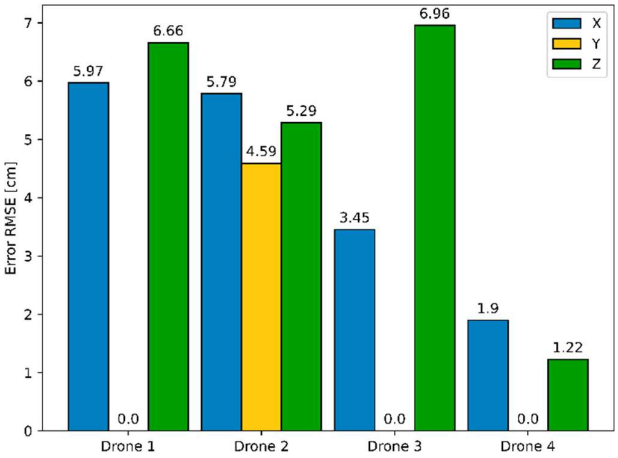


Fig. 60. Errores RMSE de la navegación en el entorno 1.

B. Comportamiento en el entorno 2

La configuración del entorno con el que se trabajará en este apartado corresponde a la representación en la Fig. 61. Las coordenadas de inicio y destino se especifican en la Tabla VI.

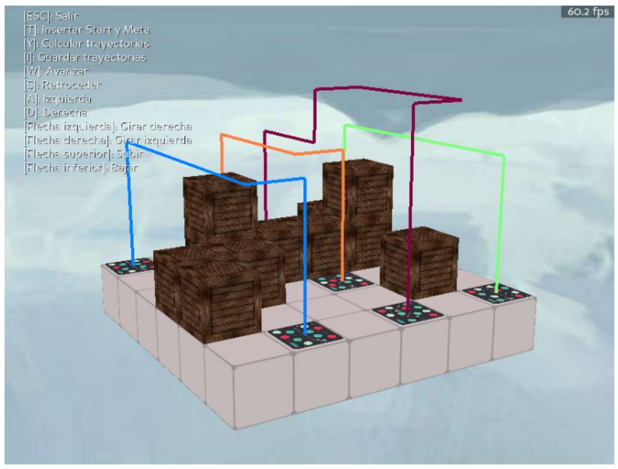


Fig. 61. Entorno 2.

TABLA VI. COORDENADAS DE INICIO Y DESTINO DEL ENTORNO 2

Coordenadas de inicio	Coordenadas de destino
(0, 0, 0)	(5, 1, 0)
(0, 2, 0)	(3, 3, 0)
(0, 3, 0)	(5, 3, 0)
(0, 5, 0)	(5, 5, 0)

1) Longitudes de las rutas: La Tabla VII indica las longitudes de las rutas del entorno 2. Su orden está dado de izquierda a derecha según la Fig. 61.

TABLA VII. LONGITUDES DE LAS RUTAS DE VUELO DEL ENTORNO 2

Rutas de vuelo referenciales	Longitud
Ruta 1	5.5m
Ruta 2	4.5m
Ruta 3	7m
Ruta 4	5m

2) Rutas ejecutadas: La Fig. 62 presenta las trayectorias trazadas por los drones durante el proceso de navegación a partir de los datos capturados por el supervisor. Su detalle se muestra en la Fig. 70 del Anexo B. Los colores indican la ejecución de las rutas de referencia especificadas en la Fig. 61.

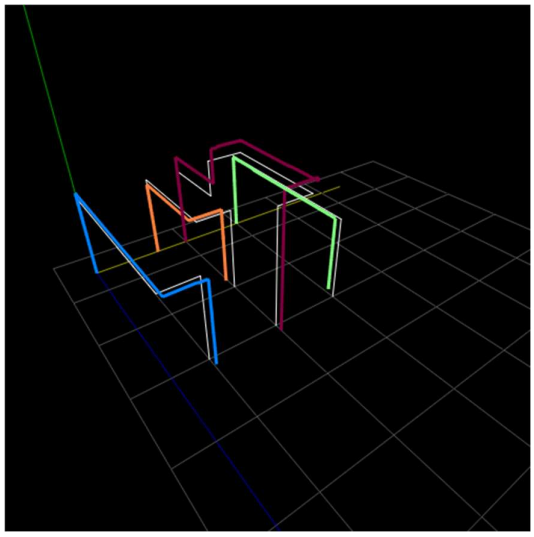


Fig. 62. Trayectorias ejecutadas por los drones en el entorno 2. Las rutas de referencia se especifican en color blanco.

3) Tiempos de ejecución de las rutas de navegación: La Fig. 63 detalla los valores de tiempo en el desarrollo de las rutas. Los colores clasificados según la Fig. 62 distinguen a cada una de ellas.

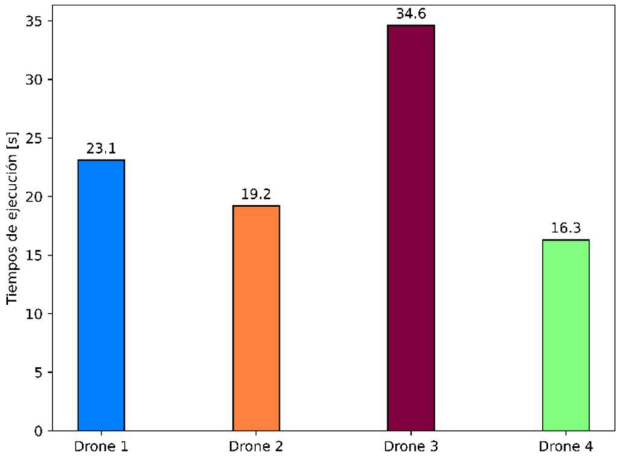


Fig. 63. Tiempos de ejecución de las rutas de navegación del entorno 2.

4) Errores en la ejecución de las rutas: Se presenta en la Fig. 64. Los valores sobre las barras especifican los errores RMSE en la ejecución de las rutas.

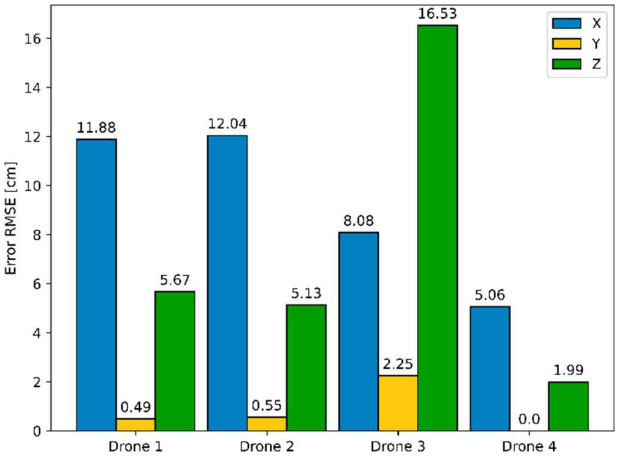


Fig. 64. Errores RMSE de la navegación en el entorno 2.

C. Comportamiento en el entorno 3

En esta configuración del entorno se omiten los obstáculos para el sistema de planificación. Su resultado se indica en la Fig. 65. Las coordenadas de inicio y destino se presentan en la Tabla VIII.

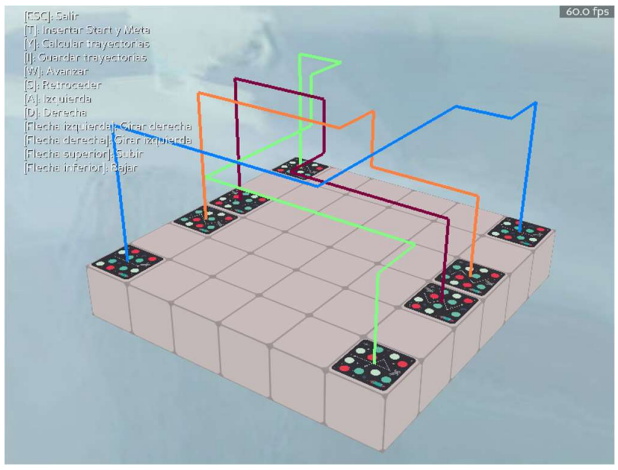


Fig. 65. Entorno 3.

TABLA VIII. COORDENADAS DE INICIO Y DESTINO DEL ENTORNO 3

Coordenadas de inicio	Coordenadas de destino
(0, 0, 0)	(5, 5, 0)
(0, 2, 0)	(5, 3, 0)
(0, 3, 0)	(5, 2, 0)
(0, 5, 0)	(5, 0, 0)

1) Longitudes de las rutas: Se exponen en la Tabla IX. El orden de su presentación sigue el mismo criterio que en apartados anteriores, comenzando desde la esquina superior izquierda de la plataforma en la Fig. 65.

TABLA IX. LONGITUDES DE LAS RUTAS DE VUELO DEL ENTORNO 3

Rutas de vuelo referenciales	Longitud
Ruta 1	7.5m
Ruta 2	5.5m
Ruta 3	5.5m
Ruta 4	7.5m

2) Rutas ejecutadas: La Fig. 66 presenta las trayectorias ejecutadas por los drones (detalle en la Fig. 71 del Anexo C) para las rutas de referencia del entorno 3 (Fig. 65).

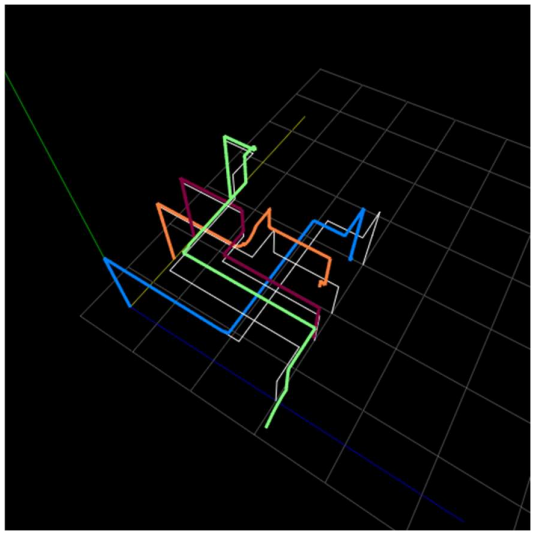


Fig. 66. Trayectorias ejecutadas por los drones en el entorno 3. Las rutas de referencia se especifican en color blanco.

3) Tiempos de ejecución de las rutas de navegación: Los tiempos desarrollados por los drones durante la ejecución de sus rutas de vuelo se indican en la Fig. 67.

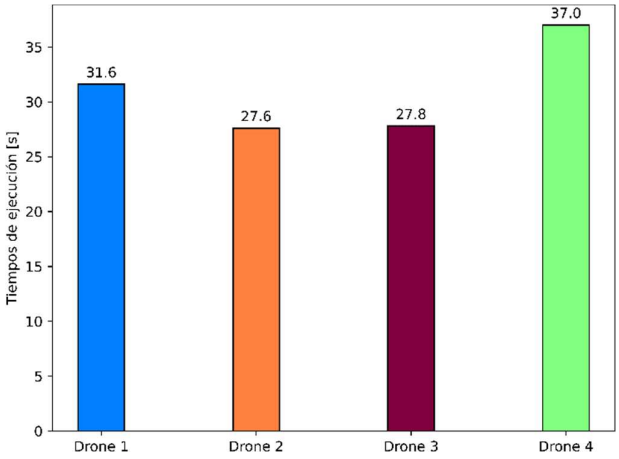


Fig. 67. Tiempos de ejecución de las rutas de navegación del entorno 3.

4) Errores en la ejecución de las rutas: Los errores de ejecución durante la navegación por el entorno 3 se detallan en la Fig. 68.

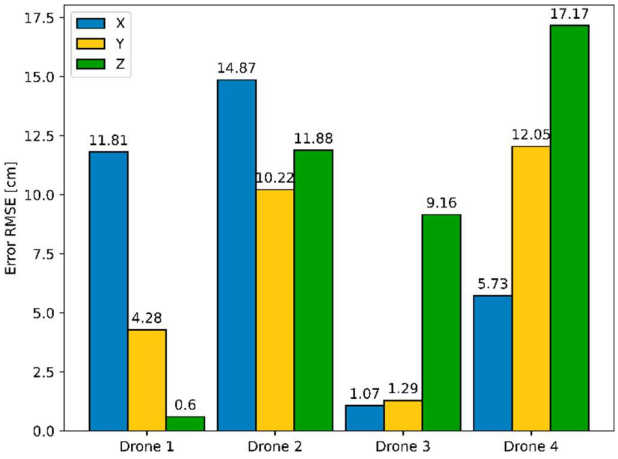


Fig. 68. Errores RMSE de la navegación en el entorno 3.

X. CONCLUSIONES

Los resultados experimentales de la sección IX indican que la variabilidad de las rutas de vuelo, en cuanto a su longitud y dirección, es el principal factor de los tiempos de ejecución de las trayectorias, y de las desviaciones de los valores observados respecto a los estimados.

De acuerdo con el entorno 3, los cambios de dirección, reflejados en mayor medida en las trayectorias 2 y 4, provocan que los valores RMSE sean superiores a sus semejantes de los entornos 1 y 2. Tales situaciones conducen a errores en el posicionamiento de las aeronaves en el entorno, cuyos efectos, conforme a la Fig. 66, se distinguen notoriamente en la altitud y en el proceso de aterrizaje de los vehículos.

Si el entorno se incrementa en tamaño y densidad de obstáculos, se generarían trayectorias complejas aumentando sus distancias y sus valores RMSE. Esto causaría que los drones sean incapaces de completar las trayectorias, y en algún punto de ellas cambiar de rumbo, terminando en colisión con objetos del entorno o aeronaves circundantes.

Los efectos acumulativos del RMSE se presentan fundamentalmente por el sistema de posicionamiento y orientación embebida de los drones, debido a que sus sistemas de control se orientan conforme a cambios de dirección y aceleración.

Si se optara por sistemas fijos de posicionamiento externo, y con sistemas de control adaptados a las condiciones cinemáticas y dinámicas de los drones (excluyendo al controlador embebido), los errores RMSE disminuirían, pero aumentarían la complejidad y los costos de la implementación.

Para entornos confinados de dimensiones similares a los tratados en los experimentos, y márgenes de separación entre rutas mayores o iguales a los propuestos, los errores acumulativos de los sistemas de control embebido no causarían mayores problemas a los presentados ni derivarían en colisiones de las aeronaves.

REFERENCIAS

[1] W. Honig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, y N. Ayanian, «Trajectory Planning for Quadrotor Swarms», IEEE Trans. Robot., vol. 34, n.º 4, pp. 856-869, ago. 2018, doi: 10.1109/TRO.2018.2853613.

[2] A. Kushleyev, D. Mellinger, C. Powers, y V. Kumar, «Towards a swarm of agile micro quadrotors», Auton. Robots, vol. 35, n.º 4, pp. 287-300, nov. 2013, doi: 10.1007/s10514-013-9349-9.

[3] W. Lucia, M. Sznaier, y G. Franze, «An obstacle avoidance and motion planning Command Governor based scheme: The Qball-X4 quadrotor case of study», en 53rd IEEE Conference on Decision and Control, Los Angeles, CA, USA, dic. 2014, pp. 6135-6140, doi: 10.1109/CDC.2014.7040350.

[4] S. Grzonka, G. Grisetti, y W. Burgard, «A Fully Autonomous Indoor Quadrotor», IEEE Trans. Robot., vol. 28, n.º 1, pp. 90-100, feb. 2012, doi: 10.1109/TRO.2011.2162999.

[5] F. Gao y S. Shen, «Online quadrotor trajectory generation and autonomous navigation on point clouds», en 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Lausanne, Switzerland, oct. 2016, pp. 139-146, doi: 10.1109/SSRR.2016.7784290.

[6] C. Copot, A. Hernandez, T. Thoa Mac, y R. De Keyse, «Collision-free path planning in indoor environment using a quadrotor», en 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, ago. 2016, pp. 351-356, doi: 10.1109/MMAR.2016.7575160.

[7] W. Giernacki, M. Skwierczynski, W. Witwicki, P. Wronski, y P. Kozierski, «Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering», en 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, ago. 2017, pp. 37-42, doi: 10.1109/MMAR.2017.8046794.

[8] K. S. Yakovlev, D. A. Makarov, y E. S. Baskin, «Automatic path planning for an unmanned drone with constrained flight dynamics», Sci. Tech. Inf. Process., vol. 42, n.º 5, pp. 347-358, dic. 2015, doi: 10.3103/S0147688215050093.

[9] F. Li, S. Zlatanova, M. Koopman, X. Bai, y A. Diakité, «Universal path planning for an indoor drone», Autom. Constr., vol. 95, pp. 275-283, nov. 2018, doi: 10.1016/j.autcon.2018.07.025.

[10] J. D. Hunter, «Matplotlib: A 2D Graphics Environment», Comput. Sci. Eng., vol. 9, n.º 3, pp. 90-95, 2007, doi: 10.1109/MCSE.2007.55.

[11] M. Musy, G. Dalmasso, y B. Sullivan, «A python module for scientific visualization and analysis of 3D objects and point clouds based on VTK (Visualization Toolkit)», Zenodo, feb. 2019, doi: 10.5281/zenodo.2561402.

[12] J.-C. Latombe, Robot Motion Planning. New York, NY, USA: Springer, 1991.

[13] S. M. LaValle, Planning Algorithms. Cambridge , NY, USA: Cambridge, 2006.

[14] H. Choset et al., Principles of Robot Motion: Theory, Algorithms, and Implementations. Cambridge, MA, USA: MIT Press, 2005.

[15] F. Bullo y S. L. Smith, Lectures on Robotic Planning and Kinematics. Santa Barbara, CA, USA: University of California, 2016.

[16] R. Siegwart, I. R. Nourbakhsh, y D. Scaramuzza, Introduction to Autonomous Mobile Robots, Segunda. Cambridge, MA, USA: MIT Press, 2011.

[17] C. Mahulea, M. Kloetzer, y R. González, Path Planning of Cooperative Mobile Robots Using Discrete Event Models. Piscataway, NJ, USA: IEEE Press, 2020.

[18] S. S. Ge y F. L. Lewis, Eds., Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications. Boca Raton, FL, USA: CRC Press, 2006.

[19] S. Craw, «Manhattan Distance», en Encyclopedia of Machine Learning, C. Sammut y G. I. Webb, Eds. Boston, MA, USA: Springer, 2010, pp. 639-639.

[20] A. Y. Bhargava, Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People. Shelter Island, NY, USA: Manning, 2016.

[21] R. Uehara, First Course in Algorithms Through Puzzles. Singapore, SG: Springer, 2019.

[22] U. Manber, Introduction to Algorithms: a Creative Approach. Reading, MA, USA: Addison-Wesley, 1989.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, y C. Stein, Eds., Introduction to Algorithms, Tercera. Cambridge, MA, USA: MIT Press, 2009.

[24] N. Correll, Introduction to Autonomous Robots. Boulder, CO, USA: University of Colorado, 2016.

[25] K. G. Murty, Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. Ann Arbor, MI, USA: World Scientific, 2014.

[26] A. Tsourdos, B. A. White, y M. Shanmugavel, Cooperative Path Planning of Unmanned Aerial Vehicles. West Sussex, UK: Wiley, 2011.

[27] F. Fahimi, Autonomous Robots: Modeling, Path Planning, and Control. New York, NY, USA: Springer, 2008.

[28] HobbyKing, «Q-BOT Micro Quadcopter». https://hobbyking.com/en_us (accedido oct. 10, 2020).

[29] Z. Liu y Y. Wang, «Trajectory planning and tracking for maneuverable quadrotors», en 2018 Chinese Control And Decision Conference (CCDC), Shenyang, jun. 2018, pp. 2900-2905, doi: 10.1109/CCDC.2018.8407620.

[30] C. Richter, A. Bry, y N. Roy, «Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments», en Robotics Research, vol. 114, M. Inaba y P. Corke, Eds. Cham: Springer International Publishing, 2016, pp. 649-666.

[31] D. Mellinger y V. Kumar, «Minimum snap trajectory generation and control for quadrotors», en 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, may 2011, pp. 2520-2525, doi: 10.1109/ICRA.2011.5980409.

[32] M. W. Spong, S. Hutchinson, y M. Vidyasagar, Robot Modeling and Control, Segunda. Hoboken, NJ, USA: Wiley, 2020.

[33] L. Biagiotti y C. Melchiorri, Trajectory Planning for Automatic Machines and Robots. Heidelberg, DE: Springer, 2008.

[34] «Tello SDK 2.0 User Guide». Ryze Technology, [En línea]. Disponible en: https://www.ryzerobotics.com/tello-edu/downloads.

[35] Ryze Technology, «Tello Official Website-Shenzhen Ryze Technology Co.,Ltd.», RYZE. https://www.ryzerobotics.com/tello-edu (accedido oct. 13, 2020).

[36] The Hybrid Group, «Gobot - Golang framework for robotics, drones, and the Internet of Things (IoT)», Gobot. https://gobot.io/ (accedido oct. 13, 2020).

[37] «Tello Mission Pad User Guide». Ryze Technology, [En línea]. Disponible en: https://www.ryzerobotics.com/tello-edu/downloads.

[38] «Tello User Manual». Ryze Technology, [En línea]. Disponible en: https://www.ryzerobotics.com/tello-edu/downloads.

[39] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, y M. J. Marín-Jiménez, «Automatic generation and detection of highly reliable fiducial markers under occlusion», Pattern Recognit., vol. 47, n.º 6, pp. 2280-2292, jun. 2014, doi: 10.1016/j.patcog.2014.01.005.

[40] COEX, «ArUco Marker Detection · Clover», COEX Clover. https://clover.coex.tech/en/ (accedido oct. 15, 2020).

A. Datos odométricos de los drones en la navegación del entorno 1

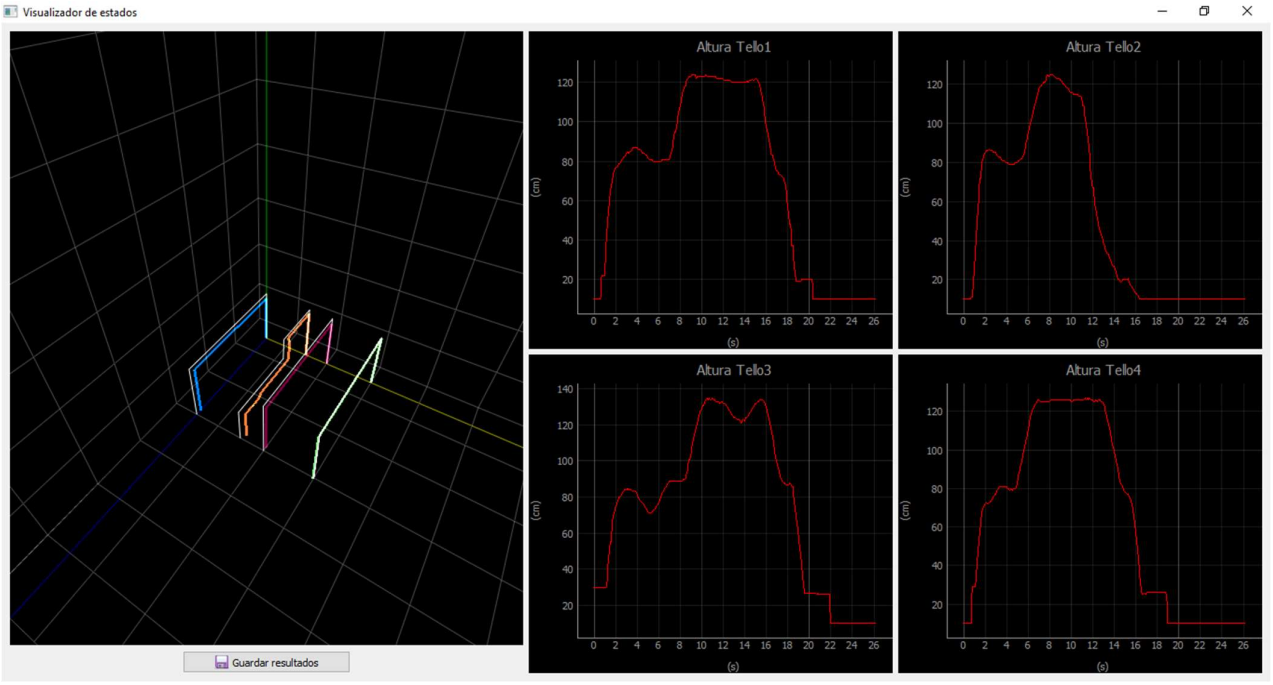


Fig. 69. Visualización de los datos odométricos en la navegación del entorno 1.

B. Datos odométricos de los drones en la navegación del entorno 2

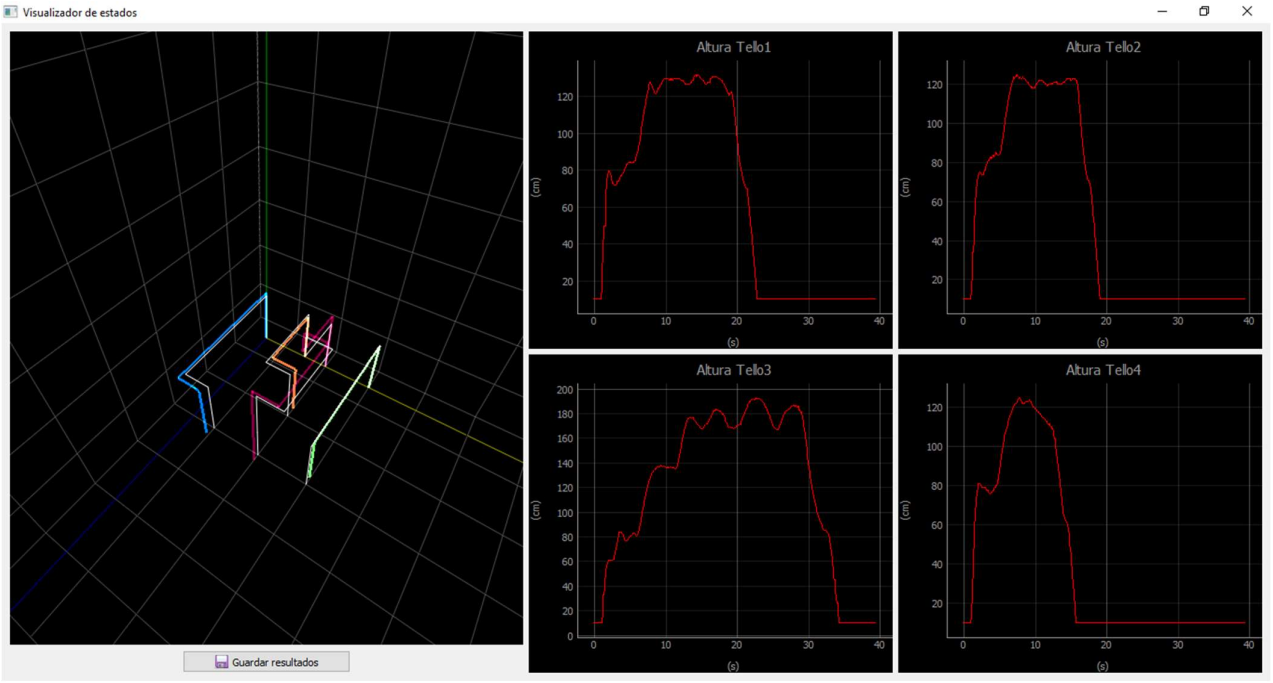


Fig. 70. Visualización de los datos odométricos en la navegación del entorno 2.

C. Datos odométricos de los drones en la navegación del entorno 3

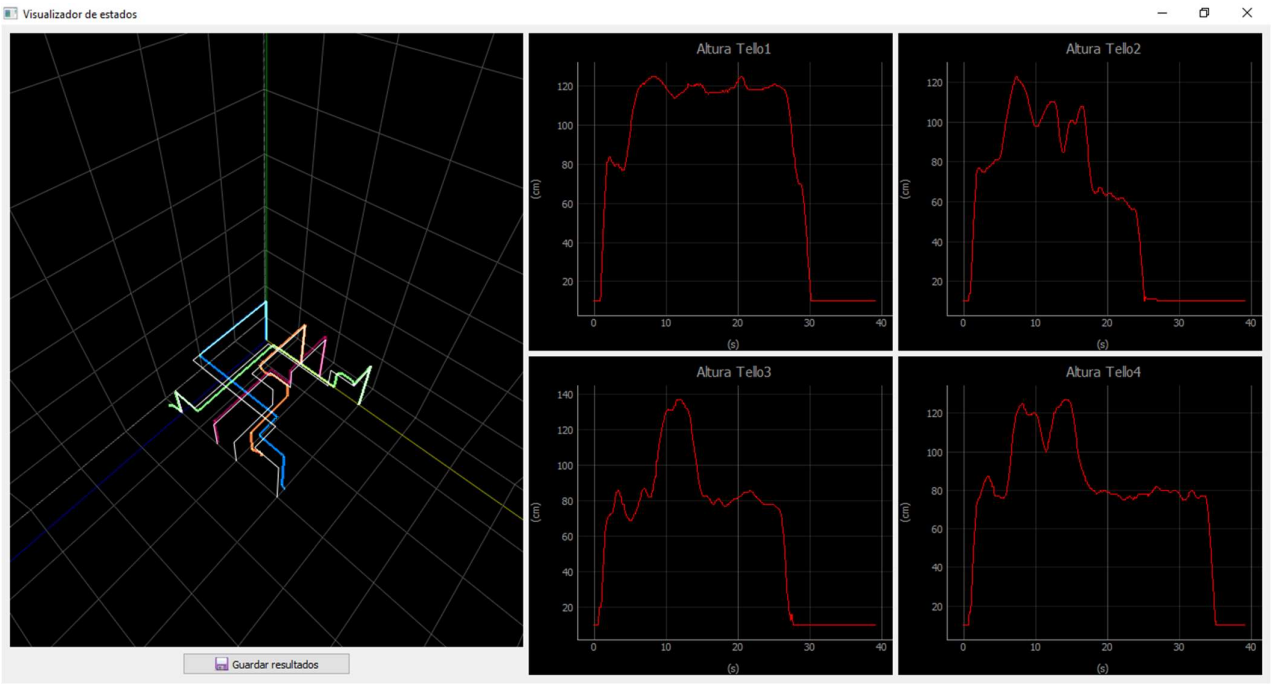


Fig. 71. Visualización de los datos odométricos en la navegación del entorno 3.