CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Planificador de trayectorias seguro y eficiente en entornos de construcción considerando distancia de Manhattan

Guillermo Gil, Antonio Moreno, José Antonio Cobano*, Fernando Caballero, Luis Merino

Service Robotics Lab, Universidad Pablo de Olavide (Sevilla), Ctra. de Utrera km. 1, 41013 Sevilla, España.

To cite this article: Gil, G., Moreno, A., Cobano, J.A., Caballero, F., Merino, L. 2026. Safe and efficient trajectory planner in construction environments considering Manhattan distance. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13883

Resumen

Este artículo presenta un planificador de trayectorias 3D en entornos de construcción, basado en campos de distancias Manhattan (MDF) en lugar de la distancia euclídea (EDF) clásica. El planificador combina un algoritmo A* consciente del coste (CA-A*) que integra la distancia a obstáculos mediante la métrica L1, seguido de un posprocesado con polinomios de grado 5 optimizados con Ceres Solver. Las simulaciones en un entorno realista de construcción muestran que el uso de la distancia Manhattan reduce significativamente el número de nodos expandidos y el tiempo de cómputo respecto a L2, manteniendo trayectorias seguras y eficientes. Además, el posprocesado reduce los ángulos de giro y mejora la suavidad sin alejarse significativamente de la trayectoria inicial.

Palabras clave: robótica aérea, planificación de trayectorias, navegación y control, sistemas autónomos.

Safe and efficient trajectory planner in construction environments considering Manhattan distance

Abstract

This paper presents a 3D trajectory planner in construction environments, based on Manhattan Distance Fields (MDF) instead of the classic Euclidean Distance Fields (EDF). The planner combines a cost-aware A* algorithm (CA-A*) that integrates obstacle distance using the L1 metric, followed by a post-processing stage with 5th-degree polynomials optimized with Ceres Solver. Simulations in a realistic construction environment show that using Manhattan distance significantly reduces the number of expanded nodes and computation time compared to L2, while maintaining safe and efficient trajectories. Furthermore, the post-processing reduces turning angles and improves smoothness without deviating significantly from the initial trajectory.

Keywords: aerial robotics, trajectory planning, navigation and control, autonomous systems.

1. Introducción

El sector de la construcción constituye un pilar fundamental de la economía de la Unión Europea (UE), ya que representa 18 millones de puestos de trabajo y contribuye con casi el 9 % del PIB. Actualmente, este sector enfrenta múltiples desafíos, frente a los cuales las tecnologías digitales se perfilan como una vía idónea para abordarlos de manera más eficiente. No obstante, la construcción es uno de los sectores menos digitalizados de la economía, por lo que la integración de estas tecnologías podría contribuir decisivamente a resolver sus problemas actuales. Entre dichas tecnologías destacan las plataformas robóticas, ampliamente utilizadas e integradas en muchos otros sectores en la actualidad. Aunque el despliegue de robots móviles y drones en la construcción ha crecido significativamente en la última década (Follini et al., 2020; Kim and Peavy, 2022), la mayoría de las soluciones robóticas disponi-

*Autor para correspondencia: jacobsua@upo.es

[1] https://www.hilti.es/content/hilti/E2/ES/es/business/business/trends/jaibot.html

bles en el mercado se basan en la teleoperación por parte de los trabajadores, como es el caso del Hilti Jaibot[1]. Por lo tanto, la navegación autónoma de robots en estos entornos constituye un área en la que aún queda trabajo por realizar. Una de las tareas clave para lograr este objetivo es la planificación de trayectorias seguras.

La representación del entorno para planificación es fundamental y en los últimos años están siendo muy usados los campos de distancia euclídea como una representación muy eficiente (Jones et al., 2006). Proporcionan la distancia al obstáculo más cercano para cualquier punto del espacio. Además poseen propiedades analíticas que pueden ser explotadas por los planificadores (Cobano et al., 2022). En la literatura hay varios trabajos publicados que usan estos campos de distancias para generar trayectorias seguras (Oleynikova et al., 2016; Gao et al., 2018; Cobano et al., 2026).

Por otro lado, el tipo de configuración de los escenarios de construcción, mapas alineados con ejes, puede favorecer el uso de la distancia de Manhattan en lugar de la distancia euclídea en estos campos de distancias y en algoritmos heurísticos para la planificación de trayectorias seguras. Este artículo presenta una planificador 3D que hace uso de la distancia de Manhattan en lugar de la euclídea para generar trayectorias seguras y suaves. El planificador consta de un algoritmo A* basado en Cobano et al. (2022) para generar una trayectoria inicial y una segunda fase de post-procesamiento donde se suaviza la trayectoria inicial. Las simulaciones presentadas muestran los buenos resultados en cuanto a eficiencia, seguridad y suavidad al considerar la distancia de Manhattan.

2. Evaluación del algoritmo A* bajo métricas L1 y L2

El algoritmo A* es uno de los más empleados para planificación de caminos en entornos discretos y continuos. Su eficiencia depende de la heurística empleada, que determina la dirección de la búsqueda y exploración hacia el objetivo. En el caso de escenario 3D, donde cada nodo tiene 26 vecinos y los espacios pueden contener obstáculos irregulares, la elección de la métrica puede afectar significativamente a la cantidad de nodos expandidos y el tiempo total de cómputo.

Esta sección realiza una evaluación teórica del funcionamiento del algoritmo A* bajo dos métricas comunes: la distancia Manhattan (L1) y la distancia euclídea (L2). Ambas se emplean tanto para el cálculo del coste acumulado $g(n)$ como de la heurística $h(n)$.

2.1. Formulación matemática del problema

El entorno tridimensional se modela como una cuadrícula 3D $\mathcal{G} \subset \mathbb{Z}^3$, donde cada nodo o celda representa un punto (x, y, z).

La función de coste que considera el algoritmo A* para seleccionar los nodos a expandir es:

$$f(n) = g(n) + h(n) \tag{1}$$

donde $g(n)$ es el coste acumulado desde el nodo inicial hasta el nodo n, y $h(n)$ es una estimación heurística del coste del nodo n hasta el objetivo o final de la trayectoria.

El algoritmo termina cuando el nodo objetivo es alcanzado o cuando no existen más nodos disponibles para explorar.

2.2. Heurísticas basadas en métricas L1 y L2

Los algoritmos heurísticos pueden considerar dos métricas clásicas en el espacio tridimensional: L1, Manhattan (2) y L2, Euclídea (3).

$$d_M(p, q) = |x_p - x_q| + |y_p - y_q| + |z_p - z_q|, \tag{2}$$

$$d_E(p, q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2 + (z_p - z_q)^2}, \tag{3}$$

Se estudia el uso de la métrica tanto en el coste acumulado $g(n)$ como en la heurística $h(n)$ en ambos casos:

$$\text{Caso L1:} \quad g(n, n') = d_M(n, n'), \quad h(n) = d_M(n, goal) \tag{4}$$

$$\text{Caso L2:} \quad g(n, n') = d_E(n, n'), \quad h(n) = d_E(n, goal) \tag{5}$$

A continuación se muestra como ambas métricas cumplen la admisibilidad y consistencia.

Admisibilidad. Una heurística es **admisible** si nunca sobreestima el coste real:

$$h(n) \leq h^*(n), \quad \forall n \in \mathcal{G} \tag{6}$$

Tanto la heurística Manhattan como la euclídea son admisibles cuando se usan junto con su respectiva métrica de coste. En los algoritmos heurísticos de búsqueda informada, como el algoritmo A*, usar una heurística admisible asegura que siempre se encontrará el camino más corto o la solución óptima.

Consistencia. Una heurística es **consistente** si cumple:

$$h(n) \leq c(n, n') + h(n'), \quad \forall n, n' \in \mathcal{G} \tag{7}$$

Ambas métricas satisfacen esta propiedad, garantizando que A* no reexpande nodos y que mantiene la optimalidad respecto a la métrica empleada.

3. Representación del entorno

La representación del entorno es crucial en los sistemas de navegación para la planificación de trayectorias en 3D. Los campos de distancias se presentan como una buena opción en los últimos años que además generan trayectorias seguras al considerar la distancia a los obstáculos más cercanos. Generalmente se usa la distancia euclídea con los denominados *Euclidean Distance Field* (Cobano et al., 2022; Oleynikova et al., 2016; Zhou et al., 2019). Estas representaciones pueden no adaptarse bien a problemas de planificación en tiempo real por la actualización del mapa. Cuando el entorno cambia, es necesario volver a recorrer la nube de puntos y calcular la distancia euclídea a todos los obstáculos cercanos para quedarse con la más pequeña. Esto conlleva una importante carga computacional. En este trabajo se trabaja con la distancia de Manhattan, que aporta la posibilidad de representar la distancia al obstáculo más cercano en cada celda con bits sin hacer uso de decimales. Esto puede repercutir satisfactoriamente en que la actualización del mapa sea más rápida.

Por tanto, la representación del entorno usada en este trabajo está basada en campos de distancias Manhattan (Manhattan Distance Field, MDF) como alternativa al clásico Euclidean Distance Field (EDF). El MDF presenta varias ventajas respecto al EDF relacionadas con el tipo de operaciones para

su cálculo y, por tanto, con el coste computacional. El MDF considera la suma de diferencias absolutas por eje y representamos cada celda de la cuadrícula con 64 bits. La distancia al obstáculo más cercano se codifica como el número de bits menos significativos puestos a 1: cuantos más bits activos, mayor es la distancia. La ventaja esta representación se debe a que usando una máscara de bits para cada celda ocupada por un obstaculo, por la información obtenida de la nube de puntos, será posible actualizar la proximidad hasta el obstáculo más cercano para cualquier celda con una simple operación AND en cada celda de la máscara. Esto evitaría recorrer todos los puntos para recalcular distancias y reduce considerablemente la carga computacional. La Figura 1 muestra un ejemplo de cómo se aplica una máscara a un MDF en 2D con 4 bits por celda.

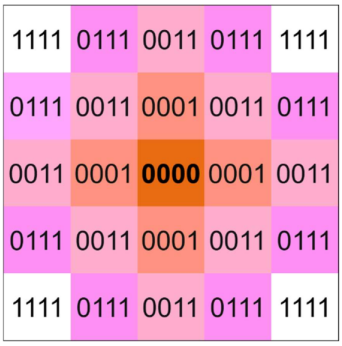

Figura 1: Máscara basada en la distancia de Manhattan a partir de la celda con un obstáculo (0000)y que usa el MDF.

La generación del MDF parte de nubes de puntos obtenidas de archivos .bt que describen el entorno. En primer lugar, se inicializan todas las celdas de la cuadrícula con todos los bits a 1 y se aplica la máscara correspondiente a cada obstáculo detectado por la nube de puntos. Para hacer viable el proceso en escenarios de gran tamaño, se paraleliza la aplicación de la máscara y la transformación del formato de distancia a bits, asegurando mediante operaciones atómicas que múltiples hilos no accedan simultáneamente a la misma dirección de memoria. Indicar también que al usar la máscara, estamos usando una distancia truncada, es decir, consideramos la distancia a los obstáculos hasta una distancia determinada.

4. Planificador de trayectorias

Esta sección describe el planificador propuesto. Se divide en dos bloques: el primero basado en un planificador heurístico, algoritmo A*, para generar una solución inicial segura que se aleja de los obstáculos; y el segundo basado en un método de post-procesamiento para suavizar la trayectoria inicial cumpliendo los requisitos de seguridad.

4.1. Planificador heurístico

El planificador propuesto en este trabajo está basado en el algoritmo A* usando L1. El objetivo es generar trayectorias seguras que consideran la distancia Manhattan a obstáculos. El algoritmo implementado está basado en (Cobano et al., 2022) donde se presentan algoritmos heurísticos que consideran el coste de distancias a obstáculos como representación del entorno, con campos de distancias euclídeas, e integran esta representación en el algoritmo además de explotar las propiedades de estos campos de distancias. Esto asegura trayectorias seguras y eficientes.

El trabajo realizado se ha enfocado en la adaptación e integración de la representación del entorno con L1 al planificador heurístico en sus funciones de coste. El algoritmo A* adaptado y que considera la distancia a obstáculos se denomina CA-A* (Cost-Aware A*). Como en (Cobano et al., 2022), la función de coste acumulado usada para considerar la distancia a los obstáculos es:

$$g(s_{i+1}) = g(s_i) + c(s_i, s_{i+1}) \tag{8}$$

donde $g(s_i)$ es el coste del nodo padre s_i y $c(s_i, s_{i+1})$ es el coste asociado con ir de s_i a s_{i+1}.

Dependiendo de cómo se defina $c(s_i, s_{i+1})$, se puede penalizar o favorecer diferentes trayectorias considerando el campo de distancias. Nosotros lo definimos así:

$$c(s_i, s_{i+1}) = \|s_{i+1} - s_i\| + \frac{c_w}{O(s_i, s_{i+1})} \tag{9}$$

donde $\|s_{i+1} - s_i\|$ es la distancia Manhattan entre los nodos s_i y s_{i+1}, c_w es un factor de peso de coste y $O(s_i, s_{i+1})$ es la integral del campo de distancias a lo largo del segmento definido por s_i y s_{i+1}:

$$O(s_i, s_{i+1}) = \frac{MDF(s_i) + MDF(s_{i+1})}{2} \cdot N \tag{10}$$

donde N es el número de nodos entre s_i y s_{i+1}.

Nótese cómo el MDF de un nodo es mayor cuanto más lejos está de un obstáculo; por lo tanto, los caminos son más seguros cuando el término integral $O(s_i, s_{i+1})$ es mayor. Considerando que el planificador minimiza el costo acumulado g, invertimos la integral del MDF en la función de costo como $O(s_i, s_{i+1})^{-1}$.

4.2. Post-procesamiento

Una vez obtenida la trayectoria inicial definida por celdas consecutivas gracias al planificador heurístico, se reprocesa con el objetivo de obtener un camino en 3D suave y navegable por un dron. Para ello, se busca obtener una representación polinómica continua y suave de la trayectoria que debe ser parecida a la inicial para mantener la seguridad. En este caso se han utilizado polinomios de grado 5, al tener el suficiente número de grados de libertad como para adaptarse a trayectorias complejas sin tener demasiados parámetros que dificulten el cálculo, quedando de la siguiente forma:

$$x(\tau) = c_0\tau^5 + c_1\tau^4 + c_2\tau^3 + c_3\tau^2 + c_4\tau + d_0 \tag{11}$$

$$y(\tau) = c_5\tau^5 + c_6\tau^4 + c_7\tau^3 + c_8\tau^2 + c_9\tau + d_1 \tag{12}$$

$$z(\tau) = c_{10}\tau^5 + c_{11}\tau^4 + c_{12}\tau^3 + c_{13}\tau^2 + c_{14}\tau + d_2 \tag{13}$$

Se utiliza como variable independiente el tiempo adimensionalizado $\tau \in [0, 1]$ para evitar inestabilidades numéricas durante el proceso de suavizado detallado a continuación.

Como primer paso a la hora de suavizar la trayectoria, se utiliza el algoritmo de Ramer–Douglas–Peucker (RDP) para eliminar waypoints de la trayectoria discreta que no aporten información. En la práctica, dicho algoritmo elimina los puntos de paso intermedios en las líneas rectas, dejando solo las

esquinas y reduciendo el número total de waypoints sin perder información.

Posteriormente, el cálculo de la trayectoria suavizada se hace empleando el optimizador no lineal Ceres Solver (Agarwal et al., 2023), que trata de ajustar un cierto vector de estados de forma que se minimice una función de pérdida objetivo que tiene la siguiente forma:

$$\min_{\mathbf{c}} \ \frac{1}{2} \sum_{i} \left(\|e_i(\mathbf{c})\|^2 \right) \tag{14}$$

Cada $\mathbf{e}_i(\mathbf{c})$ es una cierta función de coste que depende del vector de estados $\mathbf{c}$. Según la forma de parametrizar la trayectoria continua que hemos seleccionado, y dado que los coeficientes independientes (d_0, d_1, d_2) están predefinidos por el punto inicial de la trayectoria (siendo los únicos no nulos cuando $\tau = 0$), el vector de estados lo formarán los 15 coeficientes no fijados de los polinomios, de la forma

$$\mathbf{c} = \left[c_0, c_1, c_2, \ldots, c_{12}, c_{13}, c_{14} \right] \tag{15}$$

Para definir el valor inicial de los estados previamente a la optimización, se han seleccionado de forma que la trayectoria inicial sea una línea recta entre el principio y el final de la trayectoria discreta. Esto anula los coeficientes que no son de grado 1 y permite un cálculo analítico sencillo de los pocos coeficientes no nulos. Definiendo $\mathbf{c}'$ como el estado inicial, se obtiene

$$\mathbf{c}' = \left[0, 0, 0, 0, c'_4, 0, 0, 0, 0, c'_9, 0, 0, 0, 0, c'_{14} \right] \tag{16}$$

$$c'_4 = \frac{x_{\text{goal}} - d_0}{\tau_{goal}}, \quad c'_9 = \frac{y_{\text{goal}} - d_1}{\tau_{goal}}, \quad c'_{14} = \frac{z_{\text{goal}} - d_2}{\tau_{goal}} \tag{17}$$

Una vez seleccionado el estado inicial, el siguiente paso es definir las funciones de coste utilizadas por el optimizador, de forma que la trayectoria final se parezca lo más posible a la generada por el planificador heurístico.

La primera función de coste trata de hacer que el polinomio pase lo más cerca posible de los puntos de la trayectoria discreta. Para ello, previamente a la optimización, el algoritmo le asigna un determinado valor de tiempo adimensionalizado τ_k a cada waypoint k de forma proporcional a la longitud de arco acumulada. De esta forma, cada punto tendrá asociado el valor de tiempo que le correspondería si la trayectoria discreta fuera recorrida a velocidad constante.

Teniendo las parejas waypoint-tiempo, el optimizador genera una función de coste por cada waypoint, donde el coste es la diferencia entre el valor del polinomio que define la trayectoria cuando $\tau = \tau_k$ y la posición del waypoint k. Así, se penaliza por tanto que la función suavizada se aleje de los puntos generados por el planificador heurístico. La función de coste $e_{1,k}(\mathbf{c})$ asociada al waypoint k está definida de la siguiente forma:

$$\begin{aligned} e_{1,k}(\mathbf{c}) = w_{waypoint} \Big(& (x(\tau_k) - x_k)^2 + \\ & + (y(\tau_k) - y_k)^2 + (z(\tau_k) - z_k)^2 \Big) \end{aligned} \tag{18}$$

La constante $w_{waypoint}$ es un coeficiente de ponderación, o peso, que equilibra la influencia de esta función de coste respecto a las demás. La influencia de cambiar los pesos asociados a las diferentes funciones de coste sobre la trayectoria final será discutida durante la validación experimental.

La segunda función de coste trata de promover que el optimizador utilice coeficientes asociados a términos de bajo orden respecto a los de alto orden. Minimizar en la medida de lo posible coeficientes de alto orden permite obtener generalmente trayectorias más suaves, justo el objetivo de este posprocesado. La función de coste empleada es la siguiente:

$$\begin{aligned} e_2(\mathbf{c}) = w_{smoothness} \Big(& 25c_0^2 + 16c_1^2 + 9c_2^2 + 4c_3^2 + 25c_5^2 \\ & + 16c_6^2 + 9c_7^2 + 4c_8^2 + 25c_{10}^2 + 16c_{11}^2 + 9c_{12}^2 + 4c_{13}^2 \Big) \end{aligned} \tag{19}$$

Por último, la tercera función de coste trata de imponer que el final de la trayectoria optimizada coincida con el final de la trayectoria generada con el método heurístico. Al contrario de lo que ocurre con el punto inicial de la trayectoria, que es fijo independientemente del vector de estados resultante, que ambos finales coincidan depende del optimizador. El peso de esta función de coste es varios órdenes de magnitud mayor al del resto de funciones, para que el optimizador lo trate como una restricción dura:

$$\begin{aligned} e_3(\mathbf{c}) = w_{goalfix} \Big(& (x(1) - x_{goal})^2 + \\ & + (y(1) - y_{goal})^2 + (z(1) - z_{goal})^2 \Big) \end{aligned} \tag{20}$$

El optimizador iterará utilizando los gradientes analíticos de las diferentes funciones de coste, los cuales genera internamente de forma automática, hasta encontrar un valor del vector de estados $\mathbf{c}$ que minimice la función objetivo dada por la Ecuación 14. Tras la optimización, el polinomio resultante es una versión suavizada de la trayectoria calculada por el planificador heurístico.

5. Validación experimental

Esta sección muestra los resultados de las simulaciones de realizadas para evaluar el planificador propuesto en un entorno de construcción. En primer lugar, comparamos el algoritmo A* usando ambas métricas, L1 y L2, para evaluar la eficiencia y seguridad de la trayectoria. Después, mostramos resultados del planificador en varios escenarios y su funcionamiento considerando varias métricas de evaluación. El entorno considerado es una casa que se muestra en la Figura 5. Se consideran tres escenarios que vienen definidos por tres trayectorias: escenario 1, S1, con una trayectoria que entra en la casa pasando por una ventana, escenario 2, S2, con una trayectoria que va desde el salón a una habitación y escenario 3, S3, con una trayectoria que va desde fuera a una habitación del fondo atrevesando toda la casa.

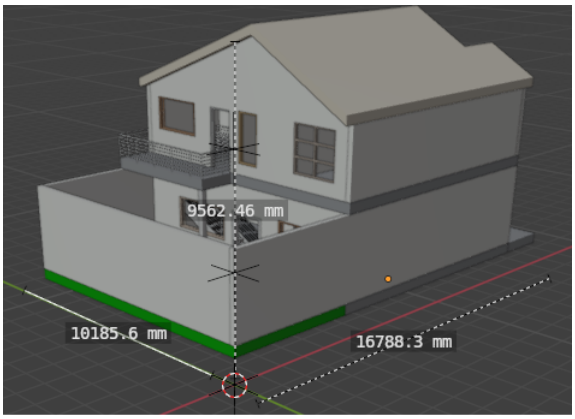

Figura 2: Entorno considerado en las simulaciones realizadas.

El ordenador y recursos usados para las simulaciones son: Intel Core i7-10700 CPU 2.90GHz, 32 GB RAM, con Ubuntu OS 22.04 LTS. La implementación se ha desarrollado en el lenguaje de programación C++ y se ha intregado en el sistema ROS (Robot Operating System) versión Noetic.

5.1. *Comparativa planificadores considerando L1 y L2*

Esta sección compara las características que ofrece el algoritmo A* original y considerando distancias a obstáculos (CA-A*), al usar las métricas L1 ó L2. También añadimos el algoritmo Lazy Theta* para mostrar cómo se comporta con la métrica L1.

La Tabla 1 muestra los resultados obtenidos respecto a nodos explorados, tiempo de computación y longitud de la trayectoria generada. Respecto al algoritmo CA-A* consideramos tres valores de *cw* diferentes (Ecuación 9). Lo primero que se puede resaltar en este tipo de escenarios es que en entornos con obstáculos alineados a los ejes, el uso de la métrica L1 produce una reducción muy significativa en el número de nodos expandidos y en el tiempo de cómputo. En cuanto a la longitud de la trayectoria generada, aunque es algo menor cuando se usa L1 con el CA-A*, es similar a la longitud con L2, lo que sugiere que las rutas obtenidas con L1 mantienen una aproximación eficiente aún bajo restricción discreta. La métrica L2 puede ofrecer trayectorias más suaves, pero a costa de un mayor número de nodos explorados y tiempo de cómputo. Estos resultados sugieren que en este tipo de entornos representados con una cuadrícula la métrica L1 presenta mejores prestaciones cuando la prioridad es la eficiencia.

Respecto al algoritmo Lazy Theta* (Nash and Tovey, 2010) destaca cómo con L1 obtiene los mismos resultados que el A*, es decir, tiende al comportamiento del algoritmo A*. Lazy Theta* es un algoritmo *any-angle* diseñado para trabajar con geometría euclídea y se basa en permitir que el padre de un nodo sea el padre de su vecino siempre que exista línea de vista, lo cual produce caminos más rectos y suaves que los generados por A*. La distancia Manhattan ó L1 no respeta esta propiedad excepto cuando los movimientos están perfectamente alineados a la cuadrícula. Por tanto, considear L1 rompe la base teórica del algoritmo Lazy Theta*. Resumiendo, Lazy Theta* obtiene ventajas únicamente cuando puede crear atajos en línea recta, por tanto, usar L1 destruye por completo la capacidad *any-angle* del algoritmo y su comportamiento es como el A*.

Tabla 1: Comparación algoritmos A* y Lazy Theta* con métricas L1 y L2 en el escenario 1: $cw1 = 100$, $cw2 = 1000$, $cw3 = 10000$.

	$Nodos$	$Tiempo(ms)$	$Longitud(m)$
A^*L1	1363	0,698	3,548
A^*L2	6175	4,086	3,548
$LazyTheta^*L1$	1363	0,699	3,548
$LazyTheta^*L2$	2168	2,016	3,393
$CA-A^*L1(cw1)$	9584	4,003	3,631
$CA-A^*L2(cw1)$	315551	264,533	4,071
$CA-A^*L1(cw2)$	148862	75,615	4,007
$CA-A^*L2(cw2)$	2983157	3925,440	4,071
$CA-A^*L1(cw3)$	1314379	1049,042	4,153
$CA-A^*L2(cw3)$	4015992	5014,172	4,341

5.2. *Resultados*

Esta sección presenta los resultados obtenidos considerando el planificador completo en los tres escenarios usando el algoritmo CA-A* con L1 para generar la trayectoria inicial con *cw2* (ver Tabla 1) y después suavizando la trayectoria. Se presentan los resultados en los escenarios S2 y S3. El objetivo es analizar las trayectorias obtenidas variando el peso $w_{smoothness}$ considerando varias métricas. El valor de $w_{waypoint}$ es 1000 y las métricas son: porcentaje de disminución de la longitud de la trayectoria inicial al suavizarla (L), el ángulo de giro máximo obtenido en la trayectoria suavizada (θ_{max}), el porcentaje de reducción de la suavidad al suavizarla considerando los cambios de ángulos entre waypoints consecutivos, y la distacia de Fréchet para evaluar cómo de diferentes son ambas trayectorias D_F. La distancia de Fréchet es una medida matemática que evalúa la similitud entre trayectorias, teniendo en cuenta las posiciones y el sentido y dirección del movimiento, es decir, no compara solo puntos fijos. Se conoce comúnmente como el "problema de la correa del perro".

La Figura 3 muestra la trayectoria inicial generada con CA-A* en el S2. La variación en altura no es significativa, pero sí se aprecian los cambios de ángulos en la trayectoria. La Figura 4 muestra nueve trayectorias suavizadas considerando nueve valores diferentes de $w_{smoothness}$: 0,01, 0,05, 0,1, 0,5, 1, 5, 10, 50, 100. Se muestran en 2D ya que la variación de la altura en S2 no es significativa. Se demuestra que en el caso de $w_{smoothness}$ bajos, 0,01 ó 0,05, la trayectoria es suavizada y se aproxima más a la inicial. La Figura 5 muestra los ángulos de giro de la trayectoria inicial y los de la trayectoria suavizada con $w_{smoothness} = 0,01$. Se puede observar como la trayectoria final es más suave.

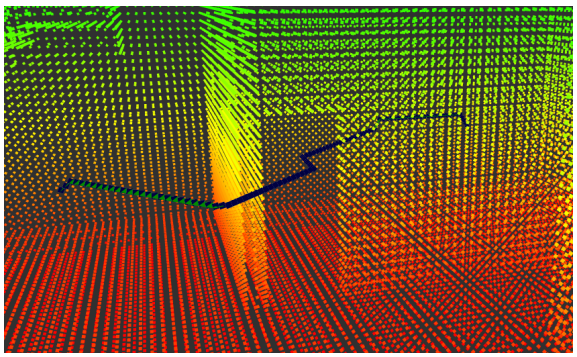

Figura 3: Trayectoria inicial en Rviz para entrar en una habitación.

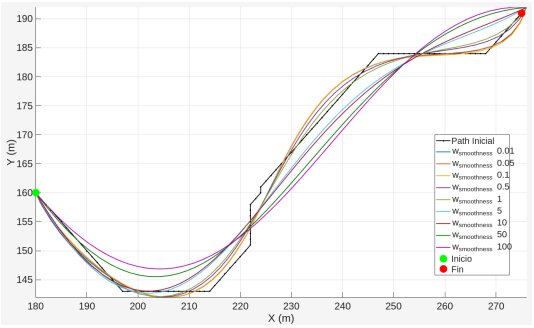

Figura 4: Trayectoria inicial y trayectorias generadas con diferentes $w_{smoothness}$.

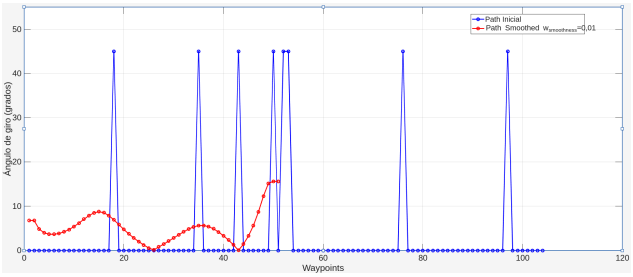

Figura 5: Ángulos de giro en la trayectoria inicial (azul) y la suavizada con $w_{smoothness} = 0,01$ (roja).

La Tabla 2 muestra estos resultados considerando los nueve valores de $w_{smoothness}$ y las métricas presentadas anteriormente. Respecto al ángulo de giro máximo de la trayectoria inicial en S2 es 45 grados y en S3 es 54,74 grados. Destaca que en todos los casos se reduce la longitud de la trayectoria inicial al suavizarla y la disminución considerable del ángulo de giro máximo dependiendo del valor de $w_{smoothness}$ y el porcentaje de mejora de la suavidad. Por último, el valor de la distancia de Fréchet muestra cómo con valores bajos de $w_{smoothness}$ la trayectoria final se asemeja más a la trayectoria inicial consiguiendo reducir la longitud y mejorando considerablemente la suavidad.

Tabla 2: Resultados obtenidos en escenarios S2 y S3.

S	$w_{smoothness}$	$L(\%)$	θ_{max}	$Suavidad(\%)$	D_F
2	0,01	-1.07	15,63	65,27	0,1142
2	0,05	-1.20	14,76	67,21	0,1155
2	0,1	-1.02	14,98	66,72	0,1190
2	0,5	-2.19	13,09	70,91	0,1101
2	1	-2.98	10,78	76,05	0,1230
2	5	-5.17	7,82	82,62	0,2118
2	10	-5.70	7,67	82,94	0,2367
2	50	-7.80	5,80	87,11	0,3152
2	100	-8.67	5,63	87,48	0,3451
3	0,01	-4.22	4,92	91,01	0,2401
3	0,05	-4.22	4,92	91,02	0,2401
3	0,1	-4.22	4,92	91,01	0,2397
3	0,5	-4.22	4,92	91,01	0,2400
3	1	-4.25	4,91	91,02	0,2417
3	5	-4.25	4,85	91,14	0,2411
3	10	-4.27	4,94	90,97	0,2435
3	50	-4.72	4,85	91,15	0,2930
3	100	-5.03	4,90	91,04	0,3227

6. Conclusiones

Este artículo presenta un planificador de trayectorias 3D en entornos de construcción y demuestra que el uso de campos de distancias Manhattan (MDF) en planificadores 3D en estos entornos puede mejorar significativamente la eficiencia computacional respecto a la distancia euclídea, reduciendo nodos explorados y tiempo de cómputo sin comprometer la seguridad ni la longitud de las trayectorias. El planificador propuesto, basado en CA-A* con métrica L1 y posprocesado con polinomios de grado 5, logra trayectorias suaves con reducciones en los ángulos de giro. Además, la trayectoria generada es similar a la inicial, calculada con el algoritmo heurístico, considerando como métrica la distancia de Fréchet.

Los resultados en escenarios realistas confirman que la representación de la distancia de Manhattan mediante máscaras de bits puede facilitar actualizaciones rápidas del mapa. Por tanto, como trabajo futuro, el objetivo es la implementación de un planificador local, permitiendo replanificaciones en tiempo real ante cambios dinámicos del entorno.

Agradecimientos

Este trabajo ha sido realizado gracias a los proyectos COBUILD (PID2024-161069OB-C31) y PICRAH (PLEC2023-010353) financiados por la “Agencia Estatal de Investigación – Ministerio de Ciencia, Innovación y Universidades”.

Referencias

Agarwal, S., Mierle, K., Team, T. C. S., 10 2023. Ceres Solver.
URL: https://github.com/ceres-solver/ceres-solver

Cobano, J. A., Merino, L., Caballero, F., 2026. Exploiting euclidean distance field properties for fast and safe 3d planning with a modified lazy theta*. Robotics and Autonomous Systems 198, 105317.
DOI: https://doi.org/10.1016/j.robot.2025.105317

Cobano, J. A., Rey, R., Merino, L., Caballero, F., 2022. Fast cost-aware lazy-theta over euclidean distance functions for 3d planning of aerial robots in building-like environments. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 13486–13493.
DOI: 10.1109/IROS47612.2022.9982180

Follini, C., Magnago, V., Freitag, K., Terzer, M., Marcher, C., Riedl, M., Giusti, A., Matt, D., 12 2020. Bim-integrated collaborative robotics for application in building construction and maintenance. Robotics 10, 2.
DOI: 10.3390/robotics10010002

Gao, F., Wu, W., Lin, Y., Shen, S., 2018. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 344–351.
DOI: 10.1109/ICRA.2018.8462878

Jones, M., Baerentzen, J., Sramek, M., 2006. 3d distance fields: a survey of techniques and applications. IEEE Transactions on Visualization and Computer Graphics 12 (4), 581–599.
DOI: 10.1109/TVCG.2006.56

Kim, K., Peavy, M., 2022. Bim-based semantic building world modeling for robot task planning and execution in built environments. Automation in Construction 138, 104247.
DOI: 10.1016/j.autcon.2022.104247

Nash, S. K., Tovey, C., 2010. Lazy theta*: Any-angle path planning and path length analysis in 3d. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI). p. 147–154.

Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., Galceran, E., 2016. Continuous-time trajectory optimization for online uav replanning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). p. 5332–5339.
DOI: 10.1109/IROS.2016.7759784

Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S., 2019. Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robotics and Automation Letters 4 (4), 3529–3536.
DOI: 10.1109/LRA.2019.2927938