Aplicación de algoritmos RRT en la planificación de movimientos óptimos en robótica[1]

Néstor García, Jan Rosell, Raúl Suárez

Instituto de Organización y Control de Sistemas Industriales (IOC),
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech,
Av. Diagonal 647, planta 11, 08028 Barcelona, Spain
{nestor.garcia.hidalgo, jan.rosell, raul.suarez}@upc.edu

Abstract

La planificación de movimientos es un área de investigación básica en robótica, sobre todo desde que los robots se convirtieron en una parte esencial en muchos campos de aplicación tales como, por ejemplo, las industrias médica y electrónica, e incluso la biología computacional o la animación por ordenador. La importancia de este problema se hace evidente cuando se intenta planificar una trayectoria para un sistema robótico con un gran número de grados de libertad (DOF), como es el caso de las manos mecánicas o los sistemas antropomorfos. Además, a veces no sólo se requiere que la trayectoria esté libre de colisiones, sino también que optimice alguna función de coste (por ejemplo, el tiempo de ejecución o su longitud). El algoritmo *Rapidly-Exploring Random Trees* (RRT) es ampliamente utilizado debido a su elevada eficacia para planificar movimientos incluso para sistemas con muchos DOF o restricciones. Además, variantes del RRT permiten encontrar trayectorias óptima (o cercanas a la óptima), acorde a una medida de calidad dada. El objetivo de este trabajo es estudiar el estado de la técnica en la planificación de movimientos y evaluar diferentes planificadores basados en el algoritmo RRT para obtener trayectorias óptimas de sistemas robóticos.

1 Introducción

Durante más de tres décadas, ha existido un gran interés en el desarrollo de algoritmos de planificación de movimientos para generar trayectorias sin colisiones que permitan a un robot moverse de una configuración inicial a una configuración final [13]. La planificación de movimientos se considera un problema importante en robótica ya que su complejidad aumenta exponencialmente con el número de grados de libertad del robot. Existen varias técnicas para la planificación de trayectorias y, considerando los enfoques más relevantes, se puede hacer la siguiente clasificación:

a) Métodos clásicos que o bien encuentran una solución o aseguran que no existe ninguna. La principal desventaja de estos métodos es su complejidad computacional y su incapacidad para hacer frente a la incertidumbre. Tales desventajas hacen que su uso no sea práctico en aplicaciones reales. Dentro de este grupo se encuentran los siguientes métodos [13]:
   - Métodos "mapa de carreteras" que capturan el espacio libre utilizando un mapa de carreteras al que se conectan las configuraciones inicial y final, obteniendo la trayectoria solución como una trayectoria dentro de ese mapa.
   - Métodos de descomposición de celdas que particionan el espacio de configuraciones libre en polígonos convexos más pequeños y la trayectoria solución se encuentra mediante una búsqueda en el grafo que conecta estos polígonos.
   - Técnicas de campos de potenciales que atraen al robot hacia la configuración final y lo alejan de los obstáculos presentes en la escena combinando distintos campos de fuerza.

b) Métodos basados en muestreo que evitan la construcción explícita del espacio de configuraciones al representar la conectividad del espacio libre de colisiones con un grafo de configuraciones muestreadas. Se han desarrollado distintas estrategias generales muy eficaces utilizando técnicas basadas en muestreo, siendo los enfoques más relevantes los planificadores *Rapidly-Exploring*

[1]Este trabajo ha sido parcialmente financiado por el Gobierno de España a través de los proyectos DPI2013-40882-P, DPI2014-57757-R y DPI2016-80077-R. El trabajo de N. García ha sido financiado por la Generalitat de Catalunya a través de la beca FI-DGR 2016.

Random Trees (RRT) [12] y los *Probabilistic Roadmaps* (PRM) [10]. Sin embargo, estos enfoques tienen algunos problemas cuando, por ejemplo, existen pasajes estrechos en el espacio de configuraciones o se debe tratar con restricciones [21]. Por ello, se han presentado diversas variantes que proponen diferentes mejoras, como por ejemplo, el sesgo del muestreo hacia regiones más prometedoras del espacio de configuraciones utilizando dominios dinámicos [23], retracción del grafo de muestras [24], el sesgo adaptativo del espacio de trabajo [25], un muestreo basado en el Análisis de Componentes Principales (PCA) [18] o en sinergias (i.e. correlación entre distintos DOF del sistema robótico) [4]. Todos estos algoritmos no toman en cuenta ninguna medida de calidad durante la búsqueda de una trayectoria, al contrario que otros enfoques como, por ejemplo, los algoritmos *Transition-based RRT* (T-RRT) [7] y *Vector-Field RRT* (VF-RRT) [11], pero que sin embargo no garantizan que la trayectoria encontrada sea óptima. Para encontrar realmente una solución que optimice una función de coste dada, se han propuesto algunas variantes como los algoritmos RRT$^{\#}$, RRT* y PRM* [2, 9].

c) Métodos basados en optimización que tratan el problema de la planificación de movimientos como si fuera un problema de optimización numérica. Estos algoritmos parten de una trayectoria (o varias [14]) que puede no estar libre de colisiones o puede no cumplir las restricciones de movimiento del robot. Entonces se intenta converger hacia una trayectoria que por un lado maximice una medida de calidad especificada y, por otro, esté libre colisiones y satisfaga las restricciones. Sin embargo, las funciones de coste que optimizan estos métodos suelen tener un gran número de mínimos locales, por lo que encontrar la solución global depende en gran medida de la trayectoria inicial. Algunos de estos enfoques basados en técnicas de optimización son los algoritmos CHOMP [15], STOMP [8] y TrajOpt [20].

Después de esta introducción, en la Sección 2 se formaliza el problema de la planificación de movimientos y se presentan algoritmos RRT para resolver dicho problema, en la Sección 3 se da una visión general de las medidas más utilizadas de calidad de una trayectoria y se presentan algoritmos RRT óptimos. En la Sección 4 se presentan algoritmos RRT que aunque no garantizan una solución óptima, aplican heurísticas que favorecen la obtención de trayectorias cercanas a la óptima. Finalmente, la Sección 5 presenta las conclusiones.

2 El algoritmo RRT

Para definir el problema de la planificación de movimientos se deben introducir algunos conceptos. Para ello considérese un sistema robótico, cuyos movimientos se quieren planificar, con d grados de libertad (DOF). Nótese que por sistema robótico puede entenderse desde un robot móvil que se traslada en el plano hasta un brazo robótico o un sistema multirobot. Entonces la configuración $\boldsymbol{q} \in \mathbb{R}^d$ representa de forma unívoca un estado del sistema robótico y el subespacio de todos los posibles valores de $\boldsymbol{q}$ es el espacio de configuraciones $\mathcal{C} \subseteq \mathbb{R}^d$. Sea $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$ el subespacio de configuraciones libres de colisiones (sin autocolisiones ni colisiones robot-robot o robot-entorno). Sea una trayectoria $\mathcal{P}$ representada como una secuencia de N configuraciones $\boldsymbol{q}_i$ consecutivamente conectadas por $N{-}1$ segmentos rectilíneos, i.e.

$$\mathcal{P} = \bigcup_{i=1}^{N-1} \textsc{Segment}(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}) \tag{1}$$

Finalmente, dadas unas configuración inicial $\boldsymbol{q}_{\text{start}} \in \mathcal{C}_{\text{free}}$ y una configuración final $\boldsymbol{q}_{\text{goal}} \in \mathcal{C}_{\text{free}}$, el problema de la planificación de movimientos consiste en encontrar una trayectoria $\mathcal{P}$ que lleve el sistema robótico de $\boldsymbol{q}_{\text{start}}$ a $\boldsymbol{q}_{\text{goal}}$ (i.e. $\boldsymbol{q}_1 = \boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_N = \boldsymbol{q}_{\text{goal}}$) de forma que todas las configuraciones de la trayectoria y los segmentos que las unen se encuentren en $\mathcal{C}_{\text{free}}$.

La idea básica del planificador RRT [12] consiste en construir un árbol de muestras con origen en la configuración inicial $\boldsymbol{q}_{\text{start}}$ y hacerlo crecer añadiendo iterativamente nuevas configuraciones mientras se explora el espacio de configuraciones $\mathcal{C}$ hasta alcanzar la configuración final $\boldsymbol{q}_{\text{goal}}$. El funcionamiento del RRT se describe en el Algoritmo 1. Primeramente se crea el árbol de muestras $\mathcal{T}$ enraizado en $\boldsymbol{q}_{\text{start}}$ (Línea 1). Iterativamente se hace crecer el árbol mientras no se cumpla cierto criterio de parada (Línea 2), cómo por ejemplo superar un número máximo de iteraciones, un tamaño

Algoritmo 1: RRT

Entrada: Configuración inicial $\boldsymbol{q}_{\text{start}} \in \mathcal{C}_{\text{free}}$
Configuración final $\boldsymbol{q}_{\text{goal}} \in \mathcal{C}_{\text{free}}$
Salida : Trayectoria $\mathcal{P} \in \mathcal{C}_{\text{free}}$ entre $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$

1: $\mathcal{T} \leftarrow \text{INITTREE}(\boldsymbol{q}_{\text{start}})$
2: **mientras no** $\text{STOPCRITERIA}(\mathcal{T})$ **hacer**
3: $\quad \boldsymbol{q}_{\text{rand}} \leftarrow \text{RANDCONF}(\,)$
4: $\quad \boldsymbol{q}_{\text{new}} \leftarrow \text{EXTEND}(\mathcal{T}, \boldsymbol{q}_{\text{rand}})$
5: $\quad$ **si** $\boldsymbol{q}_{\text{new}} = \boldsymbol{q}_{\text{goal}}$ **entonces**
6: $\quad\quad$ **devuelve** $\text{PATH}(\mathcal{T})$
7: **devuelve** $\emptyset$

Algoritmo 2: EXTEND

Entrada: Árbol de muestras $\mathcal{T}$
Configuración $\boldsymbol{q}_{\text{rand}}$
Salida : Configuración $\boldsymbol{q}_{\text{new}}$

1: $\boldsymbol{q}_{\text{near}} \leftarrow \text{NEARESTNEIGHBOR}(\mathcal{T}, \boldsymbol{q}_{\text{rand}})$
2: $\boldsymbol{q}_{\text{new}} \leftarrow \boldsymbol{q}_{\text{near}} + \min\big(\epsilon, \|\boldsymbol{q}_{\text{rand}} - \boldsymbol{q}_{\text{near}}\|\big) \dfrac{\boldsymbol{q}_{\text{rand}} - \boldsymbol{q}_{\text{near}}}{\|\boldsymbol{q}_{\text{rand}} - \boldsymbol{q}_{\text{near}}\|}$
3: **si** $\text{VALIDSEGMENT}(\boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{new}})$ **entonces**
4: $\quad \text{ADDSEGMENT}(\mathcal{T}, \boldsymbol{q}_{\text{near}}, \boldsymbol{q}_{\text{new}})$
5: $\quad$ **devuelve** $\boldsymbol{q}_{\text{new}}$
6: **devuelve** $\emptyset$

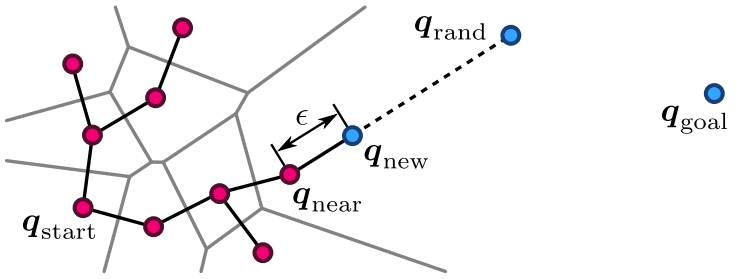


Figura 1: Funcionamiento del algoritmo RRT. Un árbol de muestras con base en la configuración inicial $\boldsymbol{q}_{\text{start}}$ se hace crecer hacia la configuración final $\boldsymbol{q}_{\text{goal}}$. Para ello, desde una configuración $\boldsymbol{q}_{\text{near}}$ se toma un paso ϵ hacia una configuración aleatoria $\boldsymbol{q}_{\text{rand}}$, para obtener una nueva configuración, $\boldsymbol{q}_{\text{new}}$. También se muestran las regiones de Voronoi asociadas a cada configuración del árbol de muestras.

máximo del árbol o un periodo de tiempo especificado. Para ello, en cada iteración se obtiene una configuración $\boldsymbol{q}_{\text{rand}}$ mediante un muestreo aleatorio en el espacio de configuraciones $\mathcal{C}$ (Línea 3). Con una probabilidad $(1-\alpha)$ la configuración $\boldsymbol{q}_{\text{rand}}$ se obtiene de una distribución uniforme en $\mathcal{C}$, pero con una probabilidad α la configuración $\boldsymbol{q}_{\text{rand}}$ es $\boldsymbol{q}_{\text{goal}}$. Este sesgo con probabilidad α hacia $\boldsymbol{q}_{\text{goal}}$ se fija generalmente en torno al 5% y permite conectar rápidamente $\mathcal{T}$ con $\boldsymbol{q}_{\text{goal}}$. Posteriormente, el árbol crece hacia $\boldsymbol{q}_{\text{rand}}$ añadiendo una nueva configuración $\boldsymbol{q}_{\text{new}}$ (Línea 4), calculada mediante la función EXTEND que se explica más adelante. Si la nueva configuración $\boldsymbol{q}_{\text{new}}$ coincide con $\boldsymbol{q}_{\text{goal}}$ (Línea 5), entonces es que el algoritmo ha encontrado solución y éste devuelve la trayectoria que conecta $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$ a través de las configuraciones del árbol $\mathcal{T}$ (Línea 6). En caso contrario, se repite este procedimiento con una nueva configuración $\boldsymbol{q}_{\text{rand}}$. Finalmente, se devuelve $\emptyset$ (Línea 7) si se satisface el criterio de parada sin haber encontrado solución.

La función EXTEND, descrita en el Algoritmo 2 e ilustrada en la Figura 1, sigue el siguiente procedimiento para extender el árbol $\mathcal{T}$ hacia la configuración $\boldsymbol{q}_{\text{rand}}$. Primero, entre las configuraciones de $\mathcal{T}$ se selecciona la configuración $\boldsymbol{q}_{\text{near}}$ más cercana a $\boldsymbol{q}_{\text{rand}}$ (Línea 1). Entonces, se toma un pequeño paso de longitud ϵ desde $\boldsymbol{q}_{\text{near}}$ hacia $\boldsymbol{q}_{\text{rand}}$ alcanzando así una nueva configuración $\boldsymbol{q}_{\text{new}}$ (Línea 2). Si el segmento rectilíneo entre $\boldsymbol{q}_{\text{near}}$ y $\boldsymbol{q}_{\text{new}}$ es válido, i.e. a lo largo del segmento no se producen colisiones del robot consigo mismo o con el entorno, se agrega el segmento al árbol (Línea 4) y se devuelve $\boldsymbol{q}_{\text{new}}$ (Línea 5). Si no es válido, el árbol no crece y se devuelve $\emptyset$ (Línea 6). Como resultado de esta forma de crecer el árbol, la probabilidad de que una muestra sea escogida para extender el árbol es proporcional al área de su región de Voronoi. Esto provoca que el árbol de muestras crezca incremental y rápidamente hacia regiones no exploradas del espacio de configuraciones (véase Figura 1). Esta capacidad del planificador RRT para explorar el espacio de configuraciones es referida a menudo como el sesgo de Voronoi del RRT.

Para aumentar la eficiencia del algoritmo RRT (i.e. reducir el tiempo de planificación) una variante denominada RRTConnect [12] utiliza dos árboles para realizar una búsqueda bidireccional de la solución. Los árboles crecen incremental y alternativamente explorando el espacio de configuraciones hasta que logran conectarse. El planificador RRTConnect se presenta en el Algoritmo 3 y fun-

Algoritmo 3: RRTCONNECT

Entrada: Configuración inicial $\boldsymbol{q}_{\text{start}} \in \mathcal{C}_{\text{free}}$
Configuración final $\boldsymbol{q}_{\text{goal}} \in \mathcal{C}_{\text{free}}$
Salida : Trayectoria $\mathcal{P} \in \mathcal{C}_{\text{free}}$ entre $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$

1: $(\mathcal{T}_{\text{A}}, \mathcal{T}_{\text{B}}) \leftarrow \big(\text{INITTREE}(\boldsymbol{q}_{\text{start}}), \text{INITTREE}(\boldsymbol{q}_{\text{goal}})\big)$
2: **mientras no** STOPCRITERIA($\mathcal{T}$) **hacer**
3: $\quad \boldsymbol{q}_{\text{rand}} \leftarrow \text{RANDCONF}(\,)$
4: $\quad \boldsymbol{q}_{\text{new}} \leftarrow \text{EXTEND}(\mathcal{T}_{\text{A}}, \boldsymbol{q}_{\text{rand}})$
5: $\quad$ **si** CONNECT($\mathcal{T}_{\text{B}}, \boldsymbol{q}_{\text{new}}$) **entonces**
6: $\quad\quad$ **devuelve** PATH($\mathcal{T}_{\text{A}}, \mathcal{T}_{\text{B}}$)
7: $\quad (\mathcal{T}_{\text{A}}, \mathcal{T}_{\text{B}}) \leftarrow (\mathcal{T}_{\text{B}}, \mathcal{T}_{\text{A}})$
8: **devuelve** $\emptyset$

Algoritmo 4: CONNECT

Entrada: Árbol de muestras $\mathcal{T}$
Configuración $\boldsymbol{q}$
Salida : `Cierto` si $\mathcal{T}$ alcanza $\boldsymbol{q}$
`Falso` si $\mathcal{T}$ no puede alcanzar $\boldsymbol{q}$

1: **haz**
2: $\quad \boldsymbol{q}_{\text{new}} \leftarrow \text{EXTEND}(\mathcal{T}, \boldsymbol{q})$
3: $\quad$ **si** $\boldsymbol{q}_{new} = \emptyset$ **entonces**
4: $\quad\quad$ **devuelve** `Falso`
5: **mientras** $\boldsymbol{q}_{\text{new}} \neq \boldsymbol{q}$
6: **devuelve** `Cierto`

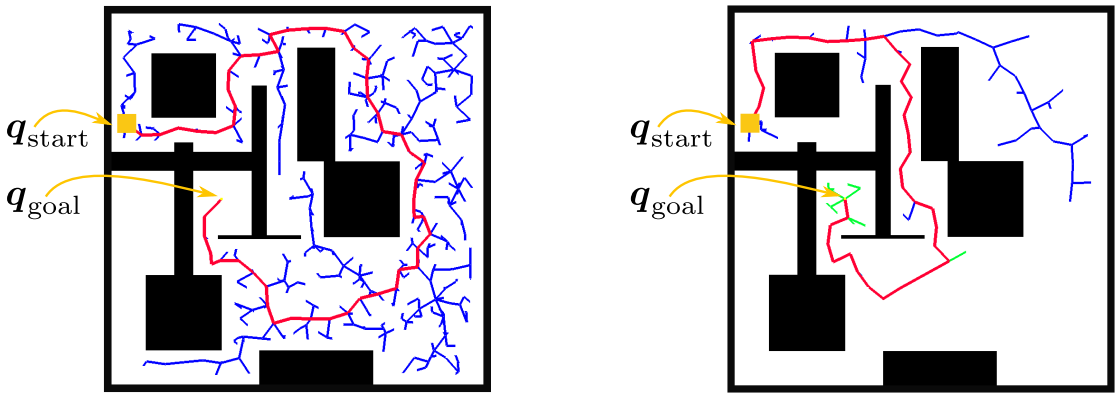


Figura 2: Un robot móvil (amarillo) debe ir desde la posición en la que se encuentra hacia una posición final deseada, evitando los obstáculos (negro). Se muestran las trayectorias (rojo) encontradas con el algoritmo RRT (izquierda) y con el algoritmo RRTConnect (derecha). También se muestran los árboles de muestras (de azul o de verde según si están enraizados en la posición inicial o final, respectivamente).

ciona de la siguiente manera. El algoritmo mantiene dos árboles de muestras, uno enraizado en $\boldsymbol{q}_{\text{start}}$ y el otro en $\boldsymbol{q}_{\text{goal}}$ (Línea 1). El primer árbol, $\mathcal{T}_{\text{A}}$, crece hasta una configuración $\boldsymbol{q}_{\text{new}}$ como un RRT clásico (Líneas 3 y 4). A continuación el segundo árbol, $\mathcal{T}_{\text{B}}$, intenta crecer hasta alcanzar $\boldsymbol{q}_{\text{new}}$ y así conectarse a $\mathcal{T}_{\text{A}}$ (Línea 5), mediante la función CONNECT explicada más adelante. Si se consigue conectar los dos árboles, el algoritmo ya ha encontrado solución y devuelve la trayectoria que conecta $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$ a través de las configuraciones de los árboles de muestras $\mathcal{T}_{\text{A}}$ y $\mathcal{T}_{\text{B}}$ (Línea 6). En caso contrario, el papel de los árboles se intercambia y el proceso se vuelve a realizar (Línea 7). Del mismo modo, si se satisface el criterio de parada y aún no se ha encontrado solución, se devuelve $\emptyset$ (Línea 8).

La función CONNECT, descrita en el Algoritmo 4, extiende un árbol $\mathcal{T}$ hacia una configuración $\boldsymbol{q}$ pero en vez de tomar un único paso incremental ϵ, enlaza consecutivas llamadas de la función EXTEND (Línea 2) hasta que se produce una colisión (Línea 3), y en ese caso la conexión no es posible, o bien se alcanza $\boldsymbol{q}$, y se consigue la conexión de los árboles (Línea 6).

A modo de ejemplo se han planificado los movimientos de un robot móvil que se traslada en un laberinto evitando los obstáculos hasta una configuración deseada. La Figura 2 muestra las trayectorias obtenidas con los algoritmos RRT y RRTConnect, observándose que para el RRT la trayectoria encontrada es más larga y el árbol ha explorado una mayor área de $\mathcal{C}$ que en el caso del RRTConnect. También se han planificado los movimientos de un sistema antropomorfo bibrazo [22], formado por dos brazos robóticos de 6 DOF equipados con manos mecánicas de 16 DOF, que debe insertar una lata dentro de una caja cilíndrica. La Figura 3 muestra la ejecución de una trayectoria encontrada con el RRTConnect. Para estos dos ejemplos (y los presentados en las siguientes secciones) se ha utilizado *The Kautham Project* [17], un entorno de planificación de movimientos y de simulación desarrollado en el Instituto de Organización y Control de Sistemas Industriales (IOC-UPC) para investigación y docencia.

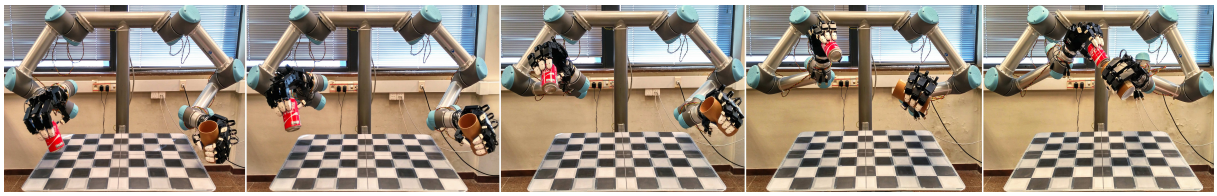


Figura 3: Un robot antropomorfo bibrazo debe insertar una lata dentro de una caja cilíndrica. Se muestran, por orden cronológico, distintas instantáneas de la ejecución real de una de las soluciones encontradas por el algoritmo RRTConnect.

3 Obtención de trayectorias óptimas

3.1 Funciones de coste

En general, los planificadores basados en el algoritmo RRT clásico permiten encontrar eficientemente una trayectoria válida (i.e. conectando las configuraciones inicial y final sin colisiones) pero en ningún caso considerando la calidad de la trayectoria obtenida. Sin embargo, en muchas aplicaciones la calidad de los movimientos a realizar es importante. El concepto de calidad de una trayectoria varía de una aplicación a otra pero siempre puede ligarse a una función de coste que debe ser minimizada. A continuación se detallan varias funciones de coste. Considérese una trayectoria $\mathcal{P} \in \mathcal{C}_{\text{free}}$ conectando dos configuraciones $\boldsymbol{q}_0$ y $\boldsymbol{q}_L$ con un camino de longitud L. Sea entonces $\mathcal{P}(s)$ una parametrización de $\mathcal{P}$ según su longitud (con $s \in [0, L]$) de forma que $\mathcal{P}(0) = \boldsymbol{q}_0$, $\mathcal{P}(L) = L$ y $\|\dot{\mathcal{P}}(s)\| = 1 \ \forall s$. Nótese que con esta nomenclatura $\mathcal{P}$ igualmente puede representar un segmento rectilíneo entre dos configuraciones o una curva más compleja. De este modo utilizando la misma nomenclatura pueden definirse distintos costes $c(\mathcal{P})$ de una trayectoria $\mathcal{P}$:

- Distancia: La trayectoria óptima según este coste es la del camino más corto en $\mathcal{C}$ que conecta dos configuraciones. Es la función de coste más frecuentemente utilizada y se calcula como

$$c(\mathcal{P}) = \int_0^L \mathrm{d}s = L \tag{2}$$

- Holgura mínima: Este coste busca maximizar la holgura mínima de la trayectoria (i.e. la mínima distancia entre cualquier elemento del robot y otro elemento con el que pueda colisionar, ya sea del entorno o del propio robot). Así se reducen las probabilidades de colisión cuando no se conoce con precisión el modelo cinemático del robot o la localización y dimensión de los obstáculos. Siendo $\textsc{Clearance}(\boldsymbol{q})$ el valor de la holgura para una configuración $\boldsymbol{q}$ dada, el coste se calcula como

$$c(\mathcal{P}) = \max_{s \in [0,L]} \frac{1}{\textsc{Clearance}\big(\mathcal{P}(s)\big)} \tag{3}$$

- Coste integral: Dada una función escalar $v : \mathcal{C} \to \mathbb{R}^+$, este coste se define de la siguiente manera

$$c(\mathcal{P}) = \int_0^L v\big(\mathcal{P}(s)\big)\,\mathrm{d}s \tag{4}$$

  La función $v(\boldsymbol{q})$ puede ser por ejemplo el consumo eléctrico, la aceleración del elemento terminal del robot o una combinación ponderada de varias funciones. Si por ejemplo se quiere un equilibrio entre minimizar la longitud y maximizar la holgura media de la trayectoria, basta con escoger por ejemplo la función $v(\boldsymbol{q}) = 1 + \textsc{Clearance}(\boldsymbol{q})^{-1}$.

- Trabajo mecánico: Este coste acumula las variaciones positivas de $v(\boldsymbol{q})$ a lo largo de la trayectoria, i.e. mide el trabajo mecánico [7] definido de la manera siguiente

$$c(\mathcal{P}) = \int_0^L \max\left(0, \frac{\partial v\big(\mathcal{P}(s)\big)}{\partial s}\right) \mathrm{d}s \tag{5}$$

  Si por ejemplo, se planifica la trayectoria de un robot móvil por un terreno irregular y $v(\boldsymbol{q})$ devuelve la elevación del terreno, la trayectoria de mínimo trabajo mecánico es aquella que evita las pendientes ascendentes pronunciadas.

- Flujo a contracorriente: Dado un campo vectorial $\boldsymbol{f}(\boldsymbol{q})$ que indica en cada configuración $\boldsymbol{q} \in \mathcal{C}$ la dirección de movimiento deseada para la trayectoria, este coste indica en qué medida la trayectoria va en contra del campo vectorial $\boldsymbol{f}$, i.e. mide el flujo a contracorriente [11] definido como

$$c(\mathcal{P}) = \int_0^L \left(\left\| \boldsymbol{f}\big(\mathcal{P}(s)\big) \right\| - \dot{\mathcal{P}}(s) \cdot \boldsymbol{f}\big(\mathcal{P}(s)\big) \right) \mathrm{d}s \tag{6}$$

  Este coste es útil, por ejemplo, cuando se tiene un cuadricóptero volando en presencia de fuertes vientos o un robot submarino en una zona con muchas corrientes. En estos casos, el camino más corto no siempre es el mejor ya que estos robots suelen estar alimentados por baterías y moverse en contra del viento o de las corrientes implica un mayor consumo energético y menor autonomía.

3.2 El algoritmo RRT*

Pese a que el planificador RRT obtiene trayectorias válidas para un gran tipo de problemas, éstas son altamente subóptimas. De hecho, aunque el RRT se ejecutara durante un tiempo infinito no se encontraría la solución óptima [9]. Para obtener una trayectoria óptima (respecto a cierta medida de coste), se ha introducido el algoritmo RRT* [9]. Este algoritmo es probabilísticamente óptimo, i.e. la solución converge hacia la trayectoria óptima a medida que el número de muestras tiende a infinito, y su coste computacional se mantiene proporcional al del RRT.

El funcionamiento del RRT*, ilustrado en la Figura 4 y descrito en los Algoritmos 5 y 6, es similar al del RRT clásico: se toma de $\mathcal{C}$ una configuración aleatoria $\boldsymbol{q}_{\text{rand}}$ y se da un paso de longitud ϵ hacia $\boldsymbol{q}_{\text{rand}}$ desde $\boldsymbol{q}_{\text{near}}$, la configuración del árbol de muestras más cercana a $\boldsymbol{q}_{\text{rand}}$. Sin embargo, $\boldsymbol{q}_{\text{new}}$ no se conecta al árbol a través de $\boldsymbol{q}_{\text{near}}$ sino que se toma en cuenta todo un conjunto Q_{near} de configuraciones del árbol vecinas a $\boldsymbol{q}_{\text{new}}$. De este modo, la configuración $\boldsymbol{q}_{\text{parent}}$ a la que se conecta $\boldsymbol{q}_{\text{new}}$ es aquella configuración en Q_{near} que implica una trayectoria de menor coste para alcanzar $\boldsymbol{q}_{\text{new}}$ desde $\boldsymbol{q}_{\text{start}}$ sujeto a que el segmento entre $\boldsymbol{q}_{\text{parent}}$ y $\boldsymbol{q}_{\text{new}}$ sea libre de colisiones (Línea 3 del Algoritmo 6). Posteriormente, la configuración $\boldsymbol{q}_{\text{new}}$ es considerada para reemplazar la configuración predecesora de cada configuración en Q_{near} (Línea 6 del Algoritmo 5). Esto significa que si para acceder desde $\boldsymbol{q}_{\text{start}}$ a una configuración $\boldsymbol{q} \in Q_{\text{near}}$ se obtiene un coste menor pasando por $\boldsymbol{q}_{\text{new}}$ que por el camino actual del árbol, $\boldsymbol{q}_{\text{new}}$ pasa a ser la nueva configuración predecesora de esa configuración $\boldsymbol{q}$ (véase Figura 4). Nótese además que el algoritmo RRT* no termina al encontrar una solución sino que continua obteniendo una trayectoria cada vez mejor hasta que se cumpla alguno de los criterios de parada.

El algoritmo RRT* se ha utilizado para planificar los movimientos de un cuadricóptero mientras realiza una inspección aérea de las chimeneas de una central eléctrica [5]. Se quiere que el cuadricóptero no se acerque demasiado a las chimeneas y que además éste siga preferiblemente un conjunto de líneas horizontales (véase Figura 5-a). Teniendo en cuenta estas restricciones se decide utilizar la función de coste presentada en Ec. (4) y se construye la función escalar v de forma que otorgue valores elevados cerca de las chimeneas y valores bajos en las líneas horizontales y en la configuración final. La Figura 5-c muestra los valores de v para varias alturas, desde el nivel del suelo en que encuentra inicialmente el cuadricóptero hasta la altura de la configuración final. El cuadricóptero ha sido modelizado como una esfera para el chequeo de colisiones y su dinámica no se ha tenido en cuenta. Si fuera necesario, la dinámica del cuadricóptero se puede considerar seleccionando puntos de paso de la trayectoria óptima y optimizando conjuntamente un sistema de polinomios a través de ellos para suavizar la trayectoria [16]. Las Figuras 5-a y 5-b muestran distintas vistas de la trayectoria final obtenida.

Con el fin de acelerar la tasa de mejora de la trayectoria y compensar la fuerte tendencia del RRT* a explorar el espacio de configuraciones, se han propuesto diversas heurísticas para mejorar el muestreo de nuevas configuraciones como, por ejemplo, un sesgo hacia todas o algunas configuraciones de la trayectoria actual [1, 6] o un sesgo basado en proyecciones aprendidas durante la planificación [19]. También se han propuesto otras mejoras como rechazar todas las nuevas configuraciones que no tengan una solución heurística estimada de coste menor al de la solución actual [1]. Y es que si se realiza un muestreo global del espacio de configuraciones, el algoritmo RRT* acaba encontrando las trayectorias óptimas que conectan $\boldsymbol{q}_{\text{start}}$ con cada configuración en $\mathcal{C}$. Esto es inconsistente con el problema que se

Algoritmo 5: RRT*

Entrada: Configuración inicial $\boldsymbol{q}_{\text{start}} \in \mathcal{C}_{\text{free}}$
Configuración final $\boldsymbol{q}_{\text{goal}} \in \mathcal{C}_{\text{free}}$
Salida : Trayectoria $\mathcal{P} \in \mathcal{C}_{\text{free}}$ entre $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$

$\mathcal{T} \leftarrow$ INITTREE($\boldsymbol{q}_{\text{start}}$)
mientras no STOPCRITERIA($\mathcal{T}$) **hacer**
  $\boldsymbol{q}_{\text{rand}} \leftarrow$ RANDCONF( )
  $\boldsymbol{q}_{\text{new}} \leftarrow$ BESTEXTEND($\mathcal{T}, \boldsymbol{q}_{\text{rand}}$)
  si $\boldsymbol{q}_{\text{new}} \neq \emptyset$ **entonces**
    REWIRE($\mathcal{T}, \boldsymbol{q}_{\text{new}}$)
devuelve PATH($\mathcal{T}$)

Algoritmo 6: BESTEXTEND

Entrada: Árbol de muestras $\mathcal{T}$
Configuración $\boldsymbol{q}$
Salida : Configuración $\boldsymbol{q}_{\text{new}}$

$\boldsymbol{q}_{\text{near}} \leftarrow$ NEARESTNEIGHBOR($\mathcal{T}, \boldsymbol{q}$)
$\boldsymbol{q}_{\text{new}} \leftarrow \boldsymbol{q}_{\text{near}} + \min(\epsilon, \|\boldsymbol{q} - \boldsymbol{q}_{\text{near}}\|) \dfrac{\boldsymbol{q} - \boldsymbol{q}_{\text{near}}}{\|\boldsymbol{q} - \boldsymbol{q}_{\text{near}}\|}$
$\boldsymbol{q}_{\text{parent}} \leftarrow$ BESTPARENT($\mathcal{T}, \boldsymbol{q}_{\text{new}}$)
si $\boldsymbol{q}_{\text{parent}} \neq \emptyset$ **entonces**
  ADDSEGMENT($\mathcal{T}, \boldsymbol{q}_{\text{parent}}, \boldsymbol{q}_{\text{new}}$)
  devuelve $\boldsymbol{q}_{\text{new}}$
devuelve $\emptyset$

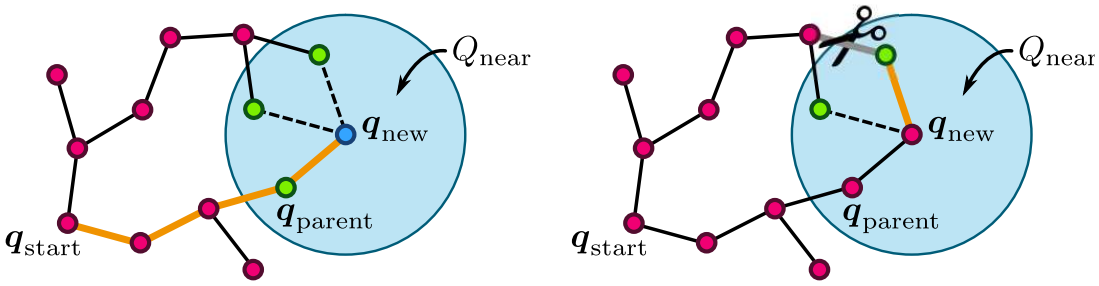


Figura 4: Funcionamiento del algoritmo RRT*. La configuración $\boldsymbol{q}_{\text{parent}}$, a la que se conecta la nueva configuración $\boldsymbol{q}_{\text{new}}$, se encuentra en el set Q_{near} de configuraciones vecinas de $\boldsymbol{q}_{\text{new}}$ y es la que ofrece la ruta de menor coste (izquierda). Las configuraciones restantes en Q_{near} son reconectadas si el camino hasta $\boldsymbol{q}_{\text{start}}$ pasando por $\boldsymbol{q}_{\text{new}}$ es de menor coste que el camino a través de la ruta actual (derecha).

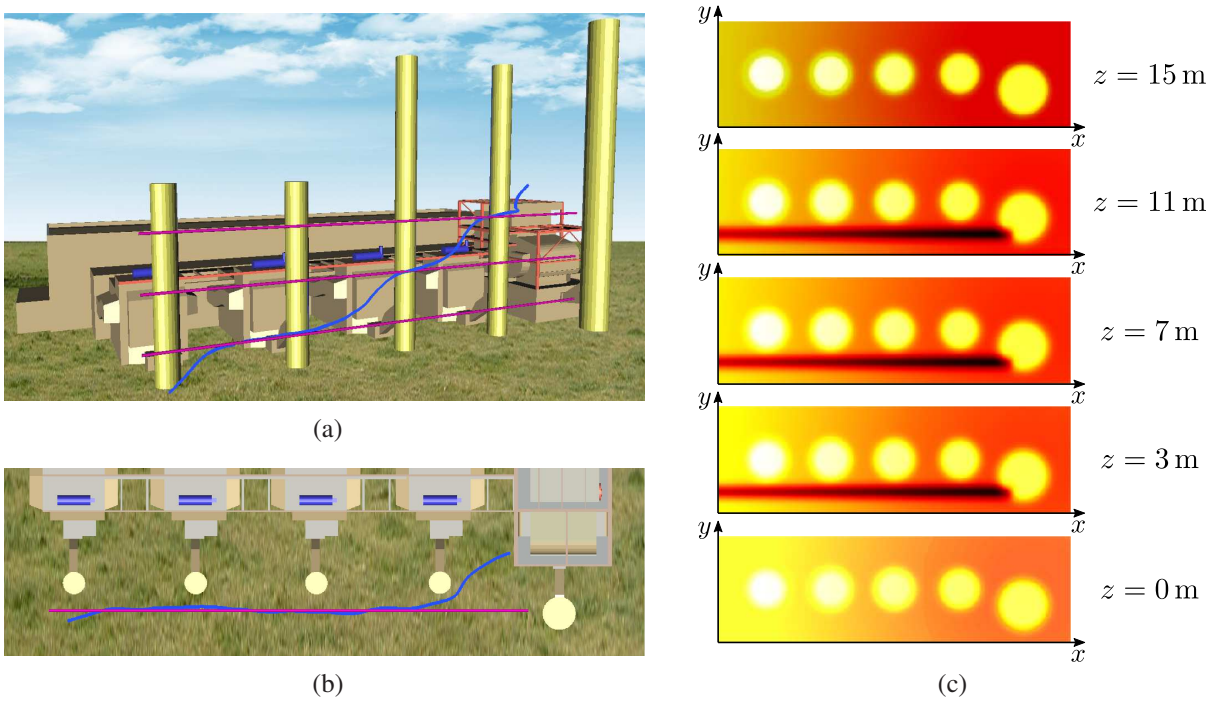


Figura 5: Un cuadricóptero parte de una configuración inicial en el nivel del suelo y debe hacer una inspección aérea de las chimeneas de una central eléctrica. Las líneas horizontales (magenta) que dirigen la trayectoria solución (azul) se muestran en una vista tridimensional (a) y en una vista en planta (b). La función $v(x, y, z)$ se muestra representada a diferentes alturas (c), utilizando el negro y el rojo para valores de v bajos y el amarillo y el blanco para valores de v altos.

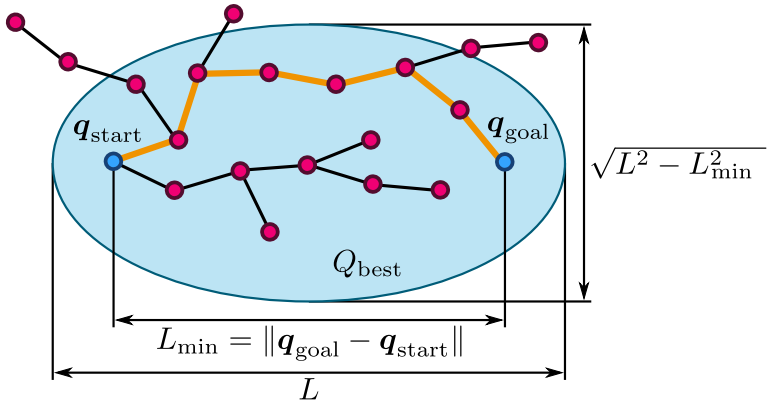


Figura 6: Dada una trayectoria (de longitud L, libre de colisiones y que conecta las configuraciones $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$), el conjunto de configuraciones Q_{best} candidatas a mejorar la trayectoria (i.e. reducir su longitud) está contenido en una elipse con focos en $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$, si el espacio de configuraciones es bidimensional.

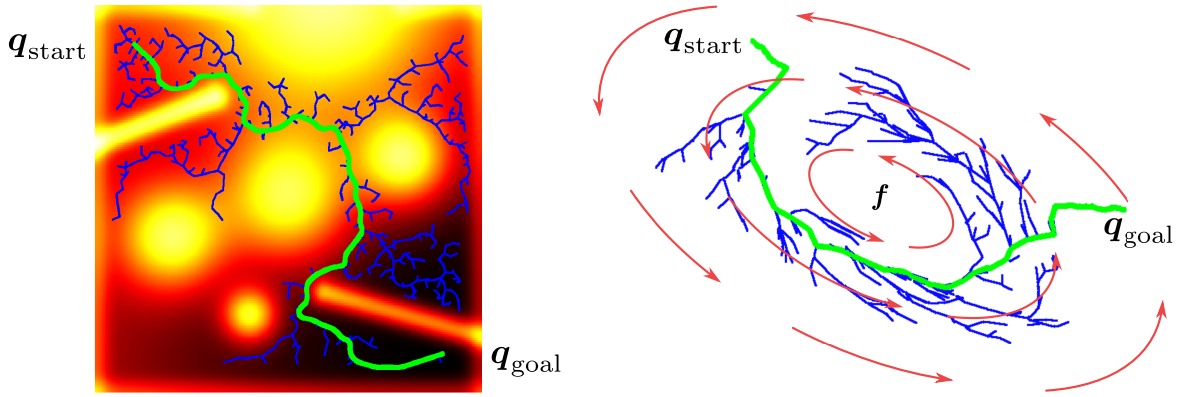
quiere resolver (i.e. encontrar sólo la trayectoria óptima que conecta $\boldsymbol{q}_{\text{start}}$ con $\boldsymbol{q}_{\text{goal}}$). El algoritmo *Informed RRT** [3] ataca este problema para el caso en que se busca la trayectoria de mínima longitud. Este algoritmo funciona exactamente igual que el RRT* hasta encontrar un primera solución. Esta primera solución es utilizada para limitar la búsqueda de futuras soluciones al subespacio Q_{best} de $\mathcal{C}$ que contiene todas las posibles soluciones mejores a la actual. Este subespacio es un hipersferoide prolato (i.e. la generalización de una elipse a un espacio d-dimensional) y puede ser muestreado directamente (véase Figura 6). Posteriores mejoras incrementales de la trayectoria hacen que el hiperesferoide vaya reduciéndose y, si no existen obstáculos, acabe degenerando en el segmento rectilíneo que une $\boldsymbol{q}_{\text{start}}$ y $\boldsymbol{q}_{\text{goal}}$.

4 Planificación de movimientos sin considerar optimalidad

El algoritmo RRT* permite obtener trayectorias óptimas de acuerdo a un criterio de calidad. No obstante, tiene una tasa de convergencia muy lenta especialmente al planificar movimientos para sistemas robóticos de muchos grados de libertad. Por esta razón, se han desarrollado algoritmos específicos para un criterio de calidad determinado que, si bien no garantizan una solución óptima, aplican heuristícas que favorecen la obtención de trayectorias cercanas a la óptima. Los algoritmos *Transition-based RRT* (T-RRT) [7] y *Vector-Field RRT* (VF-RRT) [11], ambos variantes del algoritmo RRT, son buenos ejemplos de este tipo de planificadores. Su funcionamiento es el siguiente:

- El algoritmo T-RRT aplica un filtro adaptativo para extender el árbol de muestras siguiendo los valles y puntos de silla de la función escalar $v(\boldsymbol{q})$ y así encontrar trayectorias de trabajo mecánico reducido, véase Ec. (5). Su funcionamiento es similar al del RRT clásico pero antes de añadir un nuevo segmento al árbol además de comprobarse que no tenga colisiones también se comprueba que el trabajo mecánico del segmento sea menor que un cierto umbral variable λ. El parámetro λ es ajustado dinámicamente de acuerdo a la información adquirida durante la exploración.
- El algoritmo VF-RRT favorece el crecimiento del árbol en la dirección del campo vectorial $\boldsymbol{f}(\boldsymbol{q})$ para obtener trayectorias con un flujo a contracorriente pequeño, véase Ec. (6). Su funcionamiento es similar al del RRT clásico pero el VF-RRT modifica la dirección de crecimiento del árbol para que crezca en una dirección intermedia entre la que indica la configuración aleatoria $\boldsymbol{q}_{\text{rand}}$ y la del campo vectorial $\boldsymbol{f}$. Cuál de estas direcciones tiene más influencia sobre el crecimiento del árbol depende de un parámetro variable que dinámicamente se va ajustando durante la planificación.

Se han utilizado ambos algoritmos para planificar los movimientos de un robot móvil desplazándose por un plano sin obstáculos y la Figura 7 muestras las trayectorias obtenidas. Para el algoritmo T-RRT se ha utilizado una función escalar v con varias montañas y valles (véase Figura 7-a) y para planificar con el algoritmo VF-RRT se ha fijado un campo vectorial $\boldsymbol{f}$ siguiendo un vórtice que gira en sentido contrario al de las agujas del reloj (véase Figura 7-b).



(a) (b)

Figura 7: Trayectorias (verde) y árboles de muestras (azul) obtenidos al planificar los movimientos de un robot móvil que se desplaza por un plano sin obstáculos para dos casos diferentes: a) planificando con el algoritmo T-RRT en base a una función escalar v (colores más oscuros indican valores inferiores), y b) planificando con el algoritmo VF-RRT en base a un campo vectorial $\boldsymbol{f}$ (rojo).

5 Conclusiones

El algoritmo RRT es uno de los enfoques más relevantes en la planificación de movimientos ya que permite calcular trayectorias para una gran variedad de sistemas robóticos y además de una forma eficiente aún tratando con problemas complejos. Además, las variantes óptimas de este algoritmo permiten mediante heurísticas encontrar trayectorias de gran calidad, es decir que no sólo sean realizables por el robot sino que también minimicen una función de coste dada. La versatilidad existente a la hora de definir esta función de coste permite adaptar el enfoque de estos algoritmos a problemas muy diferentes y a distintas definiciones de calidad de una trayectoria. En este trabajo se ha hecho un repaso de los algoritmos RRT más importantes, aplicándolos en varios ejemplos para mostrar su funcionamiento.

Referencias

[1] Akgun, B. y M. Stilman: *Sampling heuristics for optimal motion planning in high dimensions*. En *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, págs. 2640–2645, Septiembre 2011.

[2] Arslan, O. y P. Tsiotras: *Use of relaxation methods in sampling-based algorithms for optimal motion planning*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 2421–2428, Mayo 2013.

[3] Gammell, J.D., S. Srinivasa, y T.D. Barfoot: *Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic*. En *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, págs. 2997–3004, Septiembre 2014.

[4] García, N., J. Rosell, y R. Suárez: *Motion planning using first-order synergies*. En *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, págs. 2058–2063, Septiembre 2015.

[5] García, N., R. Suárez, y J. Rosell: *HG-RRT*: Human-Guided optimal random trees for motion planning*. En *Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation*, Septiembre 2015.

[6] Islam, F., J. Nasir, U. Malik, Y. Ayaz, y O. Hasan: *RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution*. En *Proc. IEEE Int. Conf. Mechatronics and Automation*, págs. 1651–1656, Agosto 2012.

[7] Jaillet, L., J. Cortés, y T. Siméon: *Sampling-based path planning on configuration-space costmaps*. IEEE Trans. Robotics, 26(4):635–646, Agosto 2010.

[8] Kalakrishnan, M., S. Chitta, E. Theodorou, P. Pastor, y S. Schaal: *STOMP: Stochastic trajectory optimization for motion planning*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 4569–4574, Mayo 2011.

[9] Karaman, S. y E. Frazzoli: *Sampling-based algorithms for optimal motion planning*. Int. J. Robotics Research, 30(7):846–894, Junio 2011.

[10] Kavraki, L.E., P. Svestka, J. C. Latombe, y M.H. Overmars: *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. IEEE Trans. Robotics and Automation, 12(4):566–580, Agosto 1996.

[11] Ko, I., B. Kim, y F.C. Park: *Randomized path planning on vector fields*. Int. J. Robotics Research, 33(13):1664–1682, Octubre 2014.

[12] Kuffner, J.J. y S. LaValle: *RRT-Connect: An efficient approach to single-query path planning*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 995–1001, Abril 2000.

[13] Latombe, J. C.: *Robot Motion Planning*. Kluwer Academic Publishers, USA, 1991.

[14] Pan, J., Z. Chen, y P. Abbeel: *Predicting initialization effectiveness for trajectory optimization*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 5183–5190, Mayo 2014.

[15] Ratliff, N., M. Zucker, J.A. Bagnell, y S. Srinivasa: *CHOMP: Gradient optimization techniques for efficient motion planning*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 489–494, Mayo 2009.

[16] Richter, C., A. Bry, y N. Roy: *Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments*. Proc. Int. Symp. Robotics Research, Diciembre 2013.

[17] Rosell, J., A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, y N. García: *The Kautham project: A teaching and research tool for robot motion planning*. En *Proc. IEEE Int. Conf. Emerging Technologies and Factory Automation*, Septiembre 2014.

[18] Rosell, J. y R. Suárez: *Using hand synergies as an optimality criterion for planning human-like motions for mechanical hands*. En *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, págs. 232–237, Noviembre 2014.

[19] Rowekamper, J., G.D. Tipaldi, y W. Burgard: *Learning to guide random tree planners in high dimensional spaces*. En *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, págs. 1752–1757, Noviembre 2013.

[20] Schulman, J., Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, y P. Abbeel: *Motion planning with sequential convex optimization and convex collision checking*. Int. J. Robotics Research, 33(9):1251–1270, Agosto 2014.

[21] Stilman, M.: *Global manipulation planning in robot joint space with task constraints*. IEEE Trans. Robotics, 26(3):576–584, Junio 2010.

[22] Suárez, R., J. Rosell, y N. García: *Using synergies in dual-arm manipulation tasks*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 5655–5661, Mayo 2015.

[23] Yershova, A., L. Jaille, T. Siméon, y S. LaValle: *Dynamic-Domain-RRTs: Efficient exploration by controlling the sampling domain*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 3856–3861, Abril 2005.

[24] Zhang, L. y D. Manocha: *An efficient retraction-based RRT planner*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 3743–3750, Mayo 2008.

[25] Zucker, M. Kuffner, J. y J. Bagnell: *Adaptive workspace biasing for sampling-based planners*. En *Proc. IEEE Int. Conf. Robotics and Automation*, págs. 3757–3762, Mayo 2008.