Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 21 (2024) 262-273

www.revista-riai.org

Planificación jerárquica de movimientos de un robot trepador bípedo en estructuras tridimensionales reticulares

Marc Fabregat-Jaén[a,*], Adrián Peidró[a], Paula Mollá-Santamaría[a], Francisco José Soler[a], Óscar Reinoso[a,b]

[a]*Instituto de Investigación en Ingeniería de Elche (I3E), Universidad Miguel Hernández, Avda. de la Universidad s/n, Edificio Innova, 03202, Elche, España.*
[b]*ValgrAI: Valencian Graduate School and Research Network of Artificial Intelligence, Camí de Vera s/n, Edificio 3Q, 46022 Valencia, España.*

To cite this article: Fabregat-Jaén, M., Peidró, A., Mollá-Santamaría, P., Soler, F.J., Reinoso, Ó. 2024. Hierarchical motion planning of a biped climbing robot in three-dimensional truss structures. Revista Iberoamericana de Automática e Informática Industrial 21, 262-273. https://doi.org/10.4995/riai.2024.20779

Resumen

Los robots trepadores deben ser capaces de navegar autónomamente estructuras tridimensionales reticulares para evitar que operarios humanos se expongan a riesgos significativos al realizar tareas de mantenimiento en tales escenarios. Este artículo presenta un algoritmo de planificación jerárquica de movimientos para robots trepadores bípedos. A diferencia de las técnicas convencionales, nuestro algoritmo descompone el problema global en varios subproblemas, cada uno dedicado a gestionar aspectos específicos del proceso de generar una secuencia de puntos de adhesión. De forma inicial, se planifica la ruta global, que incluye la secuencia de caras que se atravesarán para alcanzar el punto designado, y qué puntos de transición se emplearán para cambiar de una cara a otra de la secuencia. Posteriormente, se calcula el camino que deberá recorrer el robot a lo largo de cada una de las caras que conforman la ruta global. Para la validación del método presentado, se incluyen imágenes y vídeo en un entorno de simulación.

Palabras clave: Planificación de trayectorias, Robots trepadores, Robots redundantes, Estructuras reticulares, Espacio de trabajo

Hierarchical motion planning of a biped climbing robot in three-dimensional truss structures

Abstract

Climbing robots must be capable of autonomously navigating three-dimensional truss-like structures to prevent human operators from being exposed to significant physical risks when performing maintenance tasks in such environments. This article presents a hierarchical motion planning algorithm for biped climbing robots. Unlike other conventional techniques, our algorithm decomposes the global three-dimensional problem into multiple sub-problems, each one dedicated to managing specific aspects of the process of generating the sequence of footholds. Initially, the global route is planned, which includes the sequence of faces to be traversed to reach the designated point, identifying which transition points will be used for changing from one face to another in the sequence. Subsequently, the path that the robot must follow along each of the faces comprising the global route is calculated. For the validation of the presented method, video and images taken in a simulation environment are included.

Keywords: Path planning, Climbing robots, Redundant robots, Truss structures, Workspace

1. Introducción

Toda construcción precisa de supervisiones rutinarias y tareas de conservación y mantenimiento. Estas acciones suelen desempeñarlas trabajadores humanos en ubicaciones de considerable altura, poniendo en riesgo su integridad física en el transcurso. La finalidad principal de los robots escaladores es reemplazar la ejecución de dichas labores altamente arriesgadas en ubicaciones que son intrínsecamente peligrosas para los seres humanos, o directamente inaccesibles. En la literatura se han propuesto robots trepadores para diversas aplicaciones, como el ajuste de contenedores de metal (Chen et al., 2019), la inspección de puentes metálicos (Nguyen and La, 2021), de cascos

*Autor para correspondencia: mfabregat@umh.es.

de barcos (Huang et al., 2017), y de grietas (Jang et al., 2020).

Para ejecutar estas tareas, los robots deben tener la capacidad de explorar escenarios, a menudo complejos, en tres dimensiones. Dependiendo de cómo se exploran estas estructuras, se pueden clasificar dos tipos de locomoción (Tavakoli et al., 2011). En primer lugar, los robots escaladores de movimiento continuo utilizan ruedas o carriles para desplazarse por el entorno, resultando en robots sencillos y rápidos (Chen et al., 2019; Nguyen and La, 2021; Huang et al., 2017).

Por otra parte, los robots trepadores de movimiento paso a paso constan de patas, en cuyos extremos se ubican garras, las cuales están unidas mediante cadenas cinemáticas de varios grados de libertad (GDL), que les dota de mayor maniobrabilidad para evitar obstáculos a costa de una mayor lentitud. Este artículo se enfoca en robots trepadores bípedos, que son aquellos que poseen dos garras. Durante su movimiento, a cada paso, uno de los extremos o garras permanece fijo en la estructura; mientras que la otra garra, que está libre, se mueve actuando la mencionada cadena cinemática hasta colocarla en una pose deseada. Al finalizar el movimiento, el extremo libre se pega al plano trepado, liberando la garra previamente fija. Este intercambio de roles permite llevar a cabo un nuevo paso. Algunos ejemplos de robots trepadores de estructuras de movimiento paso a paso son: HyReCRo (Peidró et al., 2019), CROC (Yang et al., 2016), W-Climbot (Zhu et al., 2020), 3DCLIMBER (Tavakoli et al., 2011), y ROMA 2 (Gimenez et al., 2002).

Considerando que los robots paso a paso trepan por estructuras metálicas reticulares, formadas por diversas vigas y barras que tienen una sección transversal considerablemente menor que su longitud, los mecanismos de adhesión que suelen usar estos robots para adherirse a dichas vigas son preferentemente de tres tipos: agarre prensil mecánico (Gimenez et al., 2002; Tavakoli et al., 2011), con el que se usan garras formadas por varios dedos que abrazan dichas vigas; adhesión magnética (Yang et al., 2016; Peidró et al., 2019), con el que se utilizan electroimanes o imanes permanentes que se pegan a las vigas gracias a su ferromagnetismo; o succión neumática (Zhu et al., 2020; Prados et al., 2023), con el que se emplean bombas de vacío o efecto Venturi para adherirse a las superficies. La solución que adopta el robot HyReCRo es la adhesión magnética, que tiene la ventaja de requerir el acceso a las vigas de la estructura solo por una de sus caras, en lugar de requerir abrazar dichas vigas por varias de sus caras de forma simultánea, como el agarre prensil mecánico. El algoritmo de planificación de trayectorias que presentamos en este artículo, por tanto, asume que las garras del robot requieren acceder solo a una de las caras de las vigas de la estructura para pegarse a ellas, quedando fuera del alcance del algoritmo la sujeción mediante garras prensiles.

Para ejecutar trabajos en un escenario reticular tridimensional, los robots trepadores necesitan la capacidad de navegarla autónomamente. El problema de planificación de movimientos consiste en calcular una secuencia de posiciones y orientaciones (poses), que permitan al robot alcanzar un punto deseado en la estructura, a partir de un punto de origen. Las soluciones convencionales al problema, como las propuestas para rovers, coches autónomos o robots aspiradores, están formuladas para planificar movimientos en un entorno bidimensional. No obstante, los robots trepadores añaden una nueva dimensión al trabajar en entornos verticales tridimensionales. Consecuentemente, se deben explorar espacios de búsqueda mayores, y el tiempo de cómputo que conllevan estas técnicas de planificación de trayectorias clásicas se vuelve prohibitivamente alto. Aunque la literatura ha presentado soluciones convincentes para la planificación de movimientos de robots trepadores de movimiento continuo (Breitenmoser and Siegwart, 2012; Stumm et al., 2012; Fang et al., 2020), la planificación de movimientos para robots de movimiento paso a paso sigue siendo una cuestión abierta. Estos robots requieren de una serie discreta de puntos de adhesión en vez de una trayectoria continuo, por lo que las técnicas mencionadas resultan inadecuadas.

Un reducido grupo de investigadores ha presentado soluciones al desafiante problema que supone la planificación de movimientos en robots trepadores paso a paso. El trabajo de Yang et al. (2016), propone obtener una ruta utilizando el método de Levenberg-Marquardt para obtener la solución de un sistema no lineal compuesto de funciones de coste sigmoidales, con el objetivo controlar la pose de la garra libre (efector final), mientras se evitan colisiones y se respetan límites articulares. Por otro lado, en (Chen et al., 2016) se propone una solución, basada en representar el escenario por medio de voxels, para resolver el problema con un algoritmo multicapa basado en A*. Este algoritmo utiliza hasta cuatro capas secuenciales en caso de que la capa anterior falle o caiga en un óptimo local. El enfoque del algoritmo A* también se implementa en el trabajo de Quin et al. (2016), donde se genera un grafo de *footholds* válidos y, mediante una función heurística, se evalúa cada nodo a partir de la información del escenario tomada por sensores. En una línea análoga, en (Pagano and Liu, 2017), se construye un árbol basándose en el entorno captado por los sensores del robot. Aunando los dos enfoques, Zhu et al. (2020) presentan un algoritmo multifase. En la primera fase, se determinan los puntos de transición entre las caras de la estructura mediante la optimización de un sistema no lineal, similar al enfoque de (Yang et al., 2016). En la segunda fase, se calcula la trayectoria en cada cara determinando el próximo punto de adhesión de acuerdo a la distancia a una ruta obtenida previamente con el algoritmo A*.

Este artículo presenta un algoritmo de planificación de movimientos jerárquico de un robot trepador bípedo de movimiento paso a paso para estructuras reticulares. El algoritmo consta de dos planificadores principales: el Planificador de Puntos de Transición (PPT), que, combinando el algoritmo A* con técnicas de inteligencia artificial, determina la secuencia global óptima de caras de la estructura que deben recorrerse, así como los puntos de las caras que permiten al robot pasar de una cara a la siguiente; y el Planificador de Puntos Intermedios (PPI), que determina la secuencia de puntos en los que las garras del robot deben pegarse para recorrer cada una de las caras. El planificador PPI, que opera con los puntos generados por el planificador PPT, agrega un nivel adicional a la jerarquía de la solución presentada, ya que planifica primero, de forma aproximada, un camino continuo (PPIC) dentro de cada cara; y, a continuación, refina dicho camino continuo mediante un algoritmo novel que determina el camino discreto (PPID): es decir, la secuencia de puntos discretos a los que deben ir pegándose las garras del robot para recorrer la cara. La solución presentada bebe de la idea general de fragmentar el problema global tridimensional en subproblemas en superficies bidimensionales, expuesta en (Zhu et al., 2020) . Sin embargo, los métodos utilizados para

abordar el problema presente en cada nivel de la jerarquía del planificador son distintos a los utilizados en el citado artículo. Gracias a ello, nuestra solución es aplicable a robots con una cadena cinemática más compleja, como es el caso del robot HyReCRo estudiado en este artículo.

La estructura del artículo restante está organizada de la siguiente manera. El robot HyReCRo se presenta en la Sección 2, así como el modelado y tratamiento previo de la estructura por donde se moverá el robot. Posteriormente, la Sección 3 presenta el planificador de movimientos, con los diferentes niveles de la jerarquía del algoritmo. Una simulación que valida el correcto funcionamiento del método propuesto se muestra en la Sección 4. Por último, en la Sección 5, se extraen las conclusiones del artículo y se plantean futuras líneas de investigación.

2. Un robot trepador de estructuras bípedo

2.1. Robot HyReCRo

El algoritmo propuesto a lo largo del artículo ha sido estudiado utilizando el prototipo del robot HyReCRo mostrado en la Figura 1(a). Este es un robot bípedo trepador de movimiento paso a paso, con el fin de realizar tareas de conservación e inspección en estructuras tridimensionales reticulares metálicas. El robot se compone de dos patas conectadas en serie mediante una cadera, utilizando articulaciones rotacionales (θ_j en la Figura 1(b)). La cinemática de la pata genérica j se muestra en la Figura 1(b). Cada pata está configurada por dos mecanismos paralelos de 2 GDL idénticos, cuyas longitudes se modifican por medio de 4 actuadores lineales (l_{1j}, r_{1j}, l_{2j} y r_{2j} en la Figura 1(b)), y cada pareja de mecanismos comparte un eslabón central común (cuerpo central).

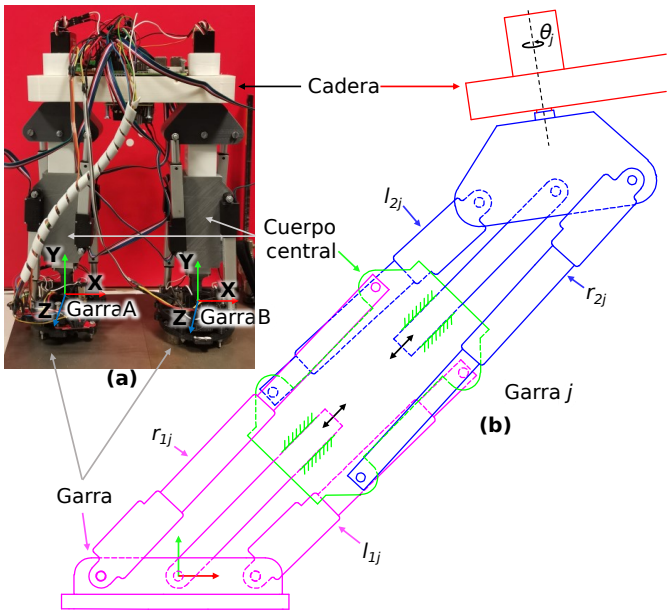

Figura 1: (a) Prototipo del robot HyReCRo. (b) Cinemática de la pata j.

Como resultado, el robot es de arquitectura híbrida serie-paralela, y posee 10 GDL, lo que conlleva una alta redundancia cinemática. El robot utiliza dos garras magnéticas ubicadas en ambos extremos de la cadena cinemática para trepar por estructuras tridimensionales reticulares metálicas (Peidró et al., 2019). Para automatizar el proceso de pegado y dotar al robot de la autonomía que precisa, en la garra se encuentran instalados tres sensores de distancia. En (Fabregat-Jaén et al., 2022), se presentó un algoritmo para el control cinemático del pegado, que utiliza la información de los sensores para identificar la superficie trepada y controlar la aproximación a la misma.

2.2. Notación y modelado de la estructura

Los robots trepadores deben desplazarse a través de escenarios tridimensionales, tales como los resultantes de estructuras reticulares presentes en los esqueletos de numerosas naves industriales, puentes o torres de distribución eléctrica. Son escenarios sumamente estructurados, compuestos de vigas interconectadas. Cada viga se puede describir como el conjunto de las caras planas que la forman. Cada una de estas caras F se modela con un polígono $\mathcal{P}$, el cual se describe por la posición de sus vértices en el espacio tridimensional. Además, cada cara lleva asociada un vector unitario $\mathbf{n}$, normal al plano de dicha cara, que apunta hacia el lado que es accesible por el robot (Zhu et al., 2020):

$$F = (\mathcal{P}, \mathbf{n}) \tag{1}$$

donde

$$\mathcal{P} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_v} \mid n_v \geq 3\} \tag{2}$$

y $\mathbf{v}_n$ es cada uno de los n_v vértices coplanares del polígono.

Por lo tanto, una estructura reticular tridimensional $\mathcal{E}$ se modela como una colección de caras, que representa los elementos que componen la estructura:

$$\mathcal{E} = \{F_1, F_2, \ldots, F_{n_f}\} \tag{3}$$

donde n_f es el número de caras que forman la estructura.

Todos los vectores están formulados relativos a un sistema de referencia inercial, que se encuentra fijado en un punto de la estructura, y que referenciaremos como "sistema de referencia del mundo" a lo largo del artículo.

2.3. Preprocesamiento de la estructura

Para planificar de manera eficiente una trayectoria utilizando el método que proponemos, es esencial obtener los puntos de transición posibles en todas las caras de la estructura. Una pareja de puntos de transición se define como dos *footholds* (o puntos de pegado), cada uno perteneciente a distintas caras, donde el robot fijaría cada uno de sus extremos para llevar a cabo la transición entre ellas. Como paso previo, cada polígono $\mathcal{P}$, que define cada cara de la estructura, se encoge como se ilustra en la Figura 2c, generando una versión escalada del polígono $\mathcal{P}$ distanciada de éste una distancia d. De este modo, para cada cara se genera un primer polígono o perímetro encogido tomando $d = d_{cncv}$ (polígonos azules a trazos en la Figura 2), y un segundo polígono encogido tomando $d = d_{cnvx}$ (polígonos verdes a trazos en la Figura 2). d_{cncv} y d_{cnvx} son distancias, medidas desde el borde de la cara, a las que se puede colocar el centro de la garra del robot para permitir que éste realice transiciones cóncavas (entre caras de distintas vigas) y convexas (entre caras de una misma viga), respectivamente, a la vez que se garantiza que las garras apoyan completamente en las caras sin quedar parcialmente al aire. La elección de estas dos distancias d_{cnvx} y d_{cncv} se ha realizado calculando el espacio de trabajo del robot a orientación constante como se describe en (Peidró et al., 2016), con las orientaciones necesarias para realizar transiciones con-

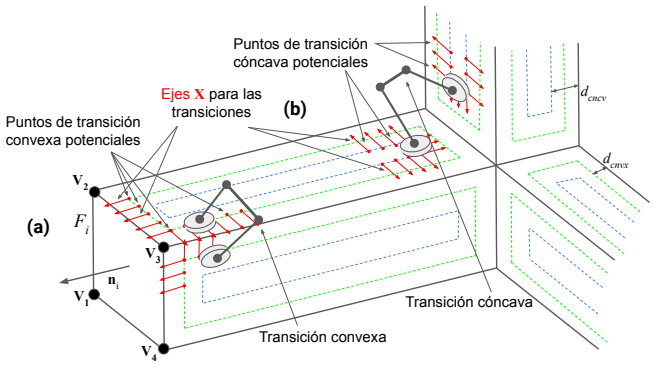

Figura 2: (a) Representación de la cara F_i del entorno, formada por el polígono $\mathcal{P}_i$, compuesto por los vértices $\mathbf{v}_i$ y el vector $\mathbf{n}_i$, normal a la cara. (b) Puntos de transición potenciales, obtenidos encogiendo los perímetros de las caras para establecer las transiciones cóncavas (puntos en azul) y convexas (puntos en verde). (c) Preceso de encogimiento de los polígonos.

vexas y cóncavas respectivamente, y tomando distancias que coloquen a la garra libre en puntos de dichos espacios de trabajo que, además, estén sobre la nueva cara.

Ambos perímetros reducidos se discretizan mediante un muestreo con una resolución especificada (cuanto más alta la resolución, mayor cantidad de puntos de transición se evaluarán posteriormente), proporcionando las posiciones potenciales de transición. Finalmente, a cada una de las posiciones se le asigna un vector (que define la orientación), el cual supone la dirección en la que el eje **X** de la garra del robot apuntaría al pegarse para iniciar la transición. Este vector de orientación, dibujado en rojo en la Figura 2, se dirigirá hacia el borde más próximo, considerando el borde como la intersección de los dos planos de las caras entre las cuales el robot llevaría a cabo la transición.

Un punto de pegado (o *foothold*) es la posición y orientación a las cuales se pega cada una de las garras del robot. Para efectuar un desplazamiento, el robot debe llevar a cabo un paso, durante el cual desplaza su extremo libre hasta la pose indicada (*foothold* deseado). Estas poses se pueden definir con una matriz de transformación homogénea 4×4, la cual implícitamente incorpora los valores de una traslación y una rotación respecto a un sistema de coordenadas:

$$\mathbf{FH} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

donde **x**, **y** y **z** son los ejes del sistema de referencia de la garra en el punto de pegado (**X**, **Y**, **Z** en la Figura 1); y **p** es la posición del *foothold* respecto al mundo.

Cada matriz determina una pose de pegado sobre una de las caras de la estructura, la cual siempre llevará un vector normal asociado (1). Es por ello que uno de los ejes que describen la orientación del *foothold* se conocerá en todo momento, debido a que estará asociado, por definición, a una cara, cuyo vector normal es conocido. En el caso del robot HyReCRo, cuando se encuentra pegado a un plano, el eje **Y** del extremo pegado coincidirá el vector normal al plano **n**. En consecuencia, mientras se expresen respecto al mismo sistema de referencia, el vector **y** del *foothold* coincidirá con el vector **n**, normal a la cara.

Así pues, los *footholds* potenciales de la estructura estarán completamente definidos, puesto que se conoce su posición **p**, resultado del proceso de discretización del perímetro encogido de cada cara; el vector **x**, que señala el borde de la transición; **y**, que corresponde al vector normal **n**; y el vector **z**, resultado del producto vectorial entre **x** y **y**.

3. Algoritmo de planificación de movimientos jerárquico

En esta sección, se propone un algoritmo jerárquico para la resolución del problema de planificación de movimientos tridimensional para un robot trepador de estructuras bípedo de movimiento paso a paso. Dados un punto de pegado inicial $\mathbf{FH}_i$ y un punto de pegado final $\mathbf{FH}_f$, el algoritmo genera una secuencia de *footholds* $\mathcal{FH}$ que posibilitarán al robot trepar desde el origen hasta el punto final, a través de la estructura. La jerarquía del método y sus niveles están ilustrados en el esquema de la Figura 3.

El nivel superior divide el problema en dos planificadores diferentes. Primero, el Planificador de Puntos de Transición (PPT), presentado en la Sección 3.1, obtiene la trayectoria global tridimensional; es decir, los *footholds* necesarios para alcanzar el objetivo (moverse desde $\mathbf{FH}_i$ hasta $\mathbf{FH}_f$). Cada una de estas transiciones necesarias entre caras viene definida por dos puntos. Por ejemplo: ${}^{a}\mathbf{FH}_{dep}$ y ${}^{b}\mathbf{FH}_{arr}$ indican los puntos de pegado de la garra fija y la garra libre, respectivamente, que permiten al robot cambiar desde la cara F_a hasta la cara F_b. Para ejecutar este planificador es necesario el procesar previamente la estructura de la forma que se ha mostrado en la Sección 2.3, con el fin de calcular los puntos de transición potenciales en todas las caras.

El segundo planificador (PPI, Sección 3.2), a partir de cada par de puntos devueltos por el planificador global PPT, obtiene los puntos de pegado intermedios necesarios para moverse a lo largo de la cara. Esto significa que, dados un punto de entrada $\mathbf{FH}_{arr}$ y otro de salida $\mathbf{FH}_{dep}$ de una cara, el planificador PPI calculará la serie de *footholds* que permitirá al robot llegar desde $\mathbf{FH}_{arr}$ hasta $\mathbf{FH}_{dep}$ a través de la cara. En el esquema mostrado, se puede observar que serán necesarios n planificadores PPI, donde n es el número de caras por las que pasa el camino global devuelto por el primer planificador PPT.

El planificador PPI añade otro nivel a la jerarquía al estar dividido en dos subniveles. En primer lugar, el planificador PPIC (Sección 3.2.1), obtiene la trayectoria continua más corta que une la pareja de puntos de transición devuelta por el primer planificador del primer nivel (PPT). Seguidamente, el planificador PPID, presentado en la Sección 3.2.2, utiliza dicha trayectoria continua para calcular la secuencia discreta de puntos de pegado que usará el robot para atravesar la cara en cuestión (desde $\mathbf{FH}_{arr}$ hasta $\mathbf{FH}_{dep}$). La unión de las salidas de cada uno de estos n bloques resulta en la secuencia de puntos de pegado $\mathcal{FH}$ que conectan el punto inicial $\mathbf{FH}_i$ con el final $\mathbf{FH}_f$ a través de la estructura.

3.1. *Planificador de Puntos de Transición (PPT)*

La finalidad de este planificador es producir la serie de puntos de transición que posibilite al robot llegar desde el punto inicial hasta el destino. Ambos puntos (origen y destino) representan poses, que incluyen posiciones y orientaciones, ubicadas generalmente en caras de diferentes elementos de la estructura. De este modo, se aborda el problema de manera integral, obteniendo en primer lugar las transiciones necesarias entre caras,

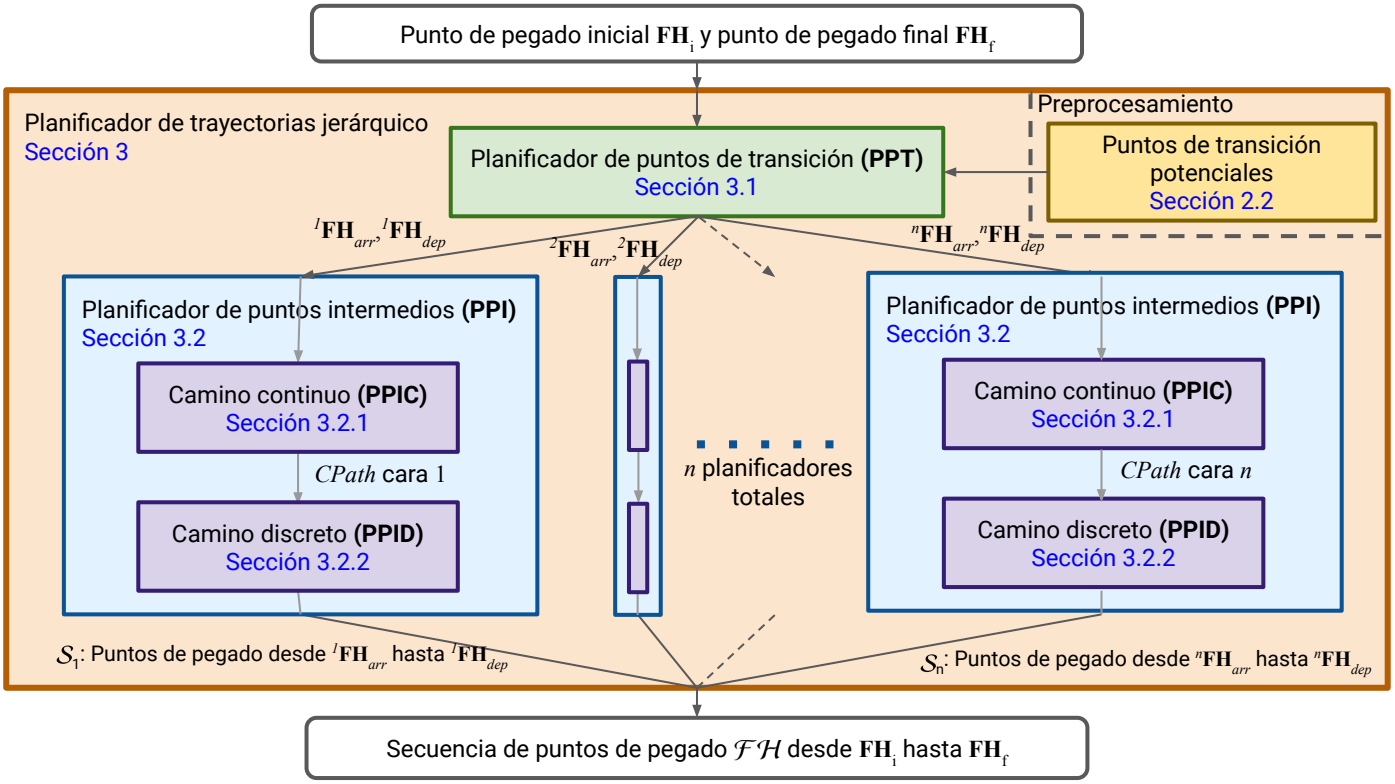

Figura 3: Esquema del algoritmo jerárquico de planificación de movimientos.

para, posteriormente, planificar el movimiento a lo largo de cada una en planificadores siguientes.

Para generar la trayectoria global (la serie de puntos de transición entre caras), se ha empleado un algoritmo fundamentado en el A*. Para implementarlo, el entorno se organiza en forma de grafo. El algoritmo A* se desplazará por el grafo en busca del camino óptimo, utilizando una función heurística para orientar la exploración del grafo (Hart et al., 1968). En nuestro caso, hemos seleccionado la distancia euclídea entre puntos como función heurística. El uso del algoritmo A* nos permite obtener el camino global óptimo de un solo paso, en contraste con la solución propuesta por Zhu et al. (2020), que para determinar las caras y puntos de transición emplean tres etapas: análisis de transitabilidad entre cada par de caras, búsqueda de aquellas caras que permiten unir punto inicial y el punto final, y optimización de los puntos de transición en dichas caras.

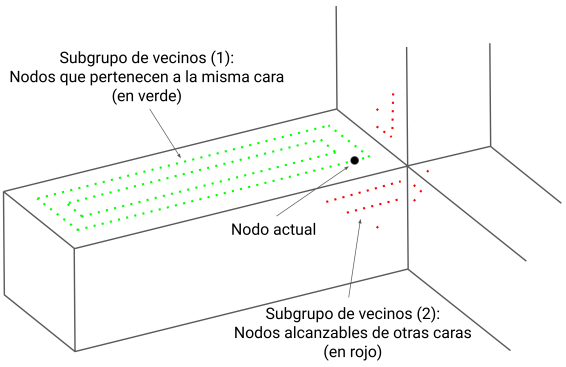

Figura 4: Vecinos de un nodo en cuestión y diferenciación de los subgrupos.

Los nodos que componen el grafo que se explora son los puntos de transición calculados con anterioridad (ver Sección 2.3 y Figura 2). En este punto de la planificación global, se destaca un aspecto clave que distingue al planificador propuesto de los métodos A* convencionales: la determinación de los vecinos de un nodo específico del grafo. Para un nodo determinado, la totalidad de sus vecinos está formada por (1) los demás puntos de pegado (nodos) pertenecientes a la misma cara, y (2) los *footholds* en otras caras que son accesibles desde el nodo en cuestión (ver Figura 4). La identificación de los nodos del subgrupo (2), es decir, aquellos puntos de transición alcanzables en otras caras, se lleva a cabo mediante un test de alcanzabilidad.

La capacidad de que una pose específica sea alcanzada por la garra libre se determina mediante un test de alcanzabilidad, partiendo de la premisa de que la garra fija está adherida al nodo actual, del cual se busca determinar los vecinos. En el caso de robots no redundantes, es sencillo de determinar, verificando si la solución de la cinemática inversa es real, evita colisiones y respeta los límites articulares. No obstante, los robots que presentan redundancia cinemática, como es el caso del robot HyReCRo, presentan infinitas posibles configuraciones como solución de su cinemática inversa. Por ende, no es posible verificar directamente la existencia de una solución válida, por lo que se recurre al test de alcanzabilidad para determinar si se puede alcanzar una posición y orientación determinadas. El rendimiento del test de alcanzabilidad es crucial para una planificación de movimientos veloz, puesto que constituye el elemento fundamental del planificador PPT. Durante el desarrollo del algoritmo, se han barajado varios métodos:

- **Iterativo**: se generan soluciones de la cinemática inversa para la pose sometida al test. Si se logra generar una

solución real, que cumpla con los límites articulares del robot, y esté libre de colisiones, la pose se identifica como alcanzable. Para robots redundantes (con n articulaciones actuadas y una pose definida por m coordenadas independientes), se seleccionan valores aleatorios para $(n-m)$ de las coordenadas articulares actuadas y se resuelven las otras m a partir de las ecuaciones de la cinemática inversa analítica (Peidro et al., 2015). En el caso de que no se obtenga una solución válida (real, que cumpla con los límites articulares y no produzca colisiones), se generan aleatoriamente nuevos valores y se comprueba que la solución a la cinemática inversa sea válida, o se haya alcanzado un número concreto de intentos (por ejemplo: 300). Este método se ha revelado como no repetible y lento. La variabilidad en la cantidad de iteraciones es considerable debido a la naturaleza aleatoria inherente al método, puesto que en situaciones en las que no existe una solución factible, se requiere esperar a que se realice el número de intentos preestablecidos, lo cual introduce una significativa fluctuación del tiempo de cálculo.

- **Analítico**: consiste en analizar la cinemática específica de cada robot con el fin de determinar de forma analítica la alcanzabilidad de una pose. Por ejemplo, se verifica si la posición deseada se encuentra a una distancia superior a la suma de las longitudes de los eslabones del manipulador cuando este está completamente extendido. Este es el enfoque seguido, por ejemplo, por Zhu et al. (2020), gracias a que su robot trepador tiene una cinemática muy sencilla y carece de redundancia, de modo que el espacio de trabajo cuando una garra está fija es una esfera, y se puede determinar fácilmente un test analítico de este tipo para determinar si el punto evaluado está al alcance. Sin embargo, en robots de cinemática más compleja o redundantes, como es el caso del robot HyReCRo estudiado en este artículo, tal criterio analítico no es fácil de formular, ya que el espacio de trabajo de este robot no es una esfera ni otro tipo de forma de geometría sencilla. Por lo tanto, el test analítico tiene un fuerte carácter *ad-hoc*, hecho a medida para cada robot, y difícil de generalizar.

- **Mediante redes neuronales**: se desarrolla una red neuronal de clasificación binaria para la determinación la alcanzabilidad de una pose, ya sea alcanzable o no alcanzable. Si bien la obtención de resultados es muy rápida una vez que la red está entrenada, se debe tener en cuenta que el proceso de entrenamiento y la generación de los datos utilizados para el entrenamiento conllevan un tiempo que puede ser considerable. Además, se debe tener en cuenta que la precisión de la red nunca puede ser perfecta, por lo que siempre existe la posibilidad de que una pose sea clasificada erróneamente.

Dada la complejidad cinemática del robot HyReCRo, el método analítico no es factible (Peidro et al., 2015). Por otro lado, el método iterativo presenta una variabilidad considerable en cuanto al tiempo de ejecución y el resultado obtenido. Ante estas consideraciones, se ha optado por el uso de redes neuronales para la determinación de la alcanzabilidad de una pose.

La resolución de la cinemática inversa del robot HyReCRo debe hacerse diferenciando entre dos situaciones (Peidro et al., 2015): una situación regular, y otra singular, en la que los ejes **Z** de las dos garras (véase la Figura 1) quedan paralelos o antiparalelos. Las redes neuronales de clasificación de la alcanzabilidad también se ven afectadas por esta distinción. Por ende, se han generado dos redes neuronales distintas: una destinada a poses regulares, contenidas en el plano de la cara a la que está pegada la garra fija; y otra específica para transiciones, que asumen posturas singulares en las que los ejes **Z** de ambas garras del robot son paralelos o antiparalelos, puesto que estas posturas son las más favorables para realizar este tipo de transiciones (Peidró et al., 2019).

La arquitectura de las redes se muestra en la Figura 5 y corresponde a un perceptrón multicapa. La capa de entrada está compuesta por 6 neuronas, equivalentes a los 6 parámetros que describen una pose en un espacio tridimensional (3 de posición (p_x, p_y, p_z) y 3 de rotación (r_x, r_y, r_z), correspondientes a ángulos de Euler **ZYX**). Las dos capas ocultas son lineales y están formadas por 32 y 64 neuronas, y ambas usan la función ReLU como función de activación. La capa de salida es de una única neurona, con función de activación sigmoide, que devuelve un valor entre 0 y 1, donde 0 indica que la pose no es alcanzable y 1 que sí lo es.

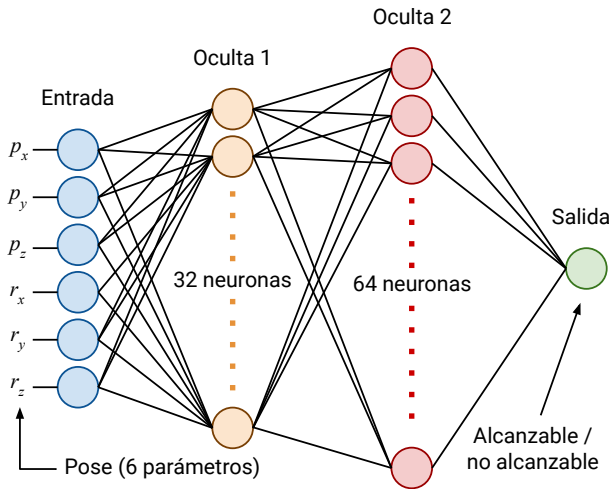

Figura 5: Arquitectura de la red neuronal de clasificación de alcanzabilidad.

Para entrenar ambas redes, se han obtenido datos de entrenamiento mediante la generación de poses aleatorias dentro de una aproximación del espacio de trabajo del robot en forma de esfera, y se han clasificado mediante el test de alcanzabilidad iterativo, almacenando los resultados (es decir, si existe solución o no). Se han generado 2×10^7 poses diferentes para el entrenamiento de cada red. Tras sintonizar los hiperparámetros y llegar a los seleccionados[1], se ha obtenido una precisión de 97,85% para transiciones entre caras y 98,32% para el resto de movimientos dentro de una cara; precisiones que resultan suficientes para los fines de este estudio. Con las redes entrenadas, formular la determinación de los vecinos es sencillo. En primer

[1]Tamaño lotes: 10, épocas: 200, tasa aprendizaje: 0.01, optimizador: SGD (Descenso Gradiente Estocástico), función pérdida: BCE (Entropía Cruzada Binaria).

lugar, se utilizan *k*-d trees (Bentley, 1975) como estructura de datos para realizar una búsqueda eficiente en una esfera, la cual se centra en la garra fija. El radio de la esfera necesita ser lo suficientemente amplio como para abarcar el espacio de trabajo completo del robot, y su elección requiere un análisis del espacio de trabajo, similar al enfoque seguido en (Peidró et al., 2017). Este paso permite descartar de manera instantánea la gran mayoría de las poses no alcanzables por el robot, evitando así la necesidad de aplicar el test de alcanzabilidad a estos puntos. Aunque esta aproximación como una esfera del espacio de trabajo es ciertamente imprecisa, únicamente se utiliza para descartar puntos que indudablemente no son alcanzables. Aquellos puntos que se incluyan en la aproximación (pertenezcan a la esfera), pero no formen parte realmente del espacio de trabajo, serán posteriormente descartados. Por último, todos los *footholds* seleccionados (que caen dentro de la esfera) se someten al test de alcanzabilidad, y se añaden a la colección de vecinos si se determinan como alcanzables y la configuración adquirida para alcanzar dicho punto no produce colisiones.

Como resultado, tal y como se ilustra en la Figura 6, el planificador PPT proporcionará la serie de puntos de transición que posibilitarán al robot alcanzar eficientemente el destino. Si una pareja de puntos de transición consecutivos se encuentra en una misma cara F_i, se designarán como punto de llegada ${}^{i}\mathbf{FH}_{arr}$ y de salida ${}^{i}\mathbf{FH}_{dep}$ de dicha cara. En el otro planificador de este nivel jerárquico, que se describe en la subsección siguiente 3.2, se calcularán los *footholds* intermedios dentro de una misma cara para cada pareja de puntos de llegada y de salida.

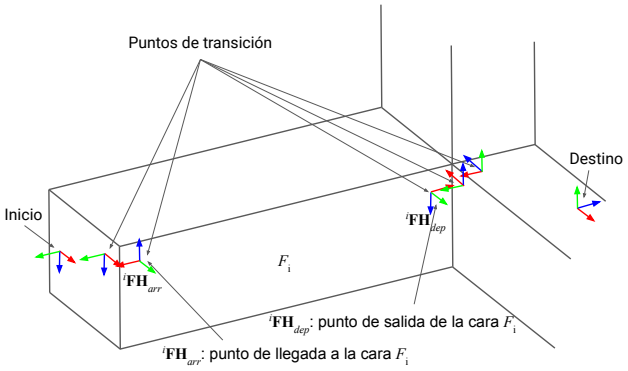

Figura 6: Puntos de transición devueltos por el planificador PPT.

3.2. *Planificador de Puntos Intermedios (PPI)*

En la ruta global generada por el planificador PPT, es posible que una pareja de puntos de transición consecutivos pertenezcan a una misma cara, tal y como se muestra con los puntos ${}^{i}\mathbf{FH}_{arr}$ y ${}^{i}\mathbf{FH}_{dep}$ en la Figura 6. En consecuencia, el robot necesita ser capaz de moverse a lo largo de la cara y trazar una ruta efectiva dentro de ella. El Planificador de Puntos Intermedios (PPI) entra en acción para determinar la ruta más eficiente en términos de longitud y número de pasos para recorrer la cara desde el punto de entrada hasta el punto de salida.

Con el planificador PPI se añade un nivel adicional de profundidad al planificador jerárquico, ya que se divide en dos planificadores. En primer lugar, se establece la trayectoria continua más corta mediante la construcción de un grafo de visibilidad. Posteriormente, se obtiene la serie de puntos de pegado discretos, adaptándolos en la medida de lo posible al camino continuo y maximizando la distancia cubierta en cada movimiento.

3.2.1. *Planificador de Puntos Intermedios Continuo (PPIC)*

El objetivo del primer planificador PPIC de este nivel es dar con la ruta continua *CPath* más corta que recorra la cara, uniendo los puntos de llegada y de salida de la cara. Para ello, en primer lugar, se reduce el perímetro del polígono que define la cara para formar un polígono ”seguro”, el cual asegura que las garras del robot se apoyen dentro de la cara completamente. A continuación, se procede a construir un grafo de visibilidad (Lee and Preparata, 1984), donde cada vértice del polígono es un nodo y la conexión entre cada par de nodos está indicada por las aristas del polígono.

Para resolver esta fase, Zhu et al. (2020) emplean un algoritmo A* para evadir o atravesar posibles obstáculos presentes en las caras. En nuestro caso, tales obstáculos no existen, ya que, en el caso de que los hubiera, se tratarían como nuevas caras que ya habrían sido procesadas por el planificador PPT. Por lo tanto, determinar el camino continuo mediante un grafo de visibilidad será la forma más computacionalmente eficiente de obtener el camino continuo más corto en cada cara.

En la Figura 7 (ignórense por el momento los puntos de pegado intermedios), se visualiza el camino continuo *CPath* calculado en una cara, junto con el perímetro “seguro” utilizado para estructurar el grafo.

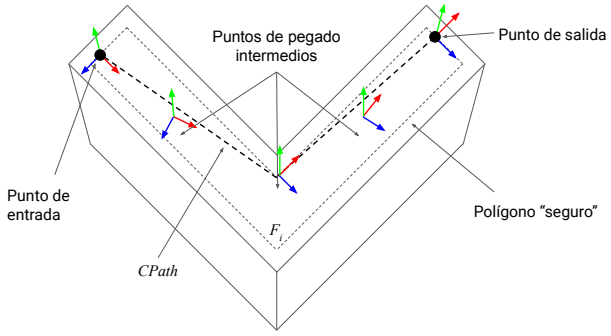

Figura 7: Salida del planificador PPI.

3.2.2. *Planificador de Puntos Intermedios Discreto (PPID)*

Una vez que se ha calculado el camino continuo, se procederá a determinar el camino discreto, compuesto por la secuencia de puntos de pegado en los que se adherirán las garras. De manera iterativa, el algoritmo calculará el siguiente mejor *foothold*, empleando como guía el camino continuo generado en el planificador PPIC. De forma intuitiva, se podría tomar el punto en el que la frontera del espacio de trabajo del robot interseca con el camino continuo devuelto por el planificador PPIC con el fin de maximizar la distancia recorrida en cada paso. Esta es la idea utilizada en (Zhu et al., 2020), y es apropiada cuando la cinemática del robot permite alcanzar cualquier posición de su espacio de trabajo con cualquier orientación. Sin embargo, en robots como el HyReCRo, donde algunas posiciones de su espacio de trabajo permiten alcanzar todas las orientaciones, pero otras solo un subconjunto de ellas, es necesario realizar una planificación mediante un algoritmo más sofisticado, que es el que se describe en esta sección.

Con el fin de acelerar la ejecución del algoritmo para ser empleado en tiempo real, se propone realizar un cálculo previo *offline* del espacio de trabajo plano para no tener que resolver la cinemática inversa para cada punto evaluado. Se entiende como espacio de trabajo plano a la intersección entre el espacio de trabajo alcanzable del robot y el plano que define la cara explorada, que coincidirá con el plano formado por los ejes **X** y **Z** de la garra fija en cada punto de pegado de la cara. En la Figura 8 se muestra el robot y su espacio de trabajo plano en perspectiva.

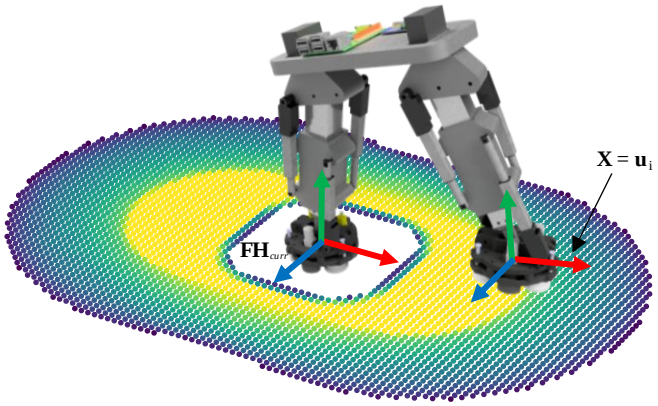

Figura 8: Vista en perspectiva del espacio de trabajo plano del robot HyReCRo.

Para generar el espacio de trabajo planar del robot, se discretiza en forma de malla, usando una resolución lo suficientemente alta como para obtener una discretización precisa y densa. Para cada punto de la malla, también se discretizan y calculan las orientaciones alcanzables. A partir de los puntos obtenidos, el modelo del espacio de trabajo se estructura como un conjunto de posiciones alcanzables (2D). A cada punto se le vincula una serie de vectores $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$, como se ilustra en la Figura 9(a), que son las orientaciones alcanzables de entre las discretizadas (1D). Cada elemento de este grupo de orientaciones es un vector unitario $\mathbf{u}_i$ paralelo a la cara, el cual sería el equivalente al eje **X** del extremo libre al pegarse en dicho punto. De modo que el espacio de trabajo planar se configura como un grupo de datos tridimensional, que recoge las poses alcanzables por el robot. De igual modo, para cada una de las poses alcanzables, se almacena una de las infinitas posibles soluciones de la cinemática inversa que permiten alcanzar dicha pose, que para ser conservadores a

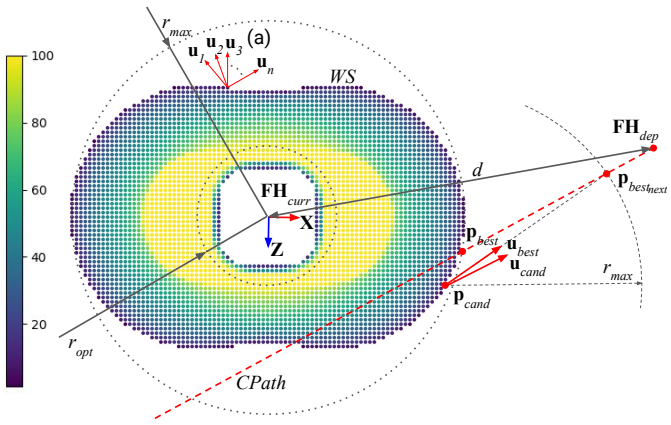

Figura 9: Espacio de trabajo plano del robot y planificador PPID (con $r = r_{max}$).

la hora de evitar colisiones con elementos el entorno, se elige como la configuración articular que maximiza el volumen ocupado por el robot El espacio de trabajo planar del robot HyReCRo se muestra en la Figura 9, donde se codifica el número de orientaciones alcanzables, de entre las discretizadas para cada punto, mediante un mapa de colores donde un valor del 100% (puntos amarillos) indica que se puede alcanzar cualquier orientación en esa posición; es decir, que el eje $\mathbf{u}_i$ mostrado en la Figura 9(a) podría apuntar en cualquier dirección del plano entre 0° y 360°.

El planificador PPID calculará iterativamente, desde el punto de llegada $\mathbf{FH}_{arr}$, el siguiente mejor punto de pegado $\mathbf{FH}_{best}$, siguiendo el camino continuo *CPath* devuelto por el planificador PPIC, hasta que se alcance el punto de salida $\mathbf{FH}_{dep}$, tal y como se muestra en el Algoritmo 1.

Algorithm 1 Planificador PPID

1: $\mathcal{S} \leftarrow \mathbf{FH}_{arr}$
2: $\mathbf{FH}_{curr} \leftarrow \mathbf{FH}_{arr}$
3: **while** $\mathbf{FH}_{curr} \neq \mathbf{FH}_{dep}$ **do**
4: $\quad d \leftarrow$ distancia entre $\mathbf{FH}_{curr}$ y $\mathbf{FH}_{dep}$
5: $\quad r \leftarrow \text{mín}(r_{max}, d - r_{opt})$
6: $\quad \mathbf{FH}_{curr} \leftarrow \text{NextBestFoothold}(\mathbf{FH}_{curr}, r)$
7: $\quad$ Añadir $\mathbf{FH}_{curr}$ a la secuencia $\mathcal{S}$
8: **return** $\mathcal{S}$ ▷ puntos de pegado desde $\mathbf{FH}_{arr}$ hasta $\mathbf{FH}_{dep}$

En cada iteración del planificador PPID, se calcula la distancia d (línea 4 del Algoritmo 1), que corresponde a la distancia entre el *foothold* de la garra fija en la iteración actual $\mathbf{FH}_{curr}$ y el destino en la cara $\mathbf{FH}_{dep}$. El alcance deseado r se obtiene a partir de la distancia d (línea 5). El alcance es la distancia que se desea cubrir con el paso de la iteración en cuestión (es decir, al pegar la garra libre estando la otra garra fijada en $\mathbf{FH}_{curr}$): $r = \text{mín}(r_{max}, d - r_{opt})$, donde r_{max} es el alcance máximo del robot, la cual es posible aproximar a partir de su espacio de trabajo plano (Figura 9); y r_{opt} se refiere al alcance óptimo, que corresponde al radio de la circunferencia formada por los puntos en el espacio de trabajo planar en los que se maximiza la cantidad de orientaciones alcanzables, como se muestra en la Figura 9. La segunda elección $(d - r_{opt})$ es la que permitiría preparar la pose de la garra fija de modo que, en el último paso, la pose deseada ($\mathbf{FH}_{dep}$) pertenezca a la circunferencia de radio r_{opt}, y que así la garra libre pueda ser pegada en dicha pose con cualquier orientación. El triedro $\mathbf{FH}_{dep}$ está impuesto por el planificador PPT, y es necesario alcanzar la orientación exigida por el triedro dado; esta elección de r asegura que el origen de este triedro caiga en la circunferencia r_{opt}, lo cual asegura la alcanzabilidad del punto con cualquier orientación.

Como muestra la línea 6 del Algoritmo 1, en cada iteración es necesario calcular el siguiente mejor punto de pegado $\mathbf{FH}_{best}$ mediante la rutina NextBestFoothold, que se muestra en el Algoritmo 2, y cuyos pasos se ilustran en la Figura 9 y se describen a continuación.

En primer lugar, se calcula la mejor posición de pegado $\mathbf{p}_{best}$ como la intersección del camino continuo *CPath* (previamente calculado por el planificador PPIC) y un círculo, centrado en el *foothold* actual del robot $\mathbf{FH}_{curr}$, y de radio igual al alcance deseado r (línea 2 del Algoritmo 2).

Algorithm 2 Cálculo del siguiente mejor punto de pegado

1: **function** NEXTBESTFOOTHOLD($\mathbf{FH}_{curr}$, r)
2: $\mathbf{p}_{best} \leftarrow$ intersección de $CPath$ y círculo($\mathbf{FH}_{curr}$, r)
3: $WS_{sort} \leftarrow WS$ contenido en $\mathcal{P}$, ordenado respecto a $\mathbf{p}_{best}$
4: **for** $\mathbf{p}_{cand}$ **in** WS_{sort} **do**
5: $\mathbf{p}_{best_{next}} \leftarrow$ intersección de $CPath$ y círculo($\mathbf{p}_{cand}$, r)
6: $\mathbf{u}_{best} \leftarrow$ vector desde $\mathbf{p}_{cand}$ que apunta hasta $\mathbf{p}_{best_{next}}$
7: $\mathbf{u}_{cand} \leftarrow$ vector en $\mathbf{p}_{cand}$ más cercano a $\mathbf{u}_{best}$
8: **if** $\mathbf{u}_{cand} \cdot \mathbf{u}_{best} \leq \varepsilon$ **and** libre de colisiones **then**
9: $\mathbf{FH}_{best} \leftarrow \{\mathbf{p}_{cand}, \mathbf{u}_{cand}\}$
10: **return** $\mathbf{FH}_{best}$

Seguidamente, el polígono $\mathcal{P}$, que define la cara de la que se está obteniendo el camino, se interseca con el espacio de trabajo planar WS (el mostrado en la Figura 9), descartando así los puntos del espacio de trabajo que caen fuera de la cara. Los puntos restantes se ordenan según su distancia a $\mathbf{p}_{best}$ (línea 3).

A continuación, se recorre cada punto del espacio de trabajo ya ordenado WS_{sort}, desde el más cercano a $\mathbf{p}_{best}$, hasta el más alejado (línea 4). Para cada posición de pegado candidata $\mathbf{p}_{cand}$, se calcula $\mathbf{u}_{best}$ como el vector unitario que señala $\mathbf{p}_{best_{next}}$ desde $\mathbf{p}_{cand}$ (línea 6). $\mathbf{p}_{best_{next}}$ es la predicción del mejor punto de pegado de la siguiente iteración, y se obtiene como la intersección de una circunferencia centrada en $\mathbf{p}_{cand}$ con un radio igual al alcance deseado r y del camino continuo $CPath$ (línea 5). De este modo $\mathbf{u}_{best}$ sería el eje $\mathbf{X}$ de la garra libre al fijarla en $\mathbf{p}_{cand}$, y, dado que el alcance del robot presenta su máximo en la dirección de este eje (ver Figura 9), se lograría maximizar el paso recorrido en la dirección de $\mathbf{u}_{best}$ en la siguiente iteración (cuando la garra libre se pegase en $\mathbf{p}_{cand}$ y pasase a ser la garra fija en la siguiente iteración del Algoritmo 1).

Seguidamente, se selecciona el vector $\mathbf{u}_{cand}$ más cercano a $\mathbf{u}_{best}$ (línea 7), a partir todos los vectores $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ que definen las orientaciones alcanzables en $\mathbf{p}_{cand}$, calculados durante el preprocesamiento del espacio de trabajo plano (Figura 9).

Para validar la elección, se debe comprobar que el vector elegido $\mathbf{u}_{cand}$ sea lo suficientemente próximo a $\mathbf{u}_{best}$. La cercanía se cuantifica mediante el producto escalar entre ambos vectores, cuyo valor indicará que serán cercanos cuando es próximo a 1. Si el resultado del producto escalar alcanza cierto valor umbral ε (e.g., $\varepsilon = 0{,}998$, que corresponde a un rango de aproximadamente $\pm 4°$) y la configuración articular adoptada para alcanzar $\mathbf{p}_{cand}$, con orientación $\mathbf{u}_{cand}$, está libre de colisiones (línea 8), se devuelve el punto de pegado $\mathbf{FH}_{best}$, formado por la posición $\mathbf{p}_{cand}$, $\mathbf{u}_{cand}$ como vector que define el eje $\mathbf{X}$, $\mathbf{Y}$ perpendicular a la cara, y el $\mathbf{Z}$ obtenido como el producto vectorial de $\mathbf{X}$ e $\mathbf{Y}$ (líneas 9-10).

Si no se cumplen las condiciones establecidas en la línea 8, el algoritmo seguirá recorriendo el espacio de trabajo plano ordenado (WS_{sort}, línea 4) hasta dar con una combinación de posición y orientación que sí lo haga.

Cabe destacar que, tanto la determinación del orden del espacio de trabajo, como la búsqueda de la posición y orientación más cercanas, hacen uso de k-d trees como estructura de datos para reducir los tiempos de búsqueda.

Finalmente, y volviendo al Algoritmo 1, tras determinar $\mathbf{FH}_{best}$, se añade a la secuencia $\mathcal{S}$ de puntos de pegado (línea 7), y todo el proceso se repite hasta que el punto de pegado actual coincide con el punto de salida de la cara $\mathbf{FH}_{dep}$ (linea 3). Tras esto, el Algoritmo 1 devuelve la secuencia $\mathcal{S}$ (línea 8) de puntos de pegado que une el punto de entrada $\mathbf{FH}_{arr}$ y el punto de salida de la cara $\mathbf{FH}_{dep}$, como se muestra en la Figura 7.

4. Simulación

Para validar el desempeño del algoritmo propuesto en este artículo, se ha llevado a cabo una simulación con el objetivo de planificar los movimientos del robot en una estructura reticular tridimensional. Los experimentos se han realizado en Python, con un procesador AMD Ryzen 7 5800U.

El entorno simulado consiste en dos vigas perpendiculares cruzadas. Estas vigas tienen longitudes de 1 y 2 metros, respectivamente, con perfiles cuadrados de lado igual a 0.15 metros. De esta manera, la solución propuesta puede ser evaluada en una estructura compleja que requiere de múltiples transiciones y movimientos a través de las caras exploradas.

El tratamiento previo de la estructura, detallado en la Sección 2.3, se ha realizado empleando un paso de 0.05 metros en la discretización de los *footholds* potenciales. Se han utilizado unas distancias de 0.25 y 0.07 metros para el cálculo de las transiciones cóncavas y convexas (d_{cncv} y d_{cnvx} en la Figura 2), respectivamente. En el planificador PPIC, se han generado los polígonos "seguros" reduciendo los perímetros de las caras una distancia equivalente a la utilizada para obtener los puntos de las transiciones convexas (0.07 metros). En lo que respecta al espacio de trabajo plano, utilizado en el planificador PPID, se ha generado con una resolución de 0.01 metros en las dimensiones de posición y 5 grados para la orientación, como se muestra en la Figura 9. Las resoluciones han sido seleccionadas tras un análisis, el cual se discute en la Sección 4.1.

En el vídeo adjunto (también disponible en: https://youtu.be/fUznNYWVfgQ) se puede observar la simulación completa, donde se muestra el camino generado por el algoritmo, así como el movimiento del robot a lo largo del mismo. En la Figura 10 se muestra el camino completo devuelto por el planificador jerárquico (Figura 10(a)) y varios instantes de la simulación (Figura 10(b-f)). Nótese que, tanto en la Figura 10, como en el vídeo adjunto (ambos corresponden al mismo experimento), el robot siempre se mueve libre de colisiones, a pesar de que, por la coloración de la representación, pueda parecer que existe alguna interferencia u oclusión. Esto se debe a cómo trata Matplotlib los gráficos 3D: aunque no haya interferencia, cuando 2 objetos se solapan desde el punto de vista de la cámara, su coloración se combina (el gris de la estructura y el azul del robot) y puede dar lugar a confusiones.

La trayectoria devuelta está compuesta de los triedros de la figura, que equivalen a los puntos de pegado donde las garras se irán adhiriendo. Es visible que, en una primera instancia, el Planificador de Puntos de Transición (PPT) determina la necesidad realizar 2 transiciones entre las caras, utilizando la serie de puntos de transición formada por los *footholds* $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ (Figura 10(a)) para llegar al punto deseado, partiendo desde el origen. En las Figuras 10(c) y 10(e) se representa al robot realizando las transiciones demandadas por el planificador PPT.

Para cada cara que se debe atravesar, el Planificador de Puntos Intermedios (PPI) determina los *footholds* que se emplearán para recorrer cada cara. El *foothold* intermedio ${}^{f}\mathbf{FH}_{i}$ representa el elemento número i en la secuencia de puntos de

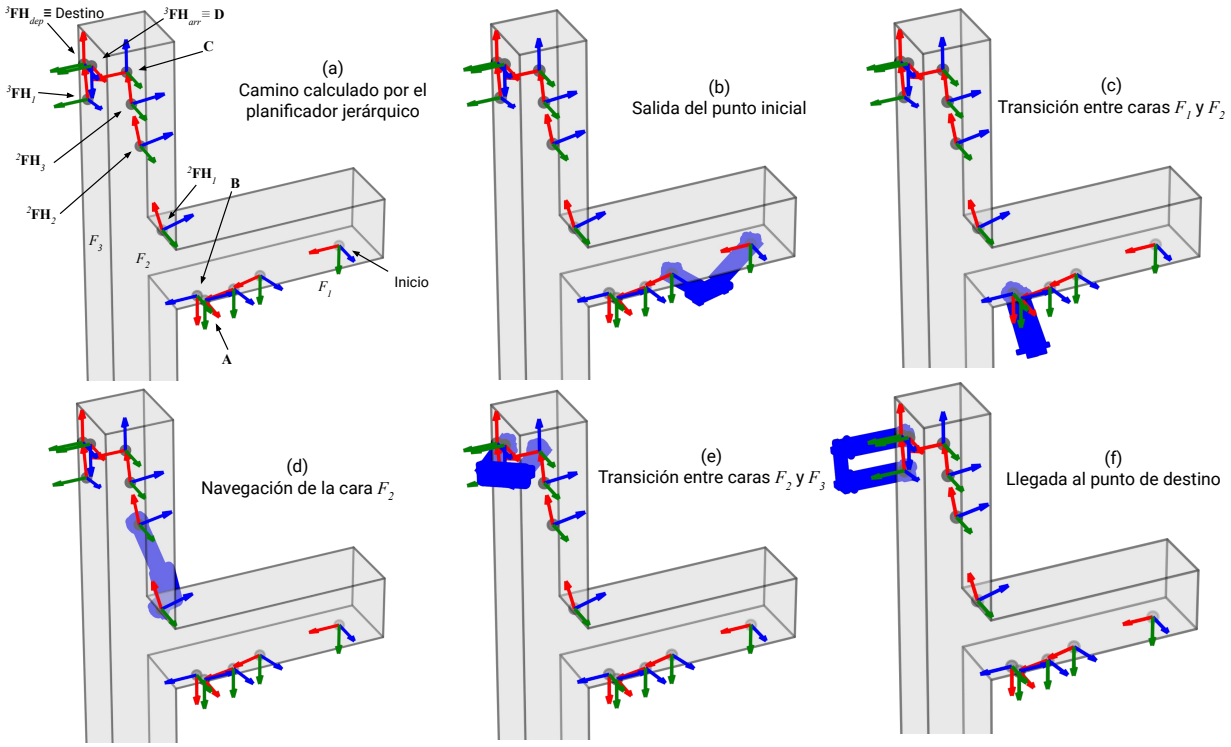

Figura 10: Camino obtenido por el planificador jerárquico (a) e instantes de la simulación (b-f).

pegado necesarios para atravesar la cara F_f. Como ejemplo, en la Figura 10(a), se observa la secuencia de puntos de pegado $\{^2\mathbf{FH}_1, ^2\mathbf{FH}_2, ^2\mathbf{FH}_3\}$ que el robot deberá seguir para recorrer la cara F_2, y en la Figura 10(d) se muestra un instante de la navegación de la misma (movimiento entre $^2\mathbf{FH}_1$ y $^2\mathbf{FH}_2$).

Es visible que, en la última cara F_3 del camino, el punto de salida $^3\mathbf{FH}_{dep}$ de la cara, que corresponde al punto de destino en la estructura, no es alcanzable directamente desde el punto de entrada $^3\mathbf{FH}_{arr}$, a pesar de que se encuentra dentro de una circunferencia con centro en la garra fija y de radio equivalente al alcance máximo (Figura 9). Esto se debe a que el punto no pertenece al espacio de trabajo, ya que se encuentra demasiado próximo a la garra que actúa como base, y ambas garras (la fija y la libre) colisionarían (ver Figura 9). El planificador PPID detecta de manera inteligente que se requiere un punto de pegado intermedio adicional $^3\mathbf{FH}_1$, y lo posiciona a una distancia que equivale al alcance óptimo r_{opt}. De esta manera, a partir del *foothold* $^3\mathbf{FH}_1$, se podrá alcanzar el punto de destino con cualquier orientación, resolviendo el problema y pudiendo llegar al destino, tal y como se representa en la Figura 10(f).

Se han obtenido las estadísticas mostradas en la Tabla 1, calculando el tiempo promedio de ejecución de cada uno de los planificadores a lo largo de 100 repeticiones del experimento.

El preprocesamiento del entorno se realiza de forma previa a la ejecución del algoritmo, lo que permite llevar a cabo esta tarea antes de que el robot sea desplegado en la estructura, sin afectar al rendimiento en tiempo real del algoritmo. El tiempo de ejecución medio del algoritmo completo es de 700,743 milisegundos, siendo 688,478 milisegundos atribuibles al planificador PPT y 12,265 milisegundos a la suma del tiempo empleado por los 3 planificadores PPI. Al tratarse de un tiempo por debajo del segundo, consideramos que el algoritmo es lo suficientemente rápido como para ser empleado en tiempo real. El planificador PPI demuestra ser considerablemente eficiente, generando los caminos en un tiempo muy corto, lo que sugiere que hay poco margen para mejoras significativas en su rendimiento. Por otro lado, el planificador PPT presenta una oportunidad para mejoras, ya que representa el 98,25% del tiempo de ejecución completo. No obstante, puede resultar complicado mejorar esta estadística si se desea mantener la exhaustividad en la búsqueda para garantizar un camino global óptimo o cercano al óptimo.

Tabla 1: Tiempo promedio de ejecución de cada planificador del algoritmo.

Sección	Nivel de la jerarquía	Tiempo (ms)
2.3	Preprocesamiento de la estructura	136,999
3	**Algoritmo completo *(online)***	**700,743**
3.1	Planificador de Puntos de Transición	688,478
3.2	Planificador de Puntos Intermedios (F_1)	3,160
	Planificador de Puntos Intermedios (F_2)	7,792
	Planificador de Puntos Intermedios (F_3)	1,313

4.1. *Discusión*

Con el fin de ofrecer datos estadísticos que permitan comparar el rendimiento del algoritmo cuando se modifican ciertos parámetros, se han realizado experimentos adicionales, cuyos resultados se muestran en la Tabla 2, en los que se modifica un solo parámetro del algoritmo en cada experimento. Se ha mantenido el escenario de la Figura 10, y se ha variado la resolución a la hora obtener los puntos de transición potenciales descritos en la Sección 2.3, así como las resoluciones, tanto de posición como de orientación, del espacio de trabajo plano del robot, que

se emplea en el planificador PPID (Sección 3.2.2). En los experimentos comparativos se ha evaluado el tiempo de ejecución del algoritmo y el número de pasos que da el robot (es decir, el número de pegados que debe hacer la garra del robot a lo largo del camino) para llegar al destino. Se ha escogido como función de mérito dicho número de pasos, que debería reducirse en la medida de lo posible, ya que es una operación donde los robots paso a paso invierten una gran fracción del tiempo total de su locomoción (Peidró et al., 2019), en contraste con los robots de movimiento continuo que no precisan de esta operación.

Según los resultados de la Tabla 2, se observa que, como es lógico, el tiempo de ejecución del algoritmo se ve afectado por la resolución de la discretización empleada. Ya sea para la discretización de los puntos potenciales, o para la discretización del espacio de trabajo plano, una discretización más fina conlleva un incremento en el tiempo de ejecución. Por el contrario, cuando se incrementa el paso de las discretizaciones, el tiempo de ejecución disminuye, pero la calidad de la solución puede verse afectada, i.e., el número de pasos necesarios para recorrer el camino puede aumentar, lo que conllevaría un incremento considerable en el tiempo empleado para que el robot recorra el camino completo. Dado que la calidad de la solución (el número de pasos) no seguirá mejorando por mucho que se afine la resolución (ya que el robot está limitado por su zancada máxima), mientras que el tiempo de cómputo sí aumentará, existen valores óptimos de los parámetros que permiten alcanzar un compromiso entre dicho tiempo y calidad, habiéndose escogido los parámetros originales de la Tabla 2 para lograr dicho compromiso.

Por otra parte, también se han comparado diferentes fases de nuestro algoritmo con el que puede considerarse como el algoritmo referencia en el estado del arte (Zhu et al., 2020). De los dos planificadores principales, únicamente el planificador PPI es comparable, puesto que el planificador PPT de Zhu et al. (2020) conlleva disponer de un test de alcanzabilidad analítico del robot, lo cual no es factible en nuestro caso, tal y como se ha mencionado en la Sección 3.1.

En cuanto al planificador PPI, se ha comparado tanto el tiempo de ejecución como la calidad de la solución obtenida (es decir, el número de pegados que realiza de las garras), para cada uno de los planificadores por los que está compuesto (PPIC y PPID). Se han realizado los experimentos en un nuevo escenario mostrado en la Figura 11.

El enfoque empleado en el planificador PPIC de Zhu et al. (2020) consiste en un algoritmo A*, frente al grafo de visibilidad empleado en nuestro caso. Promediando 100 simulaciones, el tiempo de ejecución del algoritmo A* (con vecindad a 0,1 metros) es de 8,975 milisegundos, con una longitud de camino de 2,7 metros; mientras que el grafo de visibilidad presenta un tiempo de ejecución de $0,944$ milisegundos, con una longitud de 2,631 metros. Por lo tanto, nuestro enfoque resulta en tiempos de ejecución menores y en un camino continuo más corto, como se muestra en la Figura 11(a).

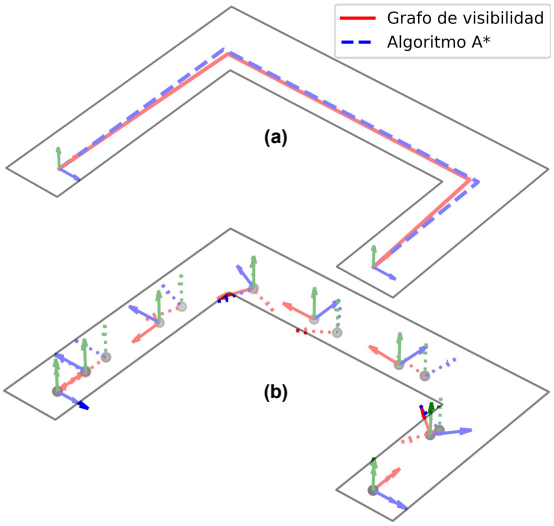

Figura 11: Comparativa planificador PPIC y PPID con diferentes enfoques.

En relación al planificador PPID, el algoritmo que proponemos presenta una mayor elaboración y generalidad para calcular el camino discreto, puesto que resulta más apropiado para robots que, como el robot HyReCRo, no tienen por qué alcanzar todas las orientaciones posibles en todos los puntos de su espacio de trabajo, mientras que el algoritmo empleado por Zhu et al. (2020) asume que el robot puede alcanzar cualquier punto con cualquier orientación. En la Figura 11(b) se muestran las secuencias de puntos de pegado obtenidas por ambos algoritmos, donde los triedros de trazo continuo representan los puntos de pegado obtenidos por nuestro algoritmo, y los triedros a trazos los obtenidos al proceder como Zhu et al. (2020), es decir, tomando simplemente el punto del espacio de trabajo más cercano al destino que corta al camino continuo, sin tener en cuenta la orientación con la que queda la garra al pegarla en dicho punto (que, por lo general, quedará orientada de forma menos favorable para ejecutar ulteriores pasos). El camino obtenido por nuestro algoritmo presenta un menor número de pasos (7 frente a 8), pero un tiempo de cómputo mayor (63,643 frente a 8,299 milisegundos). Este ligero aumento de tiempo de cómputo es preferible si nos ahorramos un paso del robot,

Tabla 2: Comparativa de diferentes parámetros de configuración, para el escenario de la figura 10. Valores medios de 100 repeticiones.

Modificaciones				**Tiempos**							**Calidad**	
Parámetro	**Valor original**	**Incremento**	**Nuevo valor**	**Preprocesamiento**	**PPT**	**PPI (F_1)**	**PPI (F_2)**	**PPI (F_3)**	**TOTAL**	**Mejora**	**Pasos**	**Incremento**
Parámetros originales				136,999 ms	688,478 ms	3,160 ms	7,792 ms	1,313 ms	837,742 ms	-	11	0
Discretización puntos potenciales	0,050 m	0,100 m	0,150 m	84,498 ms	153,899 ms	2,932 ms	989,954 ms	1,121 ms	1.232,404 ms	-32,02 %	13	2
		0,050 m	0,100 m	105,473 ms	294,708 ms	2,910 ms	7,403 ms	1,299 ms	411,793 ms	103,44 %	11	0
		0,020 m	0,070 m	128,936 ms	527,592 ms	2,946 ms	7,539 ms	1,318 ms	668,331 ms	25,35 %	11	0
		-0,020 m	0,030 m	271,241 ms	1.686,992 ms	8,653 ms	13,307 ms	6,081 ms	1.986,274 ms	-57,82 %	11	0
Discretización posición del *WS*	0,010 m	0,040 m	0,050 m	*No converge*								
		0,020 m	0,030 m	136,981 ms	678,223 ms	2,194 ms	7,003 ms	1,106 ms	825,507 ms	1,48 %	12	1
		-0,005 m	0,005 m	135,894 ms	690,109 ms	9,606 ms	27,750 ms	7,992 ms	871,351 ms	-3,86 %	11	0
Discretización orientación del *WS*	5º	15º	20º	134,583 ms	682,909 ms	1,601 ms	6,125 ms	1,198 ms	826,416 ms	1,37 %	12	1
		5º	10º	135,104 ms	688,911 ms	2,912 ms	7,329 ms	1,002 ms	835,258 ms	0,30 %	11	0
		-3º	2º	136,004 ms	694,291 ms	4,928 ms	11,583 ms	3,136 ms	849,942 ms	-1,44 %	11	0

que requiere un tiempo comparativamente mayor para pegar la garra libre en el nuevo punto y despegar la garra anterior, que supone un tiempo de unos 3 segundos (Peidró et al., 2019). Como demuestra este ejemplo, el algoritmo propuesto requiere un menor número de pasos cuando el robot se mueve en caras que, como en la Figura 11(b), tienen varios cambios de dirección. Cabe destacar que, si repetimos esta comparación en caras más simples (e.g., sin cambios de dirección), ambos algoritmos pueden generar el mismo número de pasos, porque, en ese caso, el robot no debe variar mucho su orientación y resulta menos relevante el tener en cuenta la orientación de la garra al pegarse.

5. Conclusiones

A lo largo del artículo se ha presentado un algoritmo jerárquico para la planificación de movimientos de un robot bípedo diseñado para trepar estructuras tridimensionales reticulares. Los distintos niveles de la jerarquía abordan el problema de manera estructurada y secuencial, descomponiendo la complejidad de trabajar en un entorno tridimensional en subproblemas planos. En el proceso de calcular el camino, el primer planificador (PPT) determina la secuencia de caras que el robot atravesará, así como los puntos de entrada y salida de cada una de ellas. Seguidamente, para cada cara atravesada, el planificador PPI calcula el camino para recorrerla. El planificador PPI vuelve a dividir el problema en dos: el planificador PPIC calcula el camino continuo más corto que une el punto de entrada y de salida de la cara, y el planificador PPID calcula el camino discreto (secuencia de pasos) que recorra la cara, usando como guía el camino continuo devuelto por el planificador PPIC.

Se ha validado el correcto funcionamiento del algoritmo en simulación, y se han mostrado estadísticas, imágenes y vídeo de los resultados y rendimientos obtenidos. Se ha estudidado cómo afecta la variación de varios parámetros del algoritmo en el tiempo de ejecución y en la calidad de la solución obtenida, encontrando un compromiso; y se ha comparado el rendimiento de los planificadores PPIC y PPID con los propuestos por Zhu et al. (2020), obteniendo resultados favorables en ambos casos.

En lo que se refiere a futuras líneas de investigación, se planea llevar a cabo experimentos reales con el robot HyReCRo a corto plazo. Además, planeamos desarrollar redes neuronales que proporcionen una solución a la cinemática inversa del robot redundante, en contraste con las propuestas en este artículo, que se enfocan en determinar la alcanzabilidad de un punto, lo que mejoraría el rendimiento y aceleraría el método propuesto. Asimismo, la planificación de movimientos como manipulador, es decir, la determinación de los valores de las variables articulares en cada instante cuando el robot debe mover la garra de un *foothold* al siguiente, es un problema que se pretende abordar en estudios futuros, ya que en el ejemplo de la simulación adjunta se ha usado una interpolación lineal por simplicidad, para animar el movimiento de garras entre los *footholds* calculados.

Agradecimientos

Este trabajo es parte del proyecto PID2020-116418RB-I00, financiado por MCIN/AEI/10.13039/501100011033; y parte de la ayuda PRE2021-099226, financiada por MCIN/AEI/10.13039/501100011033 y por el FSE+.

Referencias

Bentley, J. L., 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM 18 (9), 509–517.

Breitenmoser, A., Siegwart, R., 2012. Surface reconstruction and path planning for industrial inspection with a climbing robot. In: 2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI). IEEE, pp. 22–27.

Chen, W., Gu, S., Guan, Y., Zhang, H., Liu, G., Tang, H., 2016. A multi-layered path planning algorithm for truss climbing with a biped robot. In: 2016 IEEE International Conference on Information and Automation (ICIA). IEEE, pp. 1200–1205.

Chen, X., Wu, Y., Hao, H., Shi, H., Huang, H., 2019. Tracked wall-climbing robot for calibration of large vertical metal tanks. Applied Sciences 9 (13), 2671.

Fabregat-Jaén, M., Peidró, A., Reinoso, Ó., Soler, F.-J., Jiménez, L. M., May 2022. Automatización del pegado de las garras magnéticas de un robot trepador bípedo. In: Jornadas de Robótica, Educación y Bioingeniería - JREB22. Vol. 1. pp. 57–63.

Fang, L., Yang, Q., Yang, T., 2020. Research on path planning algorithm of two-machine cooperative wall climbing and sanding robot based on ant colony algorithm. In: Proceedings of the Seventh Asia International Symposium on Mechatronics: Volume I. Springer, pp. 501–511.

Gimenez, A., Abderrahim, M., Padron, V., Balaguer, C., 2002. Adaptive control strategy of climbing robot for inspection applications in construction industry. IFAC Proceedings Volumes 35 (1), 19–24.

Hart, P. E., Nilsson, N. J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4 (2), 100–107.

Huang, H., Li, D., Xue, Z., Chen, X., Liu, S., Leng, J., Wei, Y., 2017. Design and performance analysis of a tracked wall-climbing robot for ship inspection in shipbuilding. Ocean Engineering 131, 224–230.

Jang, K., An, Y.-K., Kim, B., Cho, S., 2020. Automated crack evaluation of a high-rise bridge pier using a ring-type climbing robot. Computer-Aided Civil and Infrastructure Engineering 36, 14 – 29.

Lee, D.-T., Preparata, F. P., 1984. Euclidean shortest paths in the presence of rectilinear barriers. Networks 14 (3), 393–410.

Nguyen, S. T., La, H. M., 2021. A climbing robot for steel bridge inspection. Journal of Intelligent & Robotic Systems 102, 1–21.

Pagano, D., Liu, D., 2017. An approach for real-time motion planning of an inchworm robot in complex steel bridge environments. Robotica 35 (6), 1280–1309.

Peidró, A., Gil, A., Marín, J. M., Berenguer, Y., Payá, L., Reinoso, O., 2016. Monte-carlo workspace calculation of a serial-parallel biped robot. In: Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 2. Springer, pp. 157–169.

Peidro, A., Gil, A., Marin, J. M., Reinoso, O., 2015. Inverse kinematic analysis of a redundant hybrid climbing robot. International Journal of Advanced Robotic Systems 12 (11), 163.

Peidró, A., Reinoso, Ó., Gil, A., Marín, J. M., Payá, L., 2017. An improved monte carlo method based on gaussian growth to calculate the workspace of robots. Engineering Applications of Artificial Intelligence 64, 197–207.

Peidró, A., Tavakoli, M., Marín, J. M., Reinoso, Ó., 2019. Design of compact switchable magnetic grippers for the hyrecro structure-climbing robot. Mechatronics 59, 199–212.

Prados, C., Hernando, M., Gambao, E., Brunete, A., 2023. Romerin: Organismo robótico escalador basado en patas modulares con ventosas activas. Revista Iberoamericana de Automática e Informática industrial 20 (2), 175–186.

Quin, P., Paul, G., Alempijevic, A., Liu, D., 2016. Exploring in 3d with a climbing robot: Selecting the next best base position on arbitrarily-oriented surfaces. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 5770–5775.

Stumm, E., Breitenmoser, A., Pomerleau, F., Pradalier, C., Siegwart, R., 2012. Tensor-voting-based navigation for robotic inspection of 3d surfaces using lidar point clouds. The International Journal of Robotics Research 31 (12), 1465–1488.

Tavakoli, M., Marques, L., de Almeida, A. T., 2011. 3dclimber: Climbing and manipulation over 3d structures. Mechatronics 21 (1), 48–62.

Yang, C.-h. J., Paul, G., Ward, P., Liu, D., 2016. A path planning approach via task-objective pose selection with application to an inchworm-inspired climbing robot. In: 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, pp. 401–406.

Zhu, H., Lu, J., Gu, S., Wei, S., Guan, Y., 2020. Planning three-dimensional collision-free optimized climbing path for biped wall-climbing robots. IEEE/ASME Transactions on Mechatronics 26 (5), 2712–2723.