XLIV Jornadas de Automática 2023

https://jautomatica.es

Planificación de trayectorias en robots redundantes con mapas de factibilidad y RRT

Fabregat-Jaén, M.[a,*], Peidró, A.[a], Soler, F.J.[a], Gil, A.[a], Reinoso, O.[a,b]

[a] *Instituto de Investigación en Ingeniería de Elche (I3E), Universidad Miguel Hernández de Elche, Avda. de la Universidad s/n, Edificio Innova, 03202, Elche, Alicante, 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-Jaen, M, Peidro, A, Soler, F.J, Gil, A, Reinoso, O. 2023. Motion-planning of redundant robots with feasibility maps and RRT. XLIV Jornadas de Automática, 581-586
https://doi.org/10.17979/spudc.9788497498609.581

Resumen

Los manipuladores redundantes ofrecen múltiples ventajas, tales como manipulabilidad mejorada, o evasión de singularidades u obstáculos. Sin embargo, la redundancia cinemática también introduce retos adicionales, como la necesidad de resolver un problema de cinemática inversa indeterminado. Este artículo presenta un método novedoso para la planificación de trayectorias de manipuladores redundantes, basado en la exploración de mapas de factibilidad. El método propuesto es una extensión del algoritmo Rapidly-exploring Random Trees (RRT), modificado para explorar el espacio redundante de la tarea con la finalidad de encontrar un camino factible subóptimo en el espacio articular, sacrificando optimalidad por escalabilidad a mayor número de grados de redundancia. El método es capaz de seguir una trayectoria dada en el espacio de la tarea, mientras considera otras restricciones, como la evasión de obstáculos o límites articulares. El método se ha validado en simulación.

Palabras clave: Manipuladores redundantes, Planificación de trayectorias, Evasión de obstáculos, Mapas de factibilidad, RRT

Motion-planning of redundant robots with feasibility maps and RRT

Abstract

Redundant manipulators offer multiple advantages, such as improved manipulability, or evasion of singularities and obstacles. However, kinematic redundancy also introduces additional difficulties, such as the need to solve an underdetermined inverse kinematics problem. This article presents a novel method for motion planning of redundant manipulators, based on the exploration of feasibility maps. The proposed method is an extension of the Rapidly-exploring Random Trees (RRT) algorithm, modified to explore the redundant task space with the purpose of finding a feasible suboptimal path in the joint space, sacrificing optimality for scalability to a higher number of degrees of redundancy. The method is capable of following a given trajectory in the task space, while considering other constraints, such as obstacle avoidance or joint limits. The method has been validated under simulation.

Keywords: Redundant manipulators, Motion planning, Obstacle avoidance, Feasibility maps, RRT

1. Introducción

La redundancia cinemática ocurre cuando un manipulador posee más grados de libertad (GDL) que los necesarios para cumplir una tarea. Este aspecto es común en muchos robots, como manipuladores industriales, robots humanoides o robots móviles. La presencia de estos GDL redundantes permite efectuar la tarea de varias formas, lo cual puede ser aprovechado para mejorar el desempeño del robot (Siciliano et al., 2008). Por ejemplo, los grados de redundancia pueden ser utilizados para evitar singularidades (Nakamura and Hanafusa, 1986; Wampler, 1986), mejorar la manipulabilidad (Jin et al., 2017), o evadir obstáculos (Zhang and Wang, 2004).

Sin embargo, la redundancia cinemática también introduce retos adicionales, como la necesidad de resolver un problema de cinemática inversa (CI) indeterminado para controlar el manipulador, lo cual resulta en un número infinito de soluciones para la CI de un manipulador redundante.

La forma más común de solucionar este problema es resol-

*Autor para correspondencia: mfabregat@umh.es

viendo la CI a nivel diferencial, por medio de la pseudoinversión de Moore-Penrose de la matriz Jacobiana (Whitney, 1969), para expresar las velocidades articulares $\dot{\mathbf{q}}$ en función de las velocidades de la tarea $\dot{\mathbf{x}}$. No obstante, este enfoque no asegura la evasión de singularidades cinemáticas, como comprobaron Baillieul et al. (1984). Nakamura and Hanafusa (1986) y Wampler (1986) propusieron emplear un enfoque de mínimos cuadrados amortiguados para resolver la CI, lo que resulta en una Jacobiana no singular en todo el espacio de trabajo.

El aumento del espacio de la tarea, propuesto en (Sciavicco and Siciliano, 1987), es otra forma de resolver el problema de la CI para robots redundantes. Añade parámetros adicionales a la tarea, que son utilizados para satisfacer restricciones adicionales. De esta forma, el vector de la tarea se aumenta hasta igualar el número de GDL del manipulador, lo que resulta en una tarea aumentada para la que el manipulador es no redundante.

Basado en el concepto del aumento del espacio de la tarea, Wenger et al. (1993) presentaron los mapas de factibilidad, que son utilizados para determinar si un manipulador es capaz de cumplir una tarea dada evadiendo obstáculos. En un trabajo posterior (Pámanes G et al., 2002), se utilizan mapas de factibilidad para planear trayectorias libres de colisiones, minimizando el cambio en un parámetro redundante. Este concepto también se expande en (Reveles et al., 2016) para planificar la trayectoria de un robot paralelo redundante mediante transiciones entre modos de trabajo. Sin embargo, la trayectoria no se planea de forma completamente autónoma, ya que se requiere que un operador seleccione una serie de puntos intermedios. En (Ferrentino and Chiacchio, 2019), se realiza una búsqueda exhaustiva en mapas de factibilidad empleando programación dinámica para encontrar la trayectoria óptima global.

En este artículo, presentamos un enfoque novedoso para la planificación de trayectorias de manipuladores redundantes, basado en la exploración de mapas de factibilidad. El método propuesto es una extensión del algoritmo Rapidly-exploring Random Trees (RRT) (LaValle, 1998), modificado para explorar los parámetros redundantes de la tarea para encontrar un camino factible en el espacio articular. Al igual que métodos similares, es capaz de considerar restricciones como límites articulares, autocolosiones u obstáculos. No obstante, en lugar de hacer una búsqueda exhaustiva para encontrar la trayectoria óptima, lo cual consumiría demasiado tiempo en mayores dimensiones, nuestro método emplea un muestreo basado en RRT para encontrar una trayectoria subóptima en el espacio de los parámetros redundantes, sacrificando optimalidad global por eficiencia computacional, lo que otorga buena escalabilidad a un mayor número de grados de redundancia.

El resto del artículo está organizado de la siguiente forma. La Sección 2 revisa el concepto de mapas de factibilidad, el cual se emplea en la Sección 3 para proponer un algoritmo basado en RRT para la planificación de trayectorias de un manipulador redundante. En la Sección 4, el método propuesto se evalúa en un ejemplo simulado. Finalmente, la Sección 5 presenta las conclusiones extraídas y las líneas de trabajo futuras.

2. Mapas de factibilidad

La CI consiste en obtener la configuración articular $\mathbf{q}$ de un manipulador de n GDL, a partir de una tarea $\mathbf{x}$ de m dimensiones, la cual es típicamente la posición y orientación de su efector final. La cinemática directa convierte la configuración articular $\mathbf{q}$ en la posición y orientación del efector final $\mathbf{x}$. Para robots seriales, esta relación se puede escribir como:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{1}$$

Cuando se resuelve la CI, sería deseable poder invertir (1) y obtener $\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x})$, que proporciona la configuración articular en función de la tarea. Sin embargo, en general, no es posible obtener una función inversa global, debido a que, normalmente, para manipuladores no redundantes ($n = m$), una única tarea $\mathbf{x}$ tiene múltiples configuraciones articulares $\mathbf{q}$ que la satisfacen; es decir, la función de la CI es multivaluada. El problema es mayor para manipuladores cinemáticamente redundandes ($n > m$), para los que (1) admite un número infinito de soluciones de $\mathbf{q}$ para una tarea $\mathbf{x}$ dada. El grado de redundancia r se define como la diferencia entre el número de GDL del manipulador y el número de coordenadas de la tarea: $r = n - m$.

Una forma de tratar este problema es utilizar un espacio de tarea aumentado, que consigue que la CI se convierta en un problema determinado y pueda ser resuelto especificando parámetros redundantes de la tarea. Para un manipulador efectuando una tarea $\mathbf{x}$ con r grados de redundancia cinemática, la tarea aumentada $\mathbf{x}_a$ se define como un vector de n dimensiones, para el que el manipulador se convierte en no redundante (Sciavicco and Siciliano, 1989):

$$\mathbf{x}_a = [x_1, \cdots, x_m, x_{m+1}, \cdots, x_{m+r},]^\intercal = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_r \end{bmatrix} \tag{2}$$

donde $\mathbf{x}_r$ es un vector de r dimensiones, cuyos componentes son independientes de todos los demás en $\mathbf{x}_a$, y puede elegirse libremente según las necesidades de la aplicación. El vector de parámetros redundantes $\mathbf{x}_r$ está típicamente compuesto de coordenadas articulares o una función diferenciable $\mathbf{g}(\mathbf{q})$:

$$\mathbf{x}_r = \mathbf{g}(\mathbf{q}) \tag{3}$$

Por ejemplo, las componentes de $\mathbf{x}_r$ pueden ser coordenadas de posición u orientación de un punto de interés del manipulador.

En presencia de múltiples restricciones, como límites articulares u obstáculos que deban ser evadidos, el conjunto de configuraciones factibles que satisfacen la tarea deseada se puede representar en un mapa de $(r + 1)$ dimensiones. La dimensión extra corresponde a la dimensión temporal, en función de la que está definida la tarea, que suele ser la trayectoria que sigue el efector final del robot. Estos mapas se conocen como mapas de factibilidad y fueron introducidos en (Wenger et al., 1993). Un mapa de factibilidad $\mathcal{FM}$ se define como el conjunto de puntos en el espacio $(t, x_{m+1}, \ldots, x_{m+r})$, donde t es el tiempo, para los que existe una configuración articular factible; es decir, que satisface la tarea comandada y las restricciones impuestas.

Para ilustrar el concepto, se considera un manipulador planar de 3 GDL mostrado en la Figura 1, con articulaciones Rotación-Prismática-Rotación (RPR), cuyos eslabones tienen unas longitudes de $l_1 = 0{,}5$ y $l_2 = 1$, y cuya configuración articular viene dada por el vector $\mathbf{q} = [q_1, q_2, q_3]^\intercal$. La tarea deseada $\mathbf{x} = \begin{bmatrix} p_y \end{bmatrix}$ está definida por la posición de la coordenada $\mathbf{Y}$ del efector final p_y a lo largo del tiempo t. La trayectoria deseada está descrita por la función:

$$p_y(t) = -6{,}66227766t^2 + 8{,}16227766t - 1{,}5 \tag{4}$$

con t en el intervalo [0, 1]. Este escenario resulta en un manipulador con $r = 2$ grados de redundancia cinemática para la tarea dada. Por lo tanto, la tarea aumentada se puede definir añadiendo dos parámetros redundantes al vector de la tarea $\mathbf{x}$. En este ejemplo, se consideran como parámetros redundantes las dos primeras articulaciones del manipulador $\{q_1, q_2\}$, resultando en el vector de la tarea aumentada $\mathbf{x}_a$:

$$\mathbf{x}_a = \begin{bmatrix} p_y \\ q_1 \\ q_2 \end{bmatrix} \longrightarrow \begin{array}{l} \mathbf{x} = \mathbf{f}(\mathbf{q}) = l_1 \sin(q_1) + l_2 \sin(q_1 + q_3) \\ \mathbf{x}_r = \mathbf{g}(\mathbf{q}) = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \end{array} \tag{5}$$

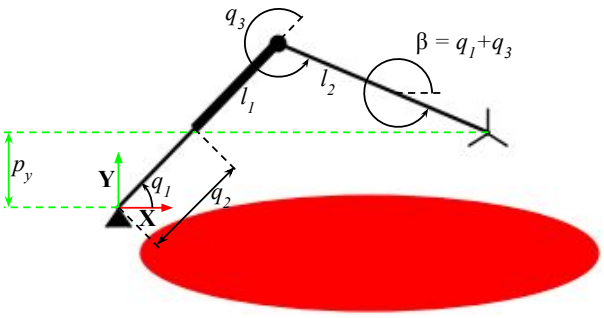

Figura 1: Manipulador RPR con un obstáculo elíptico.

Para resolver la CI, el resto de coordenadas articulares del manipulador deben ser expresadas en función de las coordenadas de la tarea aumentada. En general, esto requiriría resolver (1) y (3) a la vez. Para este ejemplo en particular, esto se reduce a resolver q_3 en función de p_y, q_1 y q_2:

$$q_3 = \beta - q_1, \quad \text{con} \quad \beta := \arcsin\left(\frac{p_y - (l_1 + q_2)\sin(q_1)}{l_2}\right) \tag{6}$$

donde l_1 y l_2 son longitudes fijas de las barras, como muestra la Figura 1. La solución (6) es válida cuando β (la suma de los ángulos q_1 y q_3) se encuentra en el primer o cuarto cuadrante. En caso contrario, la solución sería: $q_3 = \pi - \beta - q_1$.

El entorno donde trabaja el manipulador en este ejemplo se muestra en la Figura 1. La elipse roja representa una región prohibida para el efector final, y está definida por:

$$(p_x - 1{,}1)^2/1^2 + (p_y - 0{,}2)^2/0{,}25^2 \leq 1 \tag{7}$$

donde p_x y p_y son las coordenadas del efector final en el sistema de referencia de la base, identificado por los ejes **X** e **Y** en la Figura 1. La elipse es un obstáculo con el cual el efector final no debe colisionar (p_x y p_y no deben satisfacer (7)), pero el resto del manipulador puede intersecar con ella.

La Figura 2 muestra el mapa de factibilidad obtenido para este ejemplo, que tiene 3 dimensiones. Los ejes representados corresponden a la dimensión temporal t y a los dos parámetros redundantes q_1 y q_2 del vector de la tarea adicional $\mathbf{x}_r$. La Figura 2 muestra varias superficies de colores, que encierran regiones prohibidas. La roja encierra configuraciones donde el efector invade la elipse definida en (7), y la morada encierra configuraciones donde la CI no tiene solución real. El mapa de factibilidad $\mathcal{FM}$ es el volumen que excluye dichas regiones prohibidas. Cabe destacar que, aunque se han precalculado dichas regiones prohibidas para mostrarlas en la Figura 2, el algoritmo presentado en la siguiente sección no requiere precalcularlas.

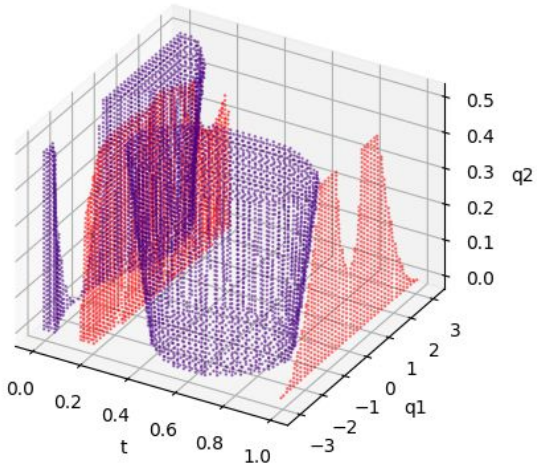

Figura 2: Mapa de factibilidad del manipulador RPR para la tarea descrita.

3. Algoritmo de planificación de trayectorias propuesto

Los mapas de factibilidad recogen todos los puntos factibles en el espacio $(t, \mathbf{x}_r)$ que satisfacen restricciones especificas para cumplir una tarea dada. Explorando los mapas de factibilidad, es posible determinar una trayectoria continua en el espacio redundante $\mathbf{x}_r$ a lo largo del tiempo, para una tarea descrita $\mathbf{x}(t)$ parametrizada en función de t, que típicamente representa el tiempo, o un parámetro de longitud de arco.

Una trayectoria o camino factible $\mathcal{P}$ en el espacio redundante $\mathbf{x}_r$ se puede establecer si existe un conjunto de puntos, conectados de forma continua, que unan la configuración inicial y una configuración deseada. En otras palabras, un camino factible $\mathcal{P} = \{\mathbf{P}_{start}, \mathbf{P}_1, \ldots, \mathbf{P}_N, \mathbf{P}_{end}\}$ se puede determinar si existe un conjunto de puntos $\mathbf{P}_i = [t_i, \mathbf{x}_r^\intercal]$ tales que el segmentos entre $\mathbf{P}_{i-1}$ y $\mathbf{P}_i$ está en $\mathcal{FM}$ para $i = 1 \ldots end$, donde $t_i > t_{i-1}$, $\mathbf{P}_{start}$ es el punto inicial y $\mathbf{P}_{end}$ es el punto final.

Cabe aclarar que el vector de la tarea aumentada $\mathbf{x}_a$ está definido de forma inequívoca para cada punto $\mathbf{P}$ del mapa de factibilidad, ya que $\mathbf{x}_a$ engloba la tarea deseada $\mathbf{x}$ (la cual depende únicamente de t) y los parámetros redundantes $\mathbf{x}_r$, que, junto con t, definen el espacio dimensional del mapa de factibilidad. Por lo tanto, la configuración articular $\mathbf{q}$ asociada a cualquier punto $\mathbf{P}$ del mapa de factibilidad se puede calcular en cualquier momento resolviendo (1) y (3).

La planificación de trayectorias es un problema fundamental de la robótica, y existen numerosos métodos para resolverlo. Por un lado, los algoritmos basados en búsqueda de grafos, como Dijkstra, A* o sus variantes, son capaces de encontrar la solución óptima en términos de una función heurística, la cual guía la búsqueda. Sin embargo, a medida que la dimensionalidad del problema aumenta, el tiempo de cómputo se vuelve prohibitivamente largo.

Por el contrario, los algoritmos basados en muestrear el espacio de búsqueda, como Probabilistic Roadmaps (PRM) (Kavraki et al., 1996) o Rapidly-exploring Random Trees (RRT) (LaValle, 1998), pueden identificar un camino factible de forma más eficiente, independientemente de la dimensionalidad del espacio, pero no garantizan la optimalidad de la solución.

RRT es un popular algoritmo de planificación de trayectorias que explora un espacio de búsqueda dado, y construye un

árbol muestreando aleatoriamente puntos de dicho espacio y estableciendo conexiones entre sí. Empezando desde un estado inicial $\mathbf{P}_{start}$, el algoritmo genera un árbol $\mathcal{T}$ añadiendo iterativamente nuevos nodos al mismo, hasta que se encuentra un camino que alcanza un estado final $\mathbf{P}_{end}$.

RRT fue introducido en (LaValle, 1998) como una estructura de datos aleatoria para la planificación de trayectorias en espacios de búsqueda de alta dimensionalidad. Desde entonces, se han propuesto multitud de variaciones del algoritmo original para abordar diferentes problemas. Por ejemplo, RRT-Connect (Kuffner and LaValle, 2000) extiende el algoritmo original para permitir la búsqueda bidireccional entre los puntos inicial y final. En (Bruce and Veloso, 2002), ERRT se propone para adecuar el algoritmo RRT a aplicaciones en tiempo real. Dynamic RRT (Ferguson et al., 2006) proporciona un método para replanificar la trayectoria cuando el entorno es cambiante. RRT* (Karaman and Frazzoli, 2011) y Anytime RRT (Karaman et al., 2011) son extensiones que garantizan la optimalidad de la trayectoria encontrada cuando se dispone de suficiente tiempo.

3.1. *Algoritmo RRT modificado*

En esta subsección, se propone un algoritmo RRT modificado, que explora los mapas de factibilidad de un robot redundante para determinar un camino factible considerando restricciones. Es una adaptación del algoritmo RRT original, con modificaciones para tener en cuenta las particularidades del problema.

La diferencia principal radica en que, en nuestra aplicación, la configuración final ($\mathbf{P}_{end}$) puede ser cualquiera mientras se cumpla la tarea comandada (la cual se cumplirá siempre que $t = t_{end}$). Por lo tanto, el algoritmo busca un camino factible que alcance el subconjunto objetivo $\mathcal{G} = \{[t, \mathbf{x}_r^\intercal] \in \mathcal{FM} : t = t_{end}\}$, independientemente de la configuación final $\mathbf{x}_r$, la cual no es relevante mientras sea factible. De entre el conjunto de caminos factibles encontrados, el algoritmo seleccionará el que minimice una función de coste $c(\mathbf{P})$, la cual será definida más adelante.

Además, a diferencia de la mayoría de algoritmos basados en RRT (Noreen et al., 2016), el algoritmo propuesto hace la planificación de forma online. Es decir, el algoritmo no depende de un mapa predefinido del espacio de búsqueda, sino que el mapa se construye a medida que se explora el espacio. El Algoritmo 1 muestra el pseudocódigo del algoritmo propuesto. En la Figura 3 se ilustran los pasos más importantes del código, que serán descritos a lo largo de los siguientes párrafos.

Algoritmo 1: Algoritmo RRT modificado.

```
1  T ← InicializarArbol(P_start)
2  para i = 1, 2, ..., i_max hacer
3  |  P_rand ← NodoAleatorio
4  |  P_parent ← MejorPadre(P_rand)
5  |  si condiciones (C1), (C2) y (C3) entonces
6  |  |  Añadir nodo P_rand con padre P_parent a T
7  |  |  P_ext ← ExtendPath(P_parent, P_rand)
8  |  |  si condiciones (C2) y (C3) entonces
9  |  |  |  añadir nodo P_ext con padre P_rand a T
10 P_best ← MejorCamino(T, c(P))
11 P_smooth ← SuavizarCamino(P_best)
12 devolver P_smooth
```

El algoritmo comienza inicializando el árbol $\mathcal{T}$ con el estado inicial $\mathbf{P}_{start}$. El árbol es una estructura de datos que establece una relación jerárquica entre sus nodos, cada uno de los cuales corresponde a un estado o punto $\mathbf{P} \in \mathcal{FM}$. Cada nodo puede tener varios descendientes pero solo un padre. El nodo inicial $\mathbf{P}_{start}$ es la raíz del árbol y no tiene padre.

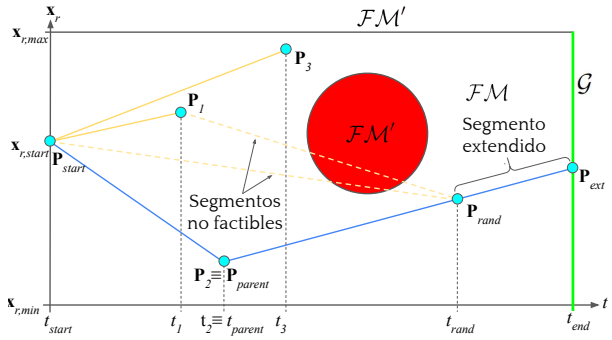

Figura 3: Algoritmo RRT modificado, donde $\mathcal{FM}'$ es el complemento de $\mathcal{FM}$.

A continuación, el algoritmo itera i_{max} veces, donde i_{max} es un número máximo de iteraciones predefinido. En cada iteración, se genera un nodo aleatorio $\mathbf{P}_{rand}$, garantizando su factibilidad al estar contenido en $\mathcal{FM}$. El algoritmo busca el mejor padre $\mathbf{P}_{parent}$ para el nodo aleatorio $\mathbf{P}_{rand}$ (explicaremos seguidamente qué significa ser el mejor padre).

El proceso de búsqueda del mejor padre debe respetar la condición de que el tiempo es estrictamente monótono creciente, por lo que la coordenada de tiempo t del nodo aleatorio $\mathbf{P}_{rand}$ debe ser mayor que la de su padre $\mathbf{P}_{parent}$ ($t_{parent} < t_{rand}$). La búsqueda se ejecuta hacia adelante en la dimensión del tiempo, comenzando desde el nodo raíz, hasta que se encuentra un nodo $\mathbf{P}_{parent}$ tal que se cumplen 3 condiciones: (C1) $t_{parent} < t_{rand}$, (C2) el segmento entre $\mathbf{P}_{parent}$ y $\mathbf{P}_{rand}$ pertenece completamente a $\mathcal{FM}$ (es decir, no interseca con regiones prohibidas), y (C3) se cumplen otros requisitos, que pueden ser especificados dependiendo de la aplicación. Por ejemplo, un requisito especificado puede ser que, cuando alguna componente de $\mathbf{x}_r$ es q_i, su velocidad $\dot{q}_i$ esté acotada:

$$\dot{q}_{i,min} \leq \frac{q_{i,rand} - q_{i,parent}}{t_{rand} - t_{parent}} \leq \dot{q}_{i,max} \tag{8}$$

donde t_j y $q_{i,j}$ son coordenadas del nodo $\mathbf{P}_j$.

La tarea del algoritmo más exigente computacionalmente es verificar la condición (C2) (si el segmento entre ambos nodos está contenido en $\mathcal{FM}$). Debido a que el mapa de factibilidad no está precalculado, la verificación debe realizarse online. La factibilidad del segmento entre $\mathbf{P}_{parent}$ y $\mathbf{P}_{rand}$ se comprueba discretizándolo con una resolución preestablecida en el eje t, y verificando si cada punto discretizado pertenece a $\mathcal{FM}$. Para acelerar el proceso, los puntos discretizados se comprueban en orden aleatorio siguiendo una distribución uniforme continua. De esta forma se aumenta la probabilidad de identificar un punto no factible en el segmento de forma temprana y resulta en una rechazo más rápido del segmento en caso de que sea inválido. Si el segmento se considera válido, se añade el nodo $\mathbf{P}_{rand}$ con padre $\mathbf{P}_{parent}$ al árbol y se extiende el segmento.

El proceso de extensión consiste en extender el segmento previamente verificado entre $\mathbf{P}_{parent}$ y $\mathbf{P}_{rand}$, hasta un nodo extendido $\mathbf{P}_{ext}$, que corresponde a la intersección entre dicho segmento extendido y el subconjunto objetivo $\mathcal{G}$. El segmento en-

tre $\mathbf{P}_{rand}$ y $\mathbf{P}_{ext}$ también debe ser verificado, comprobando las condiciones (C2) y (C3) de la forma explicada (no es necesario comprobar (C1) puesto que $t_{ext} = t_{end}$). Si el segmento extendido es factible, se añade el nodo $\mathbf{P}_{ext}$ con padre $\mathbf{P}_{rand}$ al árbol, encontrando así un camino factible completo $\mathcal{P}$.

En la Figura 3, se muestra una iteración del algoritmo en un escenario de ejemplo donde el mapa de factibilidad está acotado por t_{start} y t_{end} en la dimensión del tiempo, $\mathbf{x}_{r,min}$ y $\mathbf{x}_{r,max}$ en la dimensión de los parámetros redundantes, y una región roja que representa un obstáculo. En la iteración del instante mostrado, el árbol contiene cuatro nodos: $\mathbf{P}_{start}$, $\mathbf{P}_1$, $\mathbf{P}_2$, y $\mathbf{P}_3$.

En la iteración mostrada, se genera el nodo aleatorio $\mathbf{P}_{rand}$ y se busca el mejor padre $\mathbf{P}_{parent}$. El segmento entre $\mathbf{P}_{start}$ y $\mathbf{P}_{rand}$ se comprueba primero, y se rechaza al cruzar la región roja, que no pertenece a $\mathcal{FM}$. El resto de nodos se comprueban en el orden determinado por t, hasta que $\mathbf{P}_2$ se identifica como el mejor padre $\mathbf{P}_{parent}$. Cabe destacar que, en el momento que se identifica $\mathbf{P}_2$ como el mejor padre, $\mathbf{P}_3$ aún no ha sido comprobado, puesto que $t_3 > t_2$. A continuación, el segmento entre $\mathbf{P}_{parent}$ y $\mathbf{P}_{rand}$ se extiende hasta $\mathbf{P}_{ext} \in \mathcal{G}$, y se comprueba su factibilidad, resultando en un camino completo $\mathcal{P}$, que conecta $\mathbf{P}_{start}$ y $\mathcal{G}$.

Cuando se alcanza el número máximo de iteraciones i_{max}, el algoritmo evalúa el árbol y elige, de entre todos los caminos completos encontrados, el mejor camino $\mathcal{P}_{best}$, empleando una función de coste $c(\mathcal{P})$. Esta función evalúa el coste del camino $\mathcal{P}$, y puede ser elegida para cada problema en particular. Por ejemplo, una elección sencilla sería minimizar la suma de la norma ponderada de los segmentos que forman cada camino:

$$c(\mathcal{P}) = \sum_{i=1}^{end} \sqrt{(\mathbf{P}_i - \mathbf{P}_{i-1})\ \mathbf{W}\ (\mathbf{P}_i - \mathbf{P}_{i-1})^\intercal} \qquad (9)$$

donde $\mathbf{W}$ es una matriz diagonal de pesos w_i.

Debido a la naturaleza del algoritmo, el camino devuelto es poligonal, lo que implica velocidades discontinuas y aceleraciones infinitas. Para suavizar el camino, se interpola el mejor camino $\mathcal{P}_{best}$ con una interpolación con B-splines cúbicos (grado 3) (De Boor and De Boor, 1978).

Dado un camino $\mathcal{P} = \{\mathbf{P}_{start}, \mathbf{P}_1, \ldots, \mathbf{P}_N, \mathbf{P}_{end}\}$, formado por el nodo inicial, final, y una serie de N nodos, se establece un conjunto de puntos de control C a lo largo del camino:

$$C = \{{}^1\mathbf{C}_1, {}^1\mathbf{C}_2, \ldots, {}^1\mathbf{C}_{Nc}, {}^2\mathbf{C}_1, {}^2\mathbf{C}_2, \ldots {}^2\mathbf{C}_{N_C}, \ldots\} \qquad (10)$$

donde N_c es el número de puntos de control por segmento (por ejemplo, desde $\mathbf{P}_{start}$ hasta $\mathbf{P}_1$). El punto de control ${}^i\mathbf{C}_j$ es el j-ésimo punto de control del segmento i-ésimo que forma el camino $\mathcal{P}$. Los puntos de control en cada segmento están espaciados regularmente. La notación seguida, los puntos de control resultantes y el camino suavizado devuelto se muestran en un ejemplo en la Figura 4.

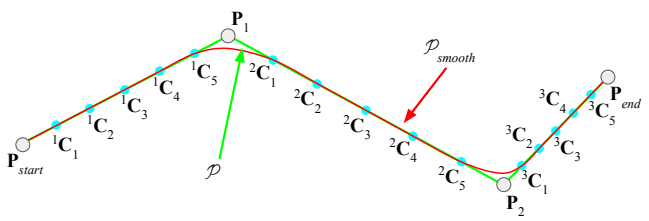

Figura 4: Suavizado $\mathcal{P}_{smooth}$ (en rojo) del camino $\mathcal{P}$ (en verde), con $N_C = 5$.

4. Simulaciones

Para validar el algoritmo propuesto, se ha simulado el escenario mostrado en la Figura 1, donde se requiere que el efector final del manipulador RPR siga la trayectoria en la coordenada $\mathbf{Y}$ definida por (4), mientras se evita posicionar el efector final dentro del obstáculo elíptico rojo definido en (7). La configuración inicial es $\mathbf{q} = [-0{,}6984\ \text{rad}, 0{,}5\ \text{m}, \ -0{,}331\ \text{rad}]^\intercal$.

Las articulaciones del manipulador están restringidas en el intervalo $[-2\pi, 2\pi]$ rad para q_1 y q_3 (lo cual permite *wrapping* del ángulo en $\pm\pi$), y $[0, 0{,}5]$ m para q_2. Además, se ha impuesto una restricción en la velocidad máxima de las articulaciones de 13 rad/s y 0,2 m/s para q_1 y q_3, y q_2, respectivamente. La ecuación (9) se emplea como función de coste $c(\mathcal{P})$, con una matriz de pesos identidad $\mathbf{W} = \mathbf{I}$.

El algoritmo se ejecuta con un número máximo de i_{max} iteraciones, lo que significa que se generarán i_{max} nodos aleatorios, sin contar los nodos extendidos. Para el suavizado del camino, se emplean $N_C = 6$ puntos de control, por segmento del camino completo, para la interpolación con B-splines.

Las simulaciones se han realizado en Python, con un procesador Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz. En la Tabla 1 se muestran los resultados obtenidos para diferentes valores de i_{max} (seleccionados arbitrariamente para representar el espectro completo), promediados a lo largo de 500 simulaciones para cada i_{max}. Se puede observar que el algoritmo presenta una tasa de fallo cuando se generan pocos nodos. La tasa de fallo es la probabilidad de que el algoritmo no encuentre un camino factible, lo cual deja de suceder a partir de $i_{max} = 1600$.

Tabla 1: Tiempos, costes, y tasas de fallo para diferentes i_{max}

i_{max}	**Tiempo ejecución (s)**	**Coste** $c(\mathcal{P}_{best})$	**% fallo**
100	0,040	5,967	73,6
500	0,230	4,743	11,2
1000	0,490	4,182	1,8
1500	0,855	3,845	0,8
2000	1,260	3,709	0
2100	**1,367**	**3,642**	**0**
2500	1,760	3,554	0
3000	2,300	3,480	0
3500	2,959	3,438	0

Para evaluar los resultados obtenidos y elegir el número de iteraciones i_{max} más adecuado, en la Figura 5 se ha representado el coste medio del camino obtenido frente al tiempo medio de ejecución, para diferentes valores de i_{max} (desde 100 hasta 5900, con un incremento de 100). Para seleccionar el valor de i_{max} más adecuado, se ha buscado el punto de la curva que minimice la circunferencia que pase por dicho punto y esté centrada en el origen (en rojo en la Figura 5). De entre las configuraciones simuladas, se ha elegido $i_{max} = 2100$ como el valor más adecuado, el cual presenta un buen compromiso entre tiempo de ejecución y coste del camino obtenido.

Finalmente, en la Figura 6 se muestra el camino suavizado $\mathcal{P}_{smooth}$ obtenido para el número de iteraciones elegido $i_{max} = 2100$. Se muestra el árbol $\mathcal{T}$ generado por el algoritmo sobre el mapa de factibilidad $\mathcal{FM}$ correspondiente. En rojo claro se muestran los caminos que no alcanzan el subconjunto

objetivo $\mathcal{G}$, mientras que en verde se dibujan los que sí alcanzan el objetivo. En azul se muestra el camino suavizado $\mathcal{P}_{smooth}$ obtenido por el algoritmo. Es un camino continuo, pero en el punto $\mathbf{P}_{wrap}$, q_1 sufre *wrapping* de $-\pi$ a π.

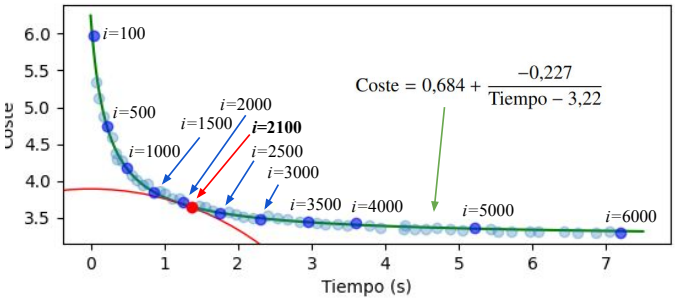

Figura 5: Coste frente a tiempo para diferentes valores de i_{max}.

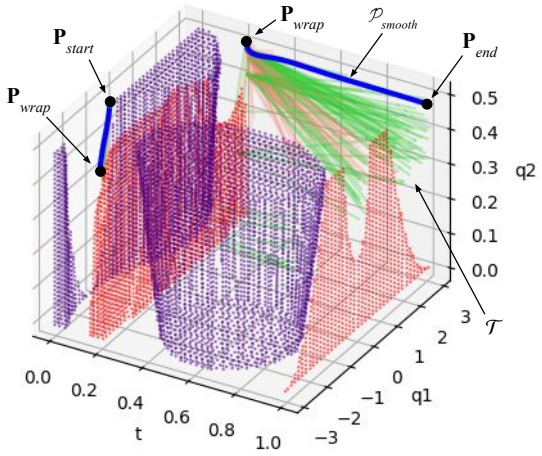

Figura 6: Árbol generado y camino suavizado obtenido por el algoritmo RRT.

5. Conclusiones

Este artículo ha presentado un algoritmo novedoso para la planificación de trayectorias de manipuladores redundantes basado en el algoritmo RRT. El método propuesto muestrea aleatoriamente puntos en el mapa de factibilidad de la tarea, sujeto a ciertas restricciones, y construye iterativamente un árbol conectando los puntos a nodos anteriores en el árbol. Los caminos que llegan al objetivo se evalúan en función de una función de coste y se selecciona el que la minimiza. El camino seleccionado se suaviza utilizando una interpolación B-spline, y se devuelve como la trayectoria a seguir por el manipulador. El algoritmo ha sido evaluado en un manipulador planar RPR y es capaz de generar caminos factibles de manera eficiente.

Trabajos futuros implicarán extender el algoritmo a manipuladores con un mayor número de grados de redundancia, lo que requerirá explorar mapas de factibilidad de mayor dimensionalidad, para los cuales las búsquedas exhaustivas del mapa requerirían una cantidad prohibitiva de tiempo. Además, se probará el método en manipuladores reales.

Agradecimientos

Este trabajo es parte del proyecto PID2020-116418RB-I00, financiado por MCIN/AEI/10.13039/501100011033; de la ayuda PRE2021-099226, financiada por MCIN/AEI /10.13039/501100011033 y por el FSE+; y del proyecto CIGE/2021/177, financiado por la Conselleria d'Innovació, Universitats, Ciència i Societat Digital.

Referencias

Baillieul, J., Hollerbach, J., Brockett, R., 1984. Programming and control of kinematically redundant manipulators. In: The 23rd IEEE Conference on Decision and Control. IEEE, pp. 768–774.

Bruce, J., Veloso, M., 2002. Real-time randomized path planning for robot navigation. In: IEEE/RSJ international conference on intelligent robots and systems. Vol. 3. IEEE, pp. 2383–2388.

De Boor, C., De Boor, C., 1978. A practical guide to splines. Vol. 27. Springer-verlag New York.

Ferguson, D., Kalra, N., Stentz, A., 2006. Replanning with rrts. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. IEEE, pp. 1243–1248.

Ferrentino, E., Chiacchio, P., 2019. A topological approach to globally-optimal redundancy resolution with dynamic programming. In: ROMANSY 22 - Robot Design, Dynamics and Control. Springer, pp. 77–85.

Jin, L., Li, S., La, H. M., Luo, X., 2017. Manipulability optimization of redundant manipulators using dynamic neural networks. IEEE Transactions on Industrial Electronics 64 (6), 4710–4720.

Karaman, S., Frazzoli, E., 2011. Sampling-based algorithms for optimal motion planning. The international journal of robotics research 30 (7), 846–894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., Teller, S., 2011. Anytime motion planning using the rrt. In: 2011 IEEE international conference on robotics and automation. IEEE, pp. 1478–1483.

Kavraki, L. E., Svestka, P., Latombe, J.-C., Overmars, M. H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation 12 (4), 566–580.

Kuffner, J. J., LaValle, S. M., 2000. Rrt-connect: An efficient approach to single-query path planning. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 2. IEEE, pp. 995–1001.

LaValle, S. M., 1998. Rapidly-exploring random trees : a new tool for path planning. The annual research report.

Nakamura, Y., Hanafusa, H., 09 1986. Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. Journal of Dynamic Systems, Measurement, and Control 108 (3), 163–171.

Noreen, I., Khan, A., Habib, Z., 2016. Optimal path planning using rrt* based approaches: A survey and future directions. International Journal of Advanced Computer Science and Applications 7 (11).

Pámanes G, J. A., Wenger, P., Zapata D, J. L., 2002. Motion planning of redundant manipulators for specified trajectory tasks. Advances in Robot Kinematics: Theory and Applications, 203–212.

Reveles, D., Wenger, P., et al., 2016. Trajectory planning of kinematically redundant parallel manipulators by using multiple working modes. Mechanism and Machine Theory 98, 216–230.

Sciavicco, L., Siciliano, B., 1987. Solving the inverse kinematic problem for robotic manipulators. In: RoManSy 6: Proceedings of the Sixth Symposium on Theory and Practice of Robots and Manipulators. Springer, pp. 107–114.

Sciavicco, L., Siciliano, B., 1989. The augmented task space approach for redundant manipulator control. In: Robot Control 1988 (Syroco'88). Elsevier, pp. 125–129.

Siciliano, B., Khatib, O., Kröger, T., 2008. Springer handbook of robotics. Vol. 200. Springer.

Wampler, C. W., 1986. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. IEEE Transactions on Systems, Man, and Cybernetics 16 (1), 93–101.

Wenger, P., Chedmail, P., Reynier, F., 1993. A global analysis of following trajectories by redundant manipulators in the presence of obstacles. [1993] Proceedings IEEE International Conference on Robotics and Automation, 901–906 vol.3.

Whitney, D. E., 1969. Resolved motion rate control of manipulators and human prostheses. IEEE Transactions on man-machine systems 10 (2), 47–53.

Zhang, Y., Wang, J., 2004. Obstacle avoidance for kinematically redundant manipulators using a dual neural network. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (1), 752–759.