Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Estrategias probabilistas para la exploración autónoma de ambientes 3D

TESIS PROFESIONAL

Que para obtener el grado de:

Maestro en Ciencias de la Computación

Presenta:

Pablo Camarillo Ramírez

Asesor:

Dr. Abraham Sánchez López

Puebla, Puebla.

Junio, 2014

Agradecimientos

En primer lugar debo agradecer a mi familia porque siempre han sido los que me han dado fuerzas y ánimos para seguir adelante y conseguir mis objetivos. A mis padres que son mi ejemplo a seguir en la vida y a mis hermanos que desde pequeño me cuidaron, me apoyaron y me enseñaron mucho de lo que ahora sé. A ellos les debo todo.

A mis profesores que me ayudaron a explotar mis habilidades y me otorgaron la visión y los conocimientos necesarios para desarrollar este trabajo. Al Dr. Abraham Sánchez López por su guía y su apoyo para la elaboración de esta tesis. A ellos toda mi admiración y respeto.

A mis amigos y compañeros que me acompañaron en esta aventura universitaria y que juntos hemos aprendido a vivir. Sería poco práctico mencionarlos a todos en estas líneas, pero de corazón les agradezco su compañía y su apoyo.

Al Consejo Nacional de Ciencia y Tecnología por el apoyo económico que me brindó durante estos dos años de estudios de la maestría.

Introducción

La planificación de movimientos basada en sensores para robots es una disciplina que dota a los sistemas mecánicos de la capacidad de poder sensar y moverse en un ambiente poblado de obstáculos [17]. En [15], [14], [26], [1], [2], [3] se describen diversas estrategias que resuelven el problema de la planificación de movimientos, pero todos ellos parten del conocimiento total o parcial del ambiente en el cual se encuentra el robot. La exploración de ambientes desconocidos es un problema que consiste en construir un modelo apropiado del ambiente en el que un robot pueda moverse. Existen diversas técnicas para resolver el problema de la exploración de ambientes 2D, tales como la adaptación de técnicas clásicas de planificación de movimientos o los métodos de planificación de movimientos usando sensores [17], sin embargo el proyecto de tesis que se describe en este documento propone extender estas técnicas para realizar la exploración de ambientes desconocidos en el espacio tridimensional.

La exploración de ambientes desconocidos es uno de los principales problemas a resolver en el campo de la robótica móvil y de servicios, ya que involucra tareas básicas como la percepción, la planificación, la localización y la navegación. El objetivo general de nuestro proyecto de investigación es proponer un sistema robótico que realice la exploración de ambientes tridimensionales desconocidos empleando diversas estrategias probabilistas de exploración robótica.

Índice general

Capítulo 1

Estado del Arte

1.1. Exploración robótica

El problema de la exploración robótica ha sido abordado desde diversos enfoques, aunque resaltan dos enfoques fuertemente estudiados: el enfoque determinista y el enfoque probabilista. En este capítulo se presentarán una serie de trabajos relacionados con el problema de la exploración autónoma de ambientes desconocidos y serán presentados en dos partes, por un lado se presentan los trabajos referentes a la estrategias de exploración (las estrategias probabilistas y las estrategias deterministas), y por otro lado se describen algunos trabajos referentes al problema de la reconstrucción de ambientes virtuales.

1.1.1. Estrategias deterministas

Plan-N Scan

El método presentado por P. Renton en 1999 [24] llamado Plan-N-Scan, describe la forma en la que un manipulador genera un *voxel map* a partir de la detección de obstáculos. El sistema incrementalmente sensa (escanea) una secuencia de objetivos seleccionados de acuerdo a sensados previos. El resultado es una malla tridimensional, conveniente para el chequeo de colisiones. Este *voxel map* o malla tridimensional, es de hecho, el mapa del ambiente 3D tal como se ilustra en la Figura 1.1.

El sistema del Plan-N-Scan está basado en la selección automática y el escaneo de objetivos libre de colisión (espacio de puntos 3D) requerido para determinar el volumen de ocupación de la escena. Este sistema hace posible el escaneo exitoso de objetivos incluso en casos dificiles donde el objetivo no puede ser visto inmediatamente. Esto es

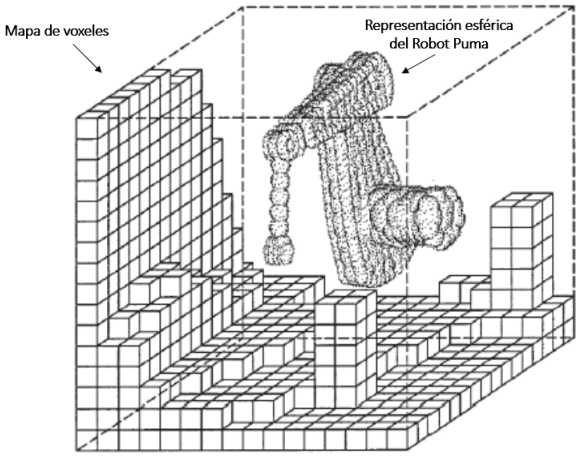


Figura 1.1: Mecanismo de sensado del método Plan-N Scan.

debido al aprovechamiento de los escaneos adquiridos incrementalmente que permiten que el sistema progrese iterativamente hacia el objetivo.

Exploración robótica a partir de multi-agentes robóticos

En la tesis de maestría de K. Hernández [29], publicada en 2010, la autora presenta una solución al problema de la exploración robótica por medio de multi-agentes robóticos y está enfocado específicamente en proponer estrategias de coordinación entre los robots, esto con el objetivo de descubrir si una estrategia de coordinación ayuda a evitar y solucionar conflictos relacionados con colisiones de tal manera que su beneficio se vea reflejado en la reducción del tiempo que se tardan en explorar un ambiente desconocido. Las estrategias de coordinación que se proponen son: **arbitraje** y **replanificación** sin mensajes, con mensajes, **pizarra** y **política cooperativa** descentralizada para la resolución de conflictos.

El diseño de las estrategias de exploración cooperativa proviene del método SRT al cual se le agregan las siguientes funcionalidades, inspiradas por la comunidad multi-agente.

- Cooperación, para incrementar la eficiencia.
- Coordinación, para evitar los conflictos.

- Comunicación como herramienta fundamental para cooperar y coordinar.

En el método de exploración cooperativa basado en SRT, cada robot construye uno o más mapas parciales del ambiente, organizados en una colección de árboles aleatorios de exploración basados en sensores SRT. En la Figura 1.2 se presenta el diagrama de flujo del método multi-SRT con comunicación. Donde GEA es el grupo de agentes enganchados, G_u es el subconjunto de GEA que contiene los robots con caminos no factibles y G_f son los robots restantes de GEA con caminos factibles. Los resultados reportados de esta estrategia de exploración indican una considerable disminución en los tiempos de exploración.

1.1.2. Estrategias probabilistas

El enfoque del problema de la exploración robótica emana de las técnicas de planificación de movimientos aleatorios (PMA), también llamadas estrategias probabilistas, las cuales construyen roadmaps en el espacio de configuraciones libre usando muestreo aleatorio y verificando las colisiones. En PMA, el problema planteado es de aprendizaje activo de orden-libre: lo que se observa del ambiente depende sólo de la última acción (búsqueda aleatoria), ya que en el mapa del ambiente esta disponible progresivamente y el robot se traslada para recopilar más información. Por lo cual, los planificadores aleatorios son considerados como estrategias de exploración orientadas a objetivos utilizando la selección aleatoria de acciones. Como ya se había mencionado, al completitud (probabilística) de estos planificadores es inherente a su naturaleza, además, se puede lograr una mayor eficiencia agregando algunas heurísticas al esquema aleatorio.

Árboles aleatorios de expansión rápida (RRT)

La base de los métodos presentados en esta sección es la construcción incremental de árboles de búsqueda que intentan explorar rápida y uniformemente el espacio de estados, ofreciendo beneficios similares a los obtenidos por otros métodos exitosos de planificación aleatoria, como los métodos de roadmap probabilísticos (PRM) [14], [22].

La propuesta del árbol aleatorio de exploración rápida (RRT, del inglés, Rapidly-Exploring Random Tree) fue presentada por Steve M. LaValle a finales del siglo pasa-

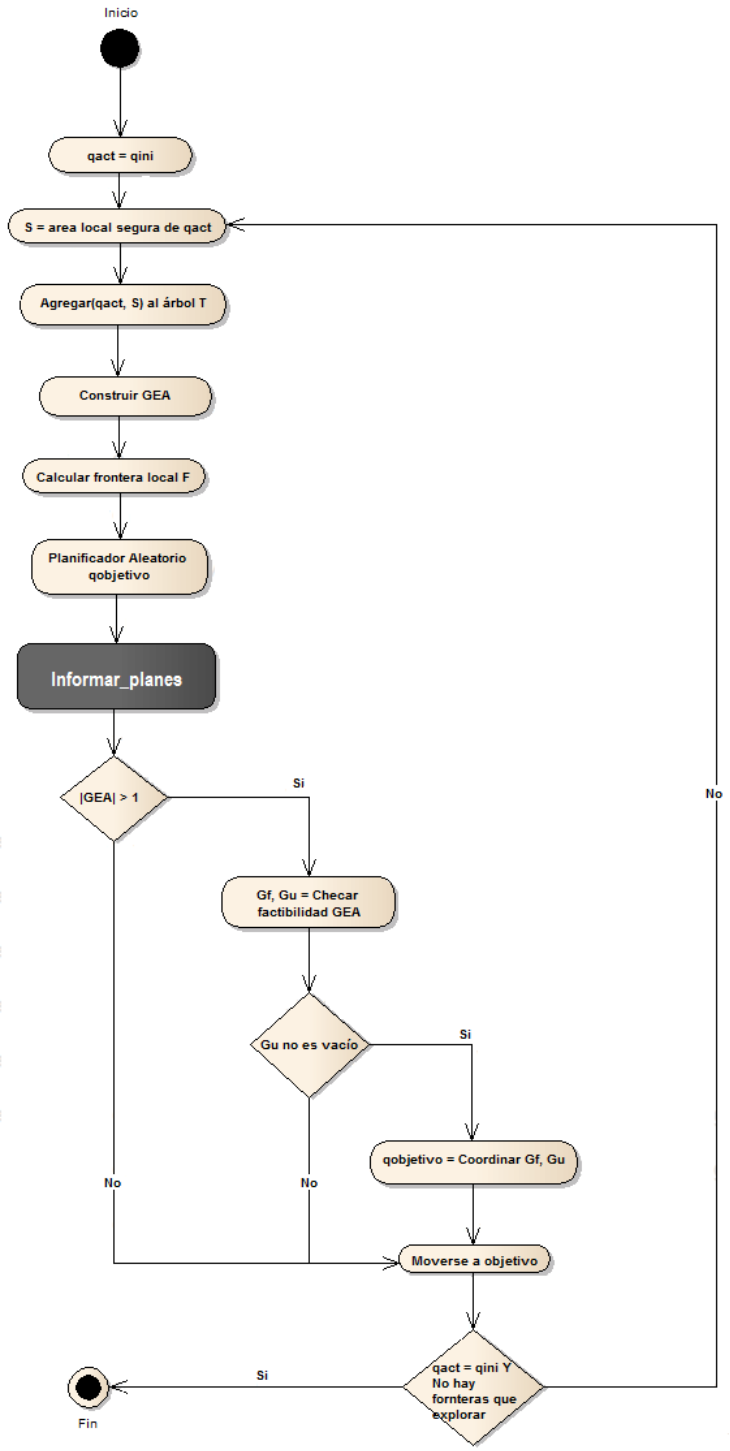


Figura 1.2: Diagrama de flujo del método multi-SRT.

do [16]. Este trabajo fue presentado como un algoritmo de planificación para búsqueda rápida en espacios de altas dimensiones que tienen tanto restricciones algebraicas (provenientes de los obstáculos) como restricciones diferenciales (originadas por la no-holonomía y la dinámica del robot). La idea clave es dirigir la exploración hacia regiones no exploradas del espacio tomando puntos en el espacio de estados e incrementalmente "jalar" la búsqueda.

Algoritmo 1: CONSTRUIR_RRT

Data: x_{ini}
Result: $\mathcal{T}$

```
1 T.ini(x_ini);
2 for k = 1 a K do
3 |   x_aleat ← ESTADO_ALEATORIO();
4 |   EXTENDER(T, x_aleat);
5 Regresar T
```

Algoritmo 2: EXTENDER

Data: $\mathcal{T}, x$
Result: *Estado*

```
1 x_prox ←VECINO_MAS_PROXIMO(x, T);
2 if NUEVO_ESTADO(x, x_prox, x_nuevo, u_nuevo) then
3 |   T.AgregarVertice(x_nuevo);
4 |   T.AgregarArista(x_prox, x_nuevo, u_nuevo);
5 |   if x_nuevo = x then
6 |   |   Estado = Alcanzado;
7 |   else
8 |   |   Estado = Avanzado;
9 Estado = Atrapado;
```

Figura 1.3: Algoritmo básico de construcción del RRT.

El algoritmo básico de construcción de RRT, se presenta en el Algoritmo 1.3. En cada iteración se intenta extender el árbol agregando un nuevo vértice en dirección a un estado seleccionado aleatoriamente. La función EXTENDER, ilustrada en la Figura 1.4, selecciona del árbol, el vértice más cercano a un estado dado. Este vértice se elige de acuerdo a una métrica, ρ. La función NUEVO_ESTADO hace un movimiento

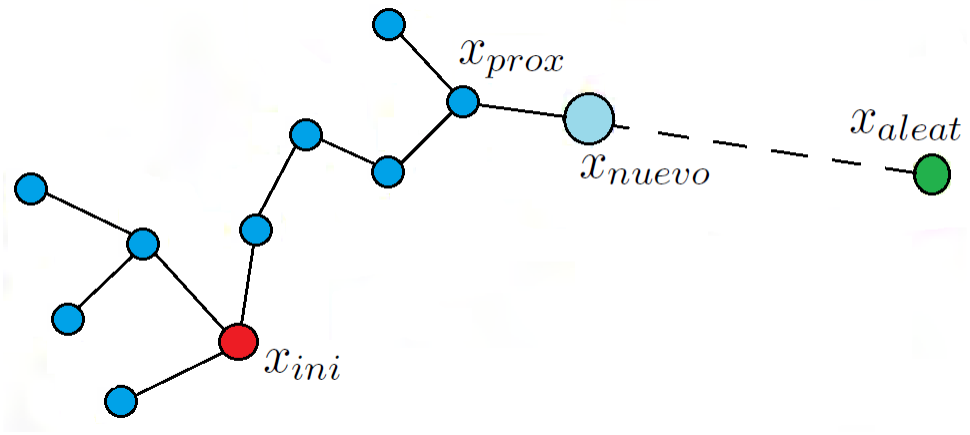


Figura 1.4: Operación de la función EXTENDER.

hacia x aplicando una entrada $u \in U$ para algún incremento Δt. Esta entrada puede seleccionarse aleatoriamente o probando todas las posibles entradas eligiendo aquella que produzca un nuevo estado tan próximo como sea posible a x (si U es infinito puede usarse una aproximación o una técnica analítica). NUEVO_ESTADO utiliza implícitamente una función de detección de colisiones para determinar si el nuevo estado (y todos los estados intermedios) satisfacen las restricciones globales. Si NUEVO_ESTADO se cumple, el nuevo estado junto con la entrada se representan por medio de x_{nuevo} y u_{nuevo}, respectivamente. Pueden ocurrir tres situaciones:

- **Alcanzado**. El nuevo vértice alcanza al estado muestreado x, (para la planificación no-holonómica, tendríamos un umbral, $\|x_{nuevo} - x\| < \epsilon$ para un pequeño ϵ, con $\epsilon > 0$).
- **Avanzado**. Un nuevo vértice $x_{nuevo} \neq x$ es agregado al RRT.
- **Atrapado**. NUEVO_ESTADO falla en producir un nuevo estado que se encuentre en X_{libre}.

Árboles aleatorios de exploración basados en sensores (SRT)

Oriolo, Vendittelli, Freda y Troso presentan en [21] un método de exploración de ambientes desconocidos de una estructura de datos incremental aleatoria llamada *árbol*

aleatorio de exploración basado en sensores (SRT, por sus siglas en inglés Sensor-Based Random Tree), la cual representa un roadmap del área explorada asociada a una región segura. Este método esta inspirado en los árboles aleatorios de exploración rápida (RRTs), la cual es una propuesta presentada por Steve LaValle en [16]. En el trabajo original de los autores [21] se exponen y comparan dos técnicas; la primera, SRT-Ball, los autores la denominan una técnica conservadora y conveniente para usar sensores con ruido. La segunda técnica de percepción llamada SRT-Star es menos conservadora, es decir, confía más en la información reportada por los sensores.

Cada nodo del SRT consiste de una configuración libre y su región segura local (RSL) asociada; la RS es simplemente la unión de todas las RSLs petenecientes al árbol. La RSL es una estimación del espacio libre circunvecino a una configuración dada del robot; en general, su forma dependerá de las características del sensor pero también puede reflejar diferentes posturas de percepción. En el capítulo 2 se profundiza el estudio de está técnica y se explica la adaptación para ambientes tridimensionales.

Árboles de exploración basado en sensores (SET)

El método SET (Árbol de Exploración basada en Sensores, por sus siglas en inglés: *Sensor-based Exploration Tree*), es una estrategia de exploración basada en fronteras para sistemas robóticos generales descrita en [9], que puede ser vista como una extensión del método para robots móviles propuesto en [8], [21]. La idea básica es guiar al robot con el sistema de sensado para realizar una exploración progresiva del mundo, sensando regiones que son contiguas a partir del campo de visión del sensor. La información obtenida sobre el espacio libre es mapeada al espacio de configuraciones del roadmap el cual es construido a partir de un procedimiento de muestreo. El roadmap se usa para seleccionar una nueva configuración de vista, la cual es agregada al SET. En el proceso de exploración, el robot alterna entre movimientos hacia adelante y movimientos hacia atrás sobre el SET, el cual esencialmente funciona como un hilo de Ariadne [1]. En el capítulo 3 se detalla la implementación de este método para nuestro sistema robótico.

1.2. Reconstrucción de mapas 3D de ambientes desconocidos

Como se ha mencionado, una de las tareas propias de la exploración robótica autónoma es la construcción de un modelo virtual que represente el ambiente explorado, para que posteriormente pueda ser usado en tareas de planificación de movimientos o navegación. En esta sección se presentan las técnicas de representación más usadas para la construcción de modelos tridimensionales y también se abordan algunos trabajos que presentan estrategias de reconstrucción de modelos tridimensionales.

1.2.1. Representación de ambientes 3D

A continuación se brindará el marco teórico necesario para entender los mecanismos de construcción de modelos tridimensionales.

Trabajar con ambientes tridimensionales implica trabajar con un mayor número de grados de libertad para el robot así como el uso de estructuras de datos geométricas, distintas a las usadas en un ambiente 2D. Tales estructuras pueden ser los voxeles o los octrees.

Voxel. El **voxel** (del inglés *volumetric pixel* es la unidad cúbica que compone un objeto tridimensional y es, por tanto, el equivalente del pixel de un objeto 2D en un objeto 3D [27].

Para crear una imagen en tres dimensiones, los voxeles tienen que sufrir una transformación de opacidad. Esta información da diferentes valores de opacidad a cada voxel. Esto es importante cuando se han de mostrar detalles de una imagen que quedaría tapada por la capa exterior más opaca de los voxeles.

Las imágenes con voxeles se usan generalmente en el campo de la medicina y se aplican, por ejemplo, en la tomografía axial computarizada o para las resonancias magnéticas. De este modo, los profesionales pueden obtener un modelo preciso en tres dimensiones del cuerpo humano. Actualmente, su uso ya se ha extendido en una multitud de campos como la medicina, ingeniería, cine, videojuegos, etc.

Al igual que los pixeles, los voxeles no contienen su posición (x,y,z) en el espacio 3D, si no que se deduce por la posición del voxel dentro del archivo de datos.

Octree. Un **octree** es una estructura de datos tipo *árbol* (en informática), en la cual cada nodo interno tiene exactamente 8 *hijos*. Las estructuras octree se usan mayormente para particionar un espacio tridimensional, dividiéndolo recursivamente en ocho octantes. Las estructuras octree son análogas tridimensionales de los quadtree bidimensionales. El nombre está formado a partir de la raíz ***oct***(octante) + ***tree***(árbol), y normalmente se escribe como *octree* en vez de *octtree*. [27]

En una estructura octree, cada nodo subdivide el espacio que representa en ocho octantes. En una región punto (PR) octree, el nodo almacena un punto tridimensional explícito, el cual es el *centro* de la subdivisión para ese nodo; el punto que define una de las esquinas para cada nodo de los ocho hijos. En un octree MX, el punto de subdivisión es implícitamente el centro del espacio que el nodo representa. El nodo raíz de una PR octree puede representar un espacio infinito; el nodo raíz de un octree MX debe representar un espacio con límite finito para que los centros implícitos estén bien definidos. En las estructuras octree nunca se consideran los arboles kd, ya que los árboles kd se dividen en una dimensión mientras que las estructuras octree se dividen alrededor de un punto. Los árboles kd además son siempre binarios, lo cual no se cumple en las estructuras octree.

Geometría solida constructiva. La **geometría sólida constructiva** (CSG, por sus siglas en inglés Constructive Solid Geometry), se define como la representación de un objeto mediante un árbol binario, donde los nodos no terminales representan operaciones booleanas (unión, diferencia, intersección y complemento) y los nodos hojas representan sólidos primitivos.

Cuando se representa un objeto mediante árboles CSG se utiliza una serie de primitivas que al combinarse por operadores booleanos generan el objeto deseado [23]. En nuestro trabajo de investigación, esta técnica de modelado será empleada para representar el modelo tridimensional del ambiente explorado. También un cuerpo sólido funcionará como el rayo que emite el sensor que se intersectará con el solido que emule el ambiente y aplicando la operación de intersección, obtendremos el modelo del am-

biente. En la Figura 1.5 se presenta un ejemplo de un modelo obtenido con la técnica de la Geometría Solida Constructiva.

Figura 1.5: Ejemplo de la operación sustracción de la geometría sólida constructiva.

Voxel coloring. Voxel Coloring es una técnica utilizada para crear una reconstrucción de la escena 3D a partir de un conjunto bastante disperso de imágenes del ambiente. Tal vez el aspecto más útil de esta técnica es el hecho de que puede crear una reconstrucción de la escena fotográfica sin correspondencia de imagen computacional. En [25] se reporta un mecanismo para realizar la representación de un ambiente tridimensional con la técnica de Voxel Coloring. La técnica de S. Seitz y Dyer [25] trabaja por medio de la descomposición de la escena para ser reconstruida en pequeños elementos de volumen, llamados voxeles.

En lugar de la correspondencias entre imágenes computacionales y el uso de la triangulación para reconstruir la geometría de la escena, la técnica del voxel coloring atraviesa el volumen de escena voxel por voxel, re-proyectando cada uno de vuelta en el conjunto de la imagen y computando una medida de la variación de color de los píxeles en la que se proyecta.

Si la diferencia entre los píxeles es baja, el voxel es considerado el color invariable, y

en consecuencia se colorea. De lo contrario, la voxel permanece sin color. Después de un barrido del volumen de toda la escena, todos los voxeles correspondientes a los puntos de la escena deben ser de color, mientras que los que no corresponden a los puntos de la escena quedan sin color [10].

Toda la escena se puede reconstruir en un solo paso, haciendo uso de la restricción de visibilidad ordinaria. En concreto, por la descomposición de la escena en capas voxel e iniciar el barrido con la capa más cercana al volumen de la cámara, todas las oclusiones (oclusión: se usa para describir la manera como un objeto cercano al viewport mask se cierra) pueden ser explicadas por la retención para cada imagen en la imagen de conjunto de una máscara binaria, en un principio los ceros, con motivo de que los píxeles ya se han utilizado para colorear un voxel. Por ejemplo, cuando se realiza el voxel coloring para una botella de salsa Tabasco en posición vertical, el algoritmo comienza en la tapa y se procede a la base. Los píxeles de la imagen utilizada para la tapa de color voxels están marcados y no se consideran en la coloración de la base, incluso si un voxel en la base pasa a proyectar en algunos de ellos [12].

Cada voxel visitado es proyectado sobre el plano de cada imagen en el conjunto imagen usando parámetros de la cámara calculados durante la calibración. En general, un voxel, que es un cubo, proyecta una especie de hexágono en el plano de la imagen. Para simplificar el algoritmo, se elige un lugar para proyectar las esquinas del voxel en el plano de la imagen y luego usar el mínimo y máximo x,y y re-proyectar las coordenadas para crear una huella rectangular. Una vez que la huella de un voxel es determinada, se encuentran los píxeles sin marcar en ella, y sus valores medios de rojo, verde y azul se calculan. Si el valor RGB calculado está dentro de un cierto umbral suministrado por el usuario del color de fondo, el voxel que se examina es considerado como un voxel de fondo y no de color. De lo contrario, la suma de los cuadrados de las diferencias se calcula entre los valores de los píxeles sin marcar y el valor medio para cada canal y se añade a un total acumulado que se mantiene para cada voxel sobre el conjunto de imágenes de entrada. Si, después de que su presencia se ha calculado para cada una de las imágenes de entrada, un voxel está decidido a no ser un voxel de fondo, estos totales se utilizan para calcular la desviación estándar de cada canal, y las desviaciones estándar para los tres canales se promedian y se comparan con un umbral del usuario.

Si la desviación estándar media es menor que el valor umbral, el voxel es considerado consistente y se colorea con el color medio de sus huellas, y todos los píxeles dentro de las huellas del voxel se marcan. En el Algoritmo 3 se presentan los pasos para reconstruir una escena 3D con la técnica de Voxel Coloring.

Algoritmo 3: Algoritmo Voxel Coloring

```
1 S = ∅
2 for i = 1 a r do
3     for v ∈ r do
4         if existe la suficiente relación de colores de los pixeles then
5             S.Agregar(v)
```

1.2.2. Estrategias de reconstrucción de ambientes 3D

Mapeo de ambientes desconocidos con robots móviles. Un trabajo reciente de Huwendi aborda el problema de la reconstrucción de ambientes tridimensionales mediante el uso de robots móviles cooperativos. La propuesta descrita en [13] consiste en realizar la reconstrucción automática del mapa tridimensional de entornos interiores con un buen tiempo de exploración usando múltiples robots. Se emplean diferentes tipos de sensores en múltiples robots autónomos con dos algoritmos diferentes. Uno de los algoritmos es un algoritmo que permite realizar la extracción de características usando una cámara estereoscópica para construir un mapa tridimensional basado en características. El otro algoritmo permite extraer las características geométricas a partir de imágenes de rango para construir un modelo tridimensional del ambiente. La idea del trabajo de Huwendi es que un robot tome el rol de *Maestro* y otros robots *esclavos* le envíen la información necesaria para reconstruir el mapa tridimensional del ambiente.

El argumento empleado por el autor para defender la idea de usar múltiples robots móviles consiste en afirmar que diversas pruebas reportadas por la literatura ofrecen mejores tiempos de exploración cuando los robots cooperativos realizan la tarea en lugar de uno solo [5], [7], [11], [18], [19], [27].

De acuerdo a [5] el uso de múltiples robots nos brindan demasiados beneficios en

comparación con los sistemas de un solo robot. Además, la cooperación de robots tiene el potencial de realizar una tarea más rápido que un solo robot [11]. En el año de 1993, Tan realizó una comparación entre agentes independientes y agentes cooperativos y probó que los agentes cooperativos se deben comunicar para mejorar el rendimiento grupal. Esto lo logran compartiendo la información sensada por cada uno de los agentes y transmitiendo experiencias pasadas y conocimiento aprendido [27]. Dos años después, en 1995, Mataric et. al [18] presenta un enfoque que consiste en utilizar la cooperación robótica en tres niveles: sensado, acción y control; este trabajo en particular enfatiza en que el uso de un protocolo de comunicación mejora el rendimiento de los robots cooperativos. Otros trabajos recientes que defienden el argumento de la cooperación robótica son los presentados en [19], [29], [23], [7], [5].

Mapas de superficie multinivel. Los mapas de superficie multinivel MLS (MLS, por sus siglas en inglés multi-nivel surface) fueron propuestos por Triebel et al [28]. Los mapas MLS emplean una estructura de malla bidimensional que almacena diferentes valores de altitud. En particular, se almacenan en cada celda de una malla discreta la altura de la superficie en el área correspondiente. A diferencia de los mapas de elevación, los MLS nos permiten almacenar múltiples superficies en cada celda. Cada superficie se representa por una campana Gaussiana con la media de elevación y su varianza σ. Esta representación le permite a un robot móvil modelar los entornos con estructuras como puentes, pasajes estrechos, edificios o minas. También permiten al robot representar estructuras verticales mediante el almacenamiento del valor vertical de las estructuras que se desean representar. En la Figura 1.6 se muestra un ejemplo de un mapa MLS.

1.3. Aportaciones

En este proyecto de investigación se propone la creación de un sistema robótico para realizar la exploración autónoma de ambientes tridimensionales. Para realizar este sistema robótico se han seleccionado diversas estrategias probabilistas reportadas en la literatura mencionada anteriormente. Las estrategias SRT y SET requieren adaptarse para realizar la tarea de exploración en ambientes tridimensionales, estas adaptaciones

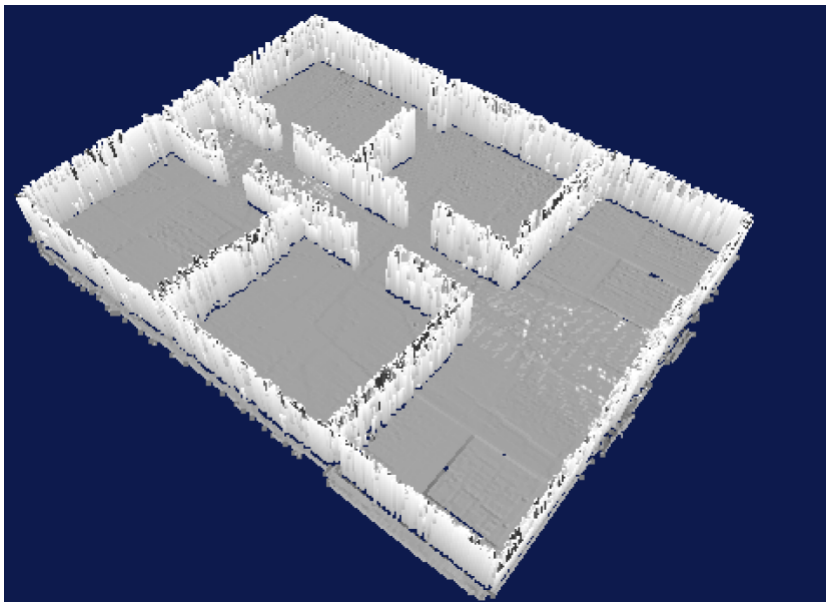


Figura 1.6: Ejemplo de mapa MLS

son unas de las principales aportaciones de este trabajo. Los capítulos 2 y 3 explican la adaptación de estas estrategias para que el manipulador realice la exploración en ambientes 3D. Una vez descritos los mecanismos de exploración del manipulador, en el capítulo 4 se describe el algoritmo híbrido formado por el robot móvil y el robot manipulador para mejorar la cobertura de la exploración. Finalmente, en el capítulo 5 se presentan las conclusiones y trabajos futuros de nuestra investigación.

Capítulo 2

Árboles aleatorios de exploración basado en sensores (SRT)

El método de exploración SRT, se presenta bajo la suposición de una perfecta localización del robot, provista por otro módulo. Esto puede suceder en ocasiones (por ejemplo, con un sistema GPS usado en misiones planetarias), pero no podemos omitir que tal suposición a menudo es ilógica en ambientes desconocidos y no estructurados. Éste método se introduce con ciertas consideraciones sobre el robot y el ambiente de trabajo.

2.1. Algoritmo básico de construcción del SRT

Ya se ha mencionado que el método SRT construye una estructura de datos llamada árbol aleatorio de exploración usando sensores (SRT), que puede considerarse como una variación del árbol aleatorio de exploración rápida (RRT). Así como el RRT, el SRT es un árbol que representa el roadmap del espacio de configuraciones libre. Cada nodo del SRT consiste de una configuración q libre de colisión que ha alcanzado el robot, junto con la descripción de la región segura local S circundante a q percibida por los sensores. El árbol se construye gradualmente, extendiendo la estructura hacia direcciones seleccionadas aleatoriamente de tal manera que la nueva configuración (y el camino que lleva a ella) esta contenida en la región segura local. El algoritmo que implementa el método SRT se describe en el Algoritmo 4.

En cada iteración k del algoritmo, se efectúa un proceso de percepción (es decir, sensado del ambiente y recopilación de datos), para obtener la región S que estima el espacio libre circundante al robot en la configuración actual, q_{act}. Un nuevo nodo, que

Algoritmo 4: Algoritmo básico de construcción del SRT

Data: $q_{ini}, K_{max}, I_{max}, \alpha, d_{min}$
Result: $\mathcal{T}$

```
1  q_act = q_ini
2  for k = 1 a K_max do
3      S ← PERCEPCION(q_act)
4      AGREGA(T, (q_act, S))
5      i ← 0
6      repeat
7          θ_rand ← DIR_ALEATORIA
8          r ← RADIO(S, θ_rand)
9          q_cand ← DESPLAZAR(q_act, θ_rand, α · r)
10         i ← i + 1
11     until VALIDA(q_cand, d_min, T) o i = I_max;
12     if VALIDA(q_cand, d_min, T) then
13         MOVER_A(q_cand)
14         q_act ← q_cand
15     else
16         MOVER_A(q_act.padre)
17         q_act ← q_act.padre
```

contiene la configuración q_{act} y su RSL asociada, se agrega al árbol $\mathcal{T}$. La forma de representar S en la estructura SRT depende de la estrategia de percepción: en general, podría usarse una descripción algebraica de sus límites.

En el punto de la configuración actual, la función DIR_ALEATORIA genera una dirección aleatoria de exploración θ_{rand} y la función RADIO calcula el radio r de S en la dirección θ_{rand}, ver la Figura 2.1. Una nueva configuración candidata q_{cand} se determina tomando un paso de longitud $\alpha \cdot r$ en dirección a θ_{rand}. La constante $\alpha \leq 1$ garantiza que q_{cand} se encuentre en el área segura S y puede alcanzarse a través de un camino contenido en S; valores próximos a 1 incrementan la capacidad de exploración del algoritmo, mientras que los valores más pequeños aumentan el margen de seguridad.

Una vez generada q_{cand} de forma aleatoria en la región segura S, pasa por un proceso de validación efectuado por la función VALIDA. Como se muestra en la Figura 2.1, q_{cand} (i) debe estar alejada de q_{act} a una distancia mayor a una distancia mínima prefijada

d_{min} y (ii) no debe situarse en la región segura local de otra configuración previa en $\mathcal{T}$. Si la validación tiene éxito, el robot se mueve a q_{cand} y el ciclo se repite. De lo contrario, el algoritmo genera otras configuraciones aleatorias desde q_{act} hasta encontrar una configuración válida o exceder un número máximo de intentos, I_{max}. En el último caso, el robot regresa al nodo padre de q_{act}, donde se ejecuta nuevamente el ciclo. Típicamente, cuando el espacio libre ha sido explorado completamente, el algoritmo fallará en encontrar una nueva dirección de exploración y el robot hará un proceso automático de retorno a la configuración inicial.

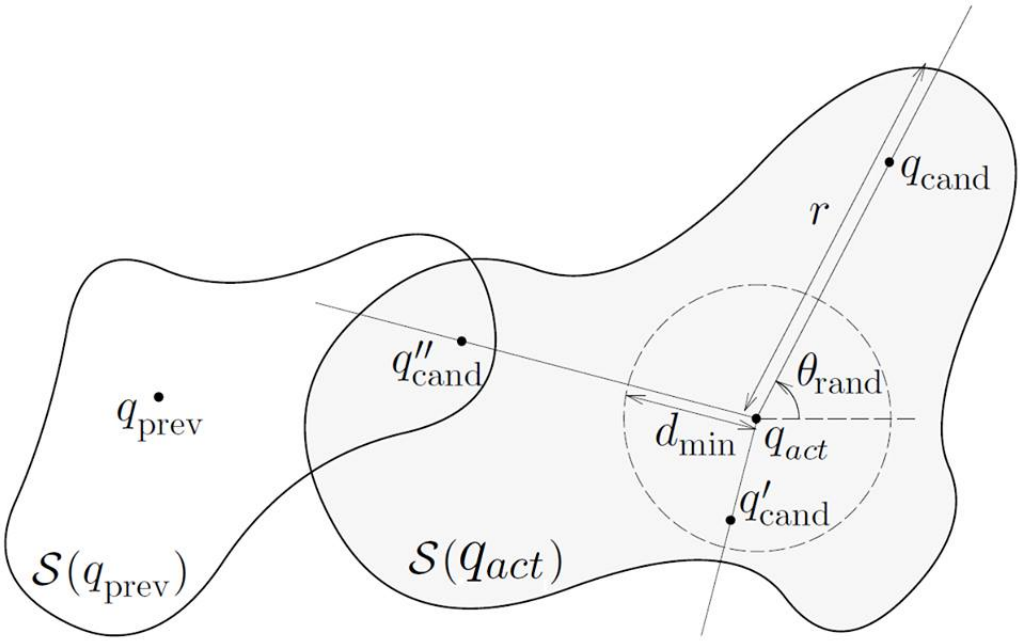


Figura 2.1: Generación de configuraciones candidatas con el método SRT. En este caso, q_{cand} es valida, mientras $q^{'}$ y $q^{''}$ no lo son, la primera se encuentra a una distancia menor a d_{min} de q_{act} y $q^{''}$ se ubica en la región segura local de otro nodo.

2.2. SRT adaptados para ambientes 3D

Uno de los principios de esta adaptación consiste en que la RLS es un poliedro irregular, con ello, el mecanismo para construir el árbol SRT debe estar en función de

encontrar el siguiente punto de exploración en ese poliedro con las mismas condiciones que impone el algoritmo básico de construcción del SRT. En primer lugar se debe considerar que el algoritmo básico de SRT asume que el robot móvil está en el centro de la RLS, ya que el móvil emite los rayos láser que forman la RLS, sin embargo en nuestro caso, el emisor del mecanismo de sensado es el efector final del manipulador, por lo tanto, no se puede asumir que la posición de éste sea el centro de nuestro poliedro que representa a la RLS. Es por lo anterior que es necesario encontrar el centroide del poliedro que representa nuestra RLS y a parir de él, encontrar la siguiente configuración de exploración.

La adaptación de este algoritmo consiste en generar dos direcciones aleatorias (θ_{rand_x} y θ_{rand_y})en vez de una, para que el RADIO que se necesita para generar la configuración q_{cand} sea el RADIO de una esfera contenida en el poliedro que representa la Region Local Segura. En la Figura 2.2 se muestra la representación de esta adaptación.

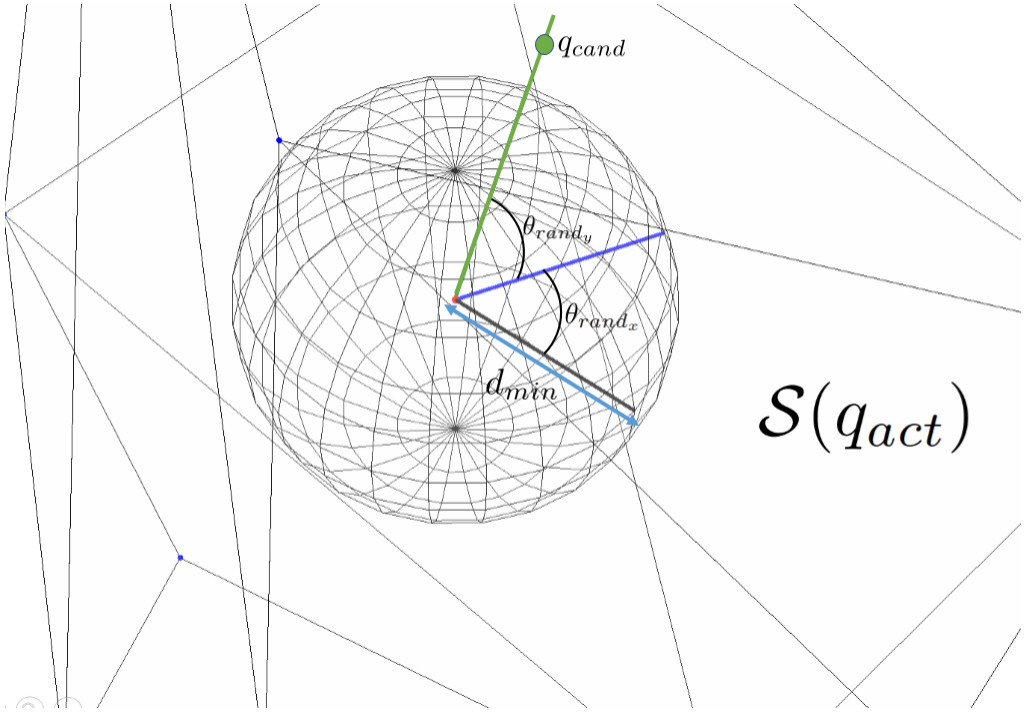


Figura 2.2: Adaptación de SRT para ambientes 3D

A partir de la generación de la primer dirección aleatoria θ_{rand_x} de rotación sobre

el eje X, se obtiene un RADIO temporal, que después se rota en la dirección aleatoria θ_{rand_y} sobre el eje Y para obtener el RADIO sobre el cual se genera la configuración candidata q_{cand}, respetando la restricción de estar mas lejos que una distancia d_{min}, en este caso, del centroide del poliedro.

2.3. Resultados

2.3.1. Cálculo de la cobertura

Uno de los índices de rendimiento para las estrategias de exploración es la cobertura que éstas tienen del ambiente explorado. Dado que los resultados que se presentan en esta clase de estrategias están basadas en simulaciones, el ambiente a explorar por el robot, es previamente conocido por el sistema que realiza la exploración. Con el conocimiento del ambiente a explorar, el cálculo de la cobertura consiste en determinar el porcentaje que representa el ambiente explorado por el robot, respecto al ambiente completo.

En nuestro caso, el cálculo del porcentaje será computado mediante el volumen del poliedro irregular que representan los objetos que conforman el ambiente (explorado y conocido). El cálculo de este volumen se realizó utilizando la simplificación del teorema de la divergencia, la cual afirma que una región cerrada por un poliedro con caras planares y un vector norma por cada una de estas caras está dado por:

$$V = \left(\frac{1}{3}\right) \sum_{i-ésima\_cara} \vec{x_i} \cdot \hat{n}_i A_i$$

Donde para cada i-ésima cara del poliedro, $\vec{x_i}$ es el centroide de esa cara, $\hat{n}_i$ es el vector normal de esa cara y A_i es su área.

2.3.2. Resultados experimentales de la estrategia SRT

Los resultados que se presentan en esta sección se obtuvieron mediante la ejecución de la simulación del algoritmo SRT para ambientes 3D que se propone en este trabajo de investigación. Estos resultados fueron obtenidos en una computadora con un procesador

de 64 bits de cuatro núcleos a 3.4 Ghz y 16 GB de memoria RAM

Los escenarios a los que fue sometido el manipulador empleado por el sistema robótico propuesto son sugeridos por la bibliografía estudiada. Para homogeneizar nuestros resultados con el resto de la literatura, utilizaremos los índices de rendimiento sugeridos en [9]. Estos índices son:

- Número de Escaneos (NE). Consiste en el número total de visiones del sensor almacenadas en las estructuras de datos al final de la exploración.
- Tiempo de Simulación (TS). Consiste en el tiempo total requerido para simular la exploración robótica.
- Número de nodos en el SRT (ND). Consiste en el número total de nodos agregados al SRT para realizar la tarea de exploración.
- Cobertura (C). Consiste en el porcentaje del volumen de la region explorada, respecto al volumen del ambiente completo.

Para realizar las simulaciones de SRT se emplearon diversos valores para α, pero el resto de los parámetros usados fueron constantes con los siguientes valores:

- **KMax**:30. Representa el número de iteraciones que realiza el algoritmo
- **IMax**:15. Representa el número de intentos que realiza el algoritmo para encontrar una configuración válida.
- **dmin**:1.5. Representa la distancia mínima, a la que debe estar la configuración candidata.

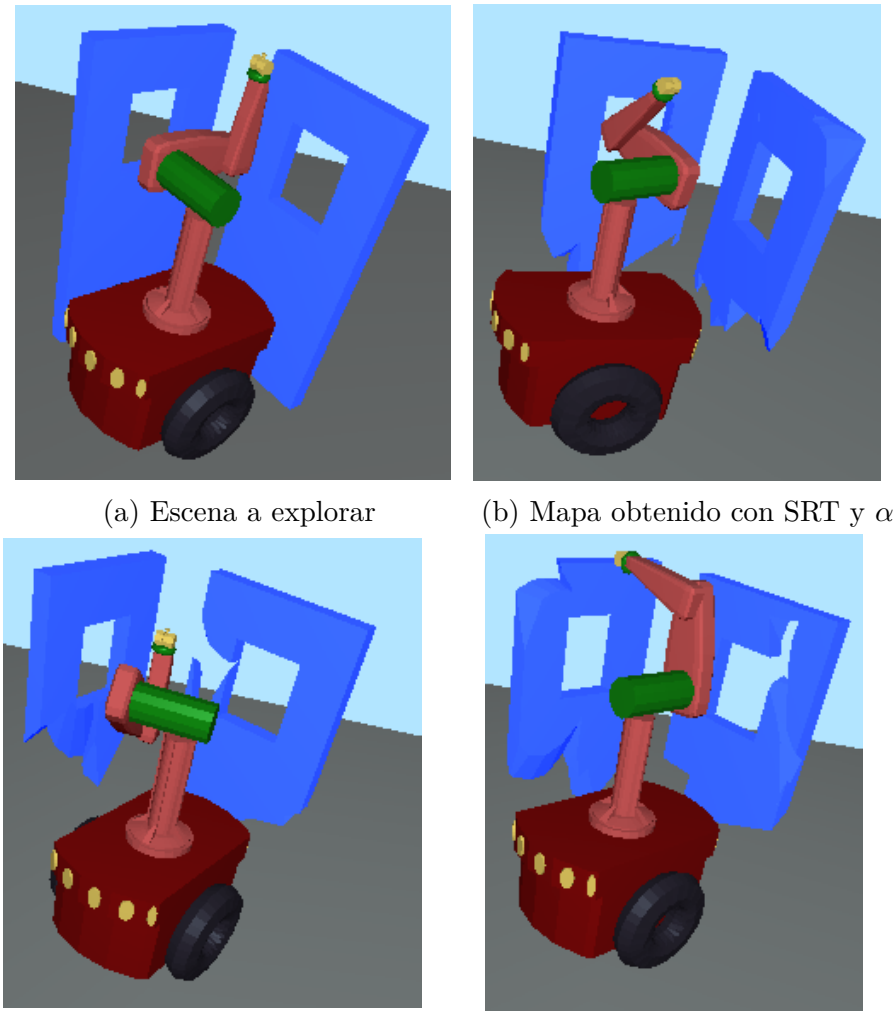
Los resultados obtenidos se presentan en el Cuadro 2.1. El mapa obtenido en cada una de las simulaciones se presenta en la Figura 2.3.

2.4. Conclusiones

La adaptación propuesta del algoritmo SRT para ambientes 3D arrojó buenos resultados en tiempo de ejecución y cobertura del ambiente para los escenarios en los

α	**NE**	**ND**	**TS**	**Cobertura**
0.6	28	30	60.57 segundos	63.1 %
0.7	29	30	58.348 segudnos	42.506 %
0.8	29	30	60.489 segundos	84.63 %
0.9	27	30	64.31 segundos	45.633 %

Cuadro 2.1: Resultados de la simulación de SRT-3D.



(a) Escena a explorar (b) Mapa obtenido con SRT y $\alpha = 0.6$

(c) Mapa obtenido con SRT y $\alpha = 0.7$ (d) Mapa obtenido con SRT y $\alpha = 0.8$

Figura 2.3: Resultados de la exploración con SRT y diferentes valores de α.

que fue probado. Es notoria la importancia del parámetro α en los resultados, ya que con valores menores a 1 (tal como sugiere la literatura), la cobertura del ambiente es mejor que con valores mayores a 1, debido a que este parámetro determina el *salto* desde el centroide de la RLS hacia la siguiente configuración candidata. Durante la ejecución de las pruebas se observó que debido a la naturaleza probabilista del algoritmo de construcción del SRT, en ocasiones valores *altos* de KMax, tales como KMax = 50, con $\alpha = 0.8$ no reportan mejores coberturas del ambiente que las simulaciones con un menor valor para el parámetro KMax, lo cual indica que no necesariamente un elevado número de iteraciones del algoritmo, va a resultar en una mejor cobertura del ambiente, ésto último depende en gran medida del tipo de muestreo que se use para generar las configuraciones aleatorias en la construcción del SRT.

Capítulo 3

Árboles de exploración basado en sensores (SET)

En el método SET, el robot construye gradualmente el *árbol de exploración basado en sensores*. Cada nodo del SET representa una configuración de visión, mientras que un arco entre dos nodos representa una ruta segura que une las dos configuraciones de visión.

3.1. Algoritmo básico de construcción del SET

Algoritmo 5: Pseudocódigo que describe la k-ésima iteración del método SET, en la cual el robot comienza en la configuración q^k

```
1  Actualizar el modelo del ambiente y el SET
2  Extraer la Frontera Local Segura FLS(q^k, k)
3  if FLS(q^k, k) ≠ ∅ then
4  |  (q^{k+1}, U(q^{k+1})) ← Búsqueda de la configuración con la máxima utilidad
   |  dentro del conjunto admisible D(q^k, k)
5  else
6  |  U(q^{k+1}) ← 0
7  if U(q^{k+1}) ≥ U_min then
8  |  Planificar un camino segura que conecte a q^k con q^{k+1}
9  |  Moverse hacia q^{k+1} y adquirir la vista del sensor
10 else
11 |  Moverse hacia la configuración q
```

Un ciclo del algoritmo SET se muestra en el Algoritmo 5. El robot comienza en

una configuración q^k. Primero se actualiza la estructura de datos del SET y el modelo del ambiente. Si una nueva configuración de visión se alcanza en la iteración previa, el nuevo nodo correspondiente se inserta en el SET, un nuevo arco se crea entre el nodo anterior y el nuevo nodo, también, se agrega la nueva visión alcanzada al modelo del ambiente $(\epsilon^k, \partial\epsilon^k_{obs})$.

Después, la Frontera Local Segura FLS (q^k, k) se extrae. Esta FLS contiene alguna porción de la frontera segura la cual es visible por el sensor desde alguna configuración $q \in \mathcal{D}(q^k, k)$. La FLS cumple la siguiente implicación:

$$FLS(q^k, k) = \emptyset \Rightarrow \mathcal{D}(q^k, k) \cap \mathcal{Q}^k = \emptyset \tag{3.1}$$

Lo anterior quiere decir que es necesario verificar que $\text{FLS}(q^k, k) = \emptyset$ antes de buscar una configuración de vista en $\mathcal{D}(q^k, k) \cap \mathcal{Q}^k$; si $\text{FLS}(q^k, k) = \emptyset$ se debe forzar a un retroceso. La estrategia de selección adoptada en este trabajo consiste en la maximización de una función de utilidad U en $\mathcal{D}(q^k, k)$. De acuerdo a la implicación 3.1, cuando la frontera local segura esta vacía, no se ejecuta la búsqueda y se establece $U(q^{k+1})$ a cero, forzando a un retroceso.

En este punto, la utilidad $U(q^{k+1})$ de la configuración de visión candidata q^{k+1} se compara con U_{min} que es un umbral fijo. Esta comparación tiene un doble objetivo: (i), obliga un paso de retroceso cuando la FLS esta vacía (i.e., cuando $U(qk + 1)$ vale cero) y (ii) filtra casi todas las acciones inútiles del robot. Por lo tanto, si $U(q^{k+1}) \geq U_{min}$, entonces q^{k+1} se convierte en la siguiente configuración de vista, el planificador del camino se invoca para calcular el camino desde q^k hacia q^{k+1}, q^{k+1} se alcanza y una nueva vista se adquiere. De otra manera, el paso de retroceso se ejecuta y el robot se mueve a la configuración q^k.

Cuando el robot no puede ejecutar avance en la construcción de las fronteras locales visitadas, se obliga un retroceso hacia la raíz del SET (la configuración inicial), así se realiza un mecanismo automático de *mensajería*.

3.2. Estrategias de obtención de la función de utilidad

El Algoritmo 5 contempla la existencia de una función $U(q^k)$ que determine la utilidad de la configuración q. La importancia de esta función de utilidad radica en que ésta es la que dirige la construcción del SET, y es por ello que en nuestro trabajo de investigación hemos propuesto dos estrategias para calcular esta función de utilidad.

3.2.1. Obtención de la función de utilidad mediante Algoritmos Genéticos

La primer estrategia de obtención de la función de utilidad de una configuración q^k, es formulando el problema de maximización de la función de utilidad con un Algoritmo Genético (descritos en [20]). Esta formulación consiste en ver el problema de maximización de la función de utilidad como un problema en el cual se busca obtener la configuración q^{k+1} dentro de la RLS de q^k. En este caso la población consiste en todas las configuraciones candidatas a ser la configuración q^{k+1}. El Algoritmo Genético (AG) se aplicará para buscar entre esta población al individuo que obtenga mejor utilidad. Por lo anterior, el objetivo del AG es hallar una configuración que devuelva la mejor *aptitud*, es decir, el volumen máximo posible para la RLS de la configuración q^{k+1}.

A continuación daremos brevemente algunos detalles de la implementación del algoritmo genético que nos permite obtener la máxima utilidad posible.

Generación de la población inicial

Como se ha mencionado anteriormente, la población consta de de un conjunto de configuraciones en la RLS de q^k. Estas configuraciones son tomadas aleatoriamente de la superficie del poliedro, es decir, son puntos aleatorios pertenecientes a las caras que conforman el poliedro. El fenotipo de cada individuo es la posición de cada individuo en el conjunto de las configuraciones candidatas. El genotipo consiste en la representación binaria de esta posición del individuo. En las siguientes secciones brindaremos un panorama de la manera en la que se realizan las operaciones genéticas.

Función de aptitud

Como se mencionó al principio de la sección, la función de aptitud consiste en el volumen del poliedro, que representa la RLS, que resulta del proceso de sensado del manipulador, en la configuración localizada en la posición representada por el fenotipo de cada individuo de la población.

Cruza

La cruza aplicada al genotipo de la población consiste en realizar la cruza en un punto en dos individuos de la misma. De manera aleatoria se selecciona un punto de cruza, a partir del cual se generan dos nuevos individuos de la manera tradicional de cruza. Una vez generados los nuevos individuos, se procede a verificar que el fenotipo correspondiente a estos nuevos genotipos es un individuo válido, es decir, que represente una posición existente en la población de configuraciones candidatas. En caso de que el fenotipo de alguno de los nuevos individuos exceda el tamaño de la población y por lo tanto no sea un individuo válido, se aplicara el operador MOD al fenotipo inválido respecto al tamaño de la población, para así, obtener una posición válida en la población y finalmente generar el genotipo correspondiente al nuevo individuo válido y agregarlo a la nueva generación.

Mutación

El proceso de mutación consiste en tomar el genotipo del individuo, seleccionar de manera aleatoria una posición en el mismo y cambiar el valor del cromosoma en esa posición: si el cromosoma es 1, se cambia por 0 y viceversa. Una vez obtenido el nuevo genotipo, se aplica el mismo proceso de validación del individuo que en el proceso de cruza.

Estrategia de selección

La estrategia de selección empleada en nuestra propuesta, es la selección elitista. En este enfoque, cada generación es evaluada mediante su función de aptitud y los individuos que reporten mejor aptitud son seleccionados para formar la siguiente generación.

Con este enfoque se busca que cada nueva generación mejore la aptitud de la población y esto se vea reflejado en un mayor volumen de exploración.

Algoritmo Genético

Finalmente, presentaremos el pseudocódigo del AG implementado para aproximar la función $U(q^k)$.

Algoritmo 6: Pseudocódigo del AG que aproxima la función $U(q^k)$

```
1 P = GenerarPoblaciónInicial
2 for k = 1 a NumeroDeGeneraciones do
3 |   CalcularAptitud(P)
4 |   M = ObtenerMejoresindividuos(P)
5 |   AplicarOperadorDeMutacion(M)
6 |   AplicarOperadorDeCruza(M)
7 |   P = CrearNuevaPoblacion(M)
```

3.2.2. Obtención de utilidad mediante puntos en la frontera

Debido a los resultados obtenidos mediante la estrategia anterior (reportados en la siguiente sección), se observó un alto tiempo de simulación, por lo que se optó por buscar otra estrategia para aproximar la función $U(q^k)$. Esta nueva estrategia consiste en elegir aleatoriamente n centroides de las caras que forman el poliedro. Cada centroide representa una configuración candidata a ser la configuración q^{k+1}. La diferencia con la estrategia del AG, consiste en que no se va a ejecutar un cierto número de generaciones, si no que simplemente se evalúan los volúmenes de cada una de las configuraciones seleccionadas y la que arroje mejor volumen, es la seleccionada como la configuración q^{k+1}.

3.3. Resultados experimentales de la estrategia SET

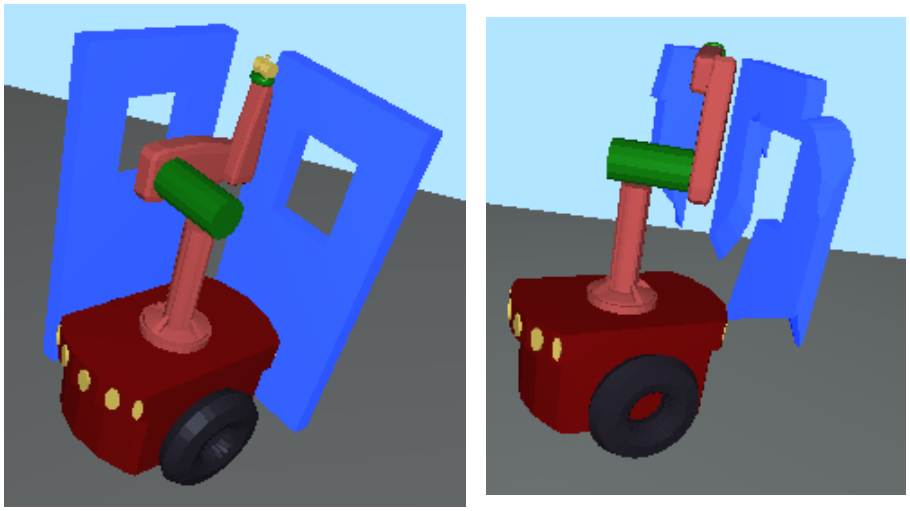
Esta sección presenta un resumen de los experimentos realizados con las dos estrategias para aproximar la función de utilidad necesaria para la construcción del SET.

Estos resultados fueron obtenidos en una computadora con un procesador de 64 bits de cuatro núcleos a 3.4 Ghz y 16 GB de memoria RAM. Los índices de rendimiento que se reportan son los mismos que se mencionaron y reportaron en la sección 2.3.

En el Cuadro 3.1 se presentan los resultados obtenidos al ejecutar la estrategia basada en el AG. El número de iteraciones empleadas en esta simulación son de 10. La cobertura correspondiente a esta prueba se muestra en la Figura 3.1.

Índices de rendimiento	
Número de nodos:	8
Llamadas al detector de colisiones:	11
Número de escaneos:	2646
Tiempo de Simulación:	48.098 minutos
Cobertura:	37.40 %

Cuadro 3.1: Resultados de la simulación con la estrategia basada en puntos en la frontera de la RLS



(a) Escena a explorar (b) Mapa obtenido con el método SET

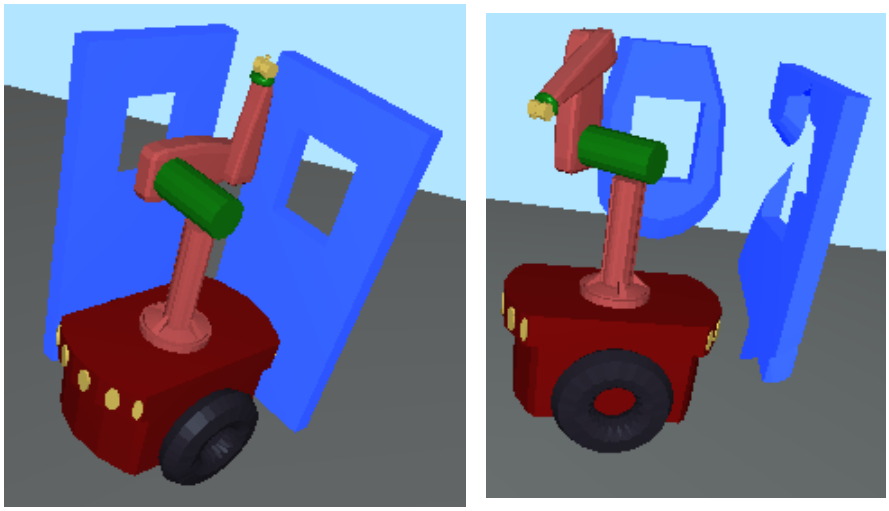
Figura 3.1: Resultados de la exploración con SET y la estrategia de AG.

En el Cuadro 3.2 se presentan los resultados obtenidos al ejecutar la estrategia basada en puntos en la frontera de la RLS. El número de iteraciones empleadas en esta

simulación son de 40, tal como las pruebas realizadas al método SRT. La cobertura correspondiente a esta prueba se muestra en la Figura 3.2.

Índices de rendimiento	
Número de nodos:	39
Llamadas al detector de colisiones:	41
Número de escaneos:	841
Tiempo de Simulación:	14.394083 minutos
Cobertura:	51.7 %

Cuadro 3.2: Resultados de la simulación con la estrategia basada en puntos en la frontera de la RLS



(a) Escena a explorar (b) Mapa obtenido con el método SET

Figura 3.2: Resultados de la exploración con SET y la estrategia basada en puntos en la frontera.

3.4. Conclusiones

Los resultados obtenidos en la ejecución de estas dos estrategias muestran un alto tiempo de simulación y una baja cobertura del ambiente (respecto a la cobertura observada en SRT) en los ambientes en los que se probaron ambas estrategias. Esto se debe,

sin duda, a la función de utilidad propuestas, ya que en ambas, es necesaria la revisión de un cierto número de configuraciones candidatas y eso consume la mayor parte del tiempo de cómputo. Estos resultados nos dan la pauta para afirmar que la estrategia de exploración de ambientes tridimensionales que se empleará en el sistema robótico general (con el robot móvil) será el método SRT.

Capítulo 4

Estrategia híbrida de exploración

En los capítulos anteriores, se han presentado algunas estrategias de exploración de ambientes tridimensionales, sin embargo uno de los objetivos de nuestro trabajo de investigación es la implementación de un mecanismo híbrido de exploración que realice la exploración móvil y la exploración tridimensional. En este capítulo se presenta una estrategia híbrida para la exploración de un ambiente tridimensional desconocido, basada en los árboles SRT.

4.1. Exploración móvil

Antes de describir el mecanismo de exploración propuesto, explicaremos la estrategia empleada para llevar a acabo la exploración con el móvil. Como se mencionó al principio del capítulo, esta estrategia de exploración móvil, está basada en los árboles SRT. Para realizar las pruebas de la estrategia de exploración móvil, se implementó el modelo tridimensional de un robot móvil holonómico llamado Pionner. El modelo del robot Pionner se muestra en la Figura 4.1.

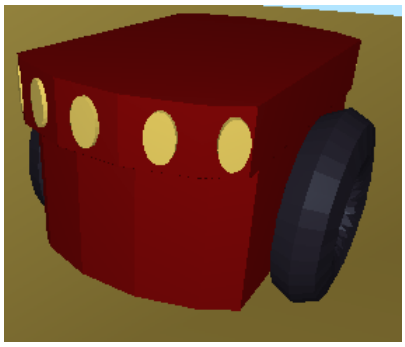


Figura 4.1: Modelo tridimensional del robot móvil Pionner

La idea de implementar la estrategia SRT en el robot móvil es para brindarle al robot manipulador la capacidad de moverse por el ambiente y de esta forma aumentar al cobertura de la exploración. A continuación se detalla la estrategia que combina la exploración móvil con la exploración tridimensional descrita en los capítulos anteriores.

4.2. Algoritmo de la estrategia de exploración propuesta

La idea de nuestra propuesta consiste en explorar un ambiente tridimensional ejecutando alternadamente los mecanismos de exploración de la estrategia de SRT adaptada para ambientes 3D (descrita en el capítulo 2) y la estrategia de SRT para el móvil (descrita en la sección anterior).

En el Algoritmo 7 se presenta el pseudocódigo de la estrategia de exploración propuesta. En general, la propuesta consiste en realizar la exploración tridimensional con la adaptación de SRT para ambientes 3D y después encontrar el siguiente punto de exploración en el ambiente, mediante una iteración de la estrategia SRT del móvil.

El Algoritmo 7 presenta una llamada al Algoritmo 8. El Algoritmo 8 es una adaptación del algoritmo básico de construcción del SRT, la cual solo ejecuta una iteración del algoritmo básico, a partir de una zona de exploración dada por la configuración q_{ant}. De esta forma, la exploración del móvil devolverá una configuración válida según el algoritmo de construcción de SRT.

4.3. Resultados experimentales de la estrategia híbrida

En esta sección se presentan los resultados del algoritmo híbrido propuesto. En primer lugar, se debe mencionar que estos resultados fueron obtenidos en una computadora con un procesador de 64 bits de cuatro núcleos a 3.4 Ghz y 16 GB de memoria RAM. De la misma forma que los algoritmo presentados anteriormente, se toman en cuenta los índices de rendimiento de exploración sugeridos por la literatura [9], [23], [7], [21].

Algoritmo 7: Algoritmo híbrido de exploración

Data: $q_{ini}, K_{max_{3D}}, K_{max_{2D}}, I_{max_{3D}}, I_{max_{2D}}, \alpha_{2D}, \alpha_{3D}, d_{min_{2D}}, d_{min_{3D}}$

$q_{act} = q_{ini}$
$q_{ant} = q_{ini}$
for $k = 1$ *a* $K_{max_{2D}}$ **do**
  for $kk = 1$ *a* $K_{max_{3D}}$ **do**
    $\mathcal{S} \leftarrow$ PERCEPCION_MANIPULADOR(q_{act})
    AGREGA($\mathcal{T}$, $\mathcal{S}$)
    $i \leftarrow 0$
    repeat
      $\theta_{rand_x} \leftarrow$ DIR_ALEATORIA $\theta_{rand_y} \leftarrow$ DIR_ALEATORIA
      $r \leftarrow$ RADIO($S, \theta_{rand_x}, \theta_{rand_y}$)
      $q_{cand} \leftarrow$ DESPLAZAR($q_{act}, \theta_{rand_x}, \theta_{rand_y}, \alpha_{3D} \cdot r$)
      $i \leftarrow i + 1$
    until *VALIDA*($q_{cand}, d_{min_{3D}}, \mathcal{T}$) *o* $i = I_{max_{3D}}$;
    if *VALIDA*($q_{cand}, d_{min_{2D}}, \mathcal{T}$) **then**
      MOVER_MANIPULADOR_A(q_{cand})
      $q_{act} \leftarrow q_{cand}$
    else
      MOVER_MANIPULADOR_A($q_{act}.padre$)
      $q_{act} \leftarrow q_{act}.padre$
  $q_{sig} \leftarrow EXPLORAR\_SRT\_MOVIL(q_{ant}, I_{max_{2D},\alpha_{2D},d_{min_2D}})$
  MOVER_MOVIL_A(q_{sig})
  $q_{ant} \leftarrow q_{sig}$

El primer ambiente que se ha sometido a su exploración, se presenta en la Figura 4.2. El primer experimento de exploración con la estrategia híbrida consiste en ejecutar 10 iteraciones (parámetro $K_{max_{2D}}$ del Algoritmo 7) de la exploración móvil y 5 iteraciones de la exploración tridimensional (parámetro $K_{max_{3D}}$ del Algoritmo 7). Los resultados de este experimento se muestran en la Figura 4.3.

En las Figuras presentadas anteriormente, se observa el *roadmap* que permite la navegación del móvil por el ambiente. Se observa que la zona de exploración es poca, debido al bajo número de iteraciones de la exploración móvil y tridimensional.

A continuación se presenta el resultado de otro experimento de exploración del

Algoritmo 8: Algoritmo que genera una configuración de exploración, a partir de SRT

Data: $q_{ant}, I_{max_{2D}}, \alpha_{2D}, d_{min_{2D}}$
Result: q_{sig}

$q_{act} = q_{ant}$
$S \leftarrow$ PERCEPCION_DEL_MOVIL(q_{act})
$i \leftarrow 0$
repeat
  $\theta_{rand} \leftarrow$ DIR_ALEATORIA
  $r \leftarrow$ RADIO(S, θ_{rand})
  $q_{cand} \leftarrow$ DESPLAZAR($q_{act}, \theta_{rand}, \alpha_{2D} \cdot r$)
  $i \leftarrow i + 1$
until *VALIDA*($q_{cand}, d_{min_{2D}}$) *o* $i = I_{max_{2D}}$;
$q_{sig} \leftarrow q_{cand}$

ambiente presentado en la Figura 4.2. En la Figura 4.4 se presenta el experimento realizado con $K_{max_{2D}} = 20$ y $K_{max_{3D}} = 5$.

4.4. Conclusiones

Para los ambientes en los que se probó la estrategia de exploración híbrida propuesta, ésta mostró una cobertura mayor que la exploración simple 3D, lo cual nos hace concluir que la hibridación propuesta cumple el objetivo de llevar al robot manipulador a configuraciones donde realiza un proceso de exploración que resulta en una mejora a la cobertura del ambiente. A pesar que los tiempos de simulación son altos, el hecho de que la cobertura mejore, nos lleva a concluir que vale la pena emplear esta estrategia para realizar la tarea de la exploración autónoma de ambientes 3D.

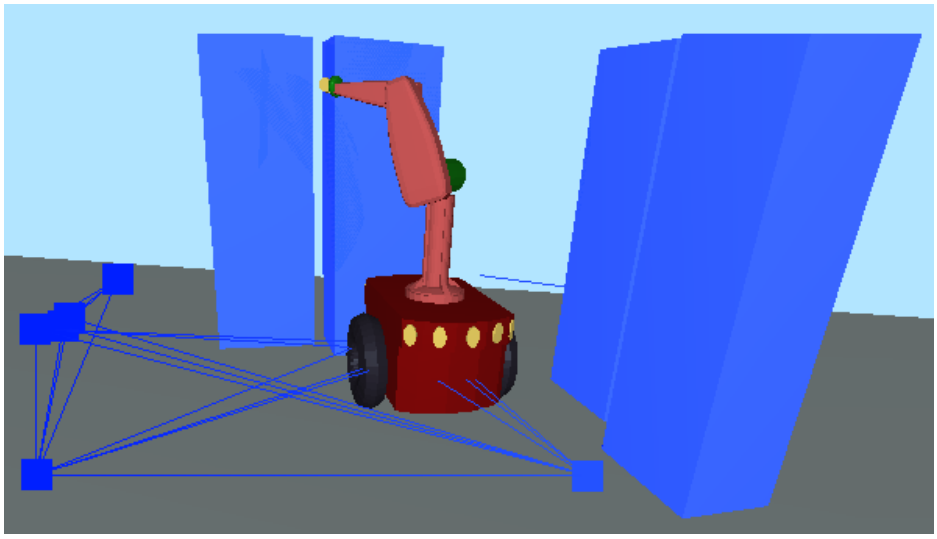


Figura 4.2: Ambiente completo a explorar con la estrategia híbrida

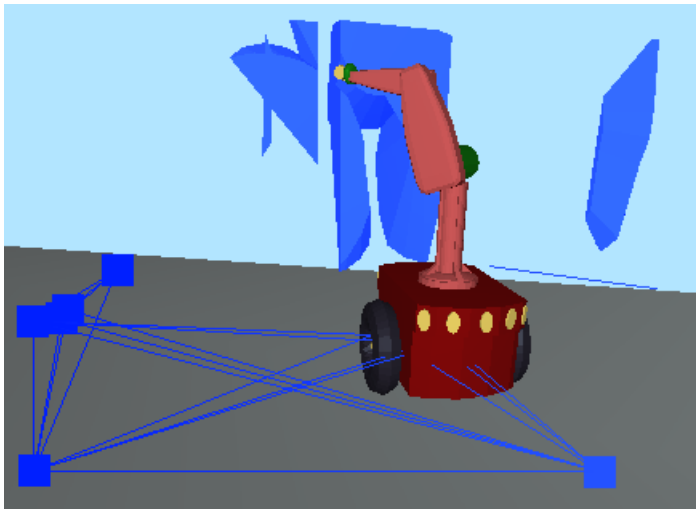


Figura 4.3: Ambiente explorado por la estrategia híbrida.

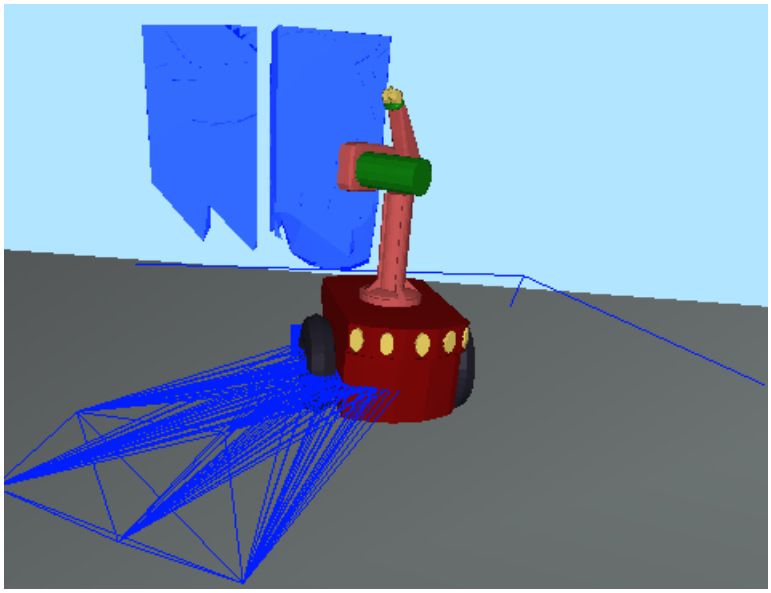


Figura 4.4: Experimento de exploración.

Capítulo 5

Conclusiones y trabajo futuro

Durante este trabajo de investigación se han implementado diversas estrategias probabilistas para crear un sistema que realice la exploración de ambientes tridimensionales de manera autónoma. Las estrategias implementadas hasta el momento son las SRT, adaptadas para ambientes 3D y el método SET con dos estrategias que aproximan una función de utilidad. A lo largo de los últimos dos capítulos de este documento se han presentado los resultados de ambas estrategias de exploración de manera separada, sin embargo, al momento de hacer un análisis sobre qué estrategia resultaría ser mejor para un sistema robótico de exploración, podríamos decir, en base a los resultados obtenidos por cada una, que SRT adaptado para ambientes tridimensionales, resulta ser más eficiente, ya que con un número de iteraciones relativamente bajo, obtiene una buena cobertura del ambiente a explorar. No por esto debemos decir que la estrategia de usar los árboles SET sea mala, pero podemos concluir que el bajo rendimiento de esta estrategia se debe a las estrategias de aproximación a la función de utilidad.

Es por lo anterior que uno de nuestros trabajos futuros se basan en la mejora de la estrategia de aproximación a la función de utilidad. Otro de nuestros trabajos futuros, que de hecho ya se esta terminando, es la implementación de una estrategia de exploración en un robot móvil y equipar a este último con el manipulador para realizar una exploración que obtenga mejor cobertura de ambientes desconocidos más grandes.

Bibliografía

[1] J.M. Ahuactzin and A. Portilla. A basic algorithm and data structures for sensor-based path planning in unknown environments. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 903–908 vol.2, 2000.

[2] Nancy M. Amato, O. Burchan Bayazit, and Lucia K. Dale. Obprm: An obstacle-based prm for 3d workspaces, 1998.

[3] Pierre Bessiere, Juan manuel Ahuactzin, El-Ghazali Talbi, and Emmanuel Mazer. The .ªriadne's clew.ªlgorithm: Global planning with local methods, 1993.

[4] M.Z. C. S. G. LEE. *A GEOMETRIC APPROACH IN SOLVING THE INVERSE KINMATICS OF PUMA ROBOTS*. 1983.

[5] Y. Uny Cao, Alex S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:226–234, 1997.

[6] G. Sarabia Cortés. Planificación reactiva de movimientos para robots y manipuladores usando técnicas probabilísticas. Master's thesis, FCC - BUAP, Puebla, México, 2012.

[7] Antonio Franchi, Luig Freda, Giuseppe Oriolo, and Marilena Vendittelli. A randomized strategy for cooperative robot exploration. In *IEEE International Conference in Robotics and Automation*, pages 768–774. IEEE, 2007.

[8] Luigi Freda and Giuseppe Oriolo. Frontier-based probabilistic strategies for sensor-based exploration. In *ICRA*, pages 3881–3887. IEEE, 2005.

[9] Luigi Freda, Giuseppe Oriolo, and Francesco Vecchioli. Sensor-based exploration for general robotic systems. In *IROS'08*, pages 2157–2164, 2008.

[10] J. Irving Vásquez Gómez. Planificación de vistas para reconstrucción tridimensional de objetos con robots móviles. Master's thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México, 2011.

[11] Didier Guzzoni, Adam Cheyer, Luc Julia, and Kurt Konolige. Many robots make short work: Report of the sri international mobile robot team. *AI Magazine*, 18(1):55–64, 1997.

[12] Rob Hess. Adventures in voxel coloring cs 559 term project, 2005.

[13] Ashraf SS Huwedi. 3d mapping of an unknown environment by cooperatively mobile robots. In *The International Arab Conference of Information Technology*, 2013.

[14] Lydia Kavraki, Petr Svestka, Jean claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, pages 566–580, 1996.

[15] J. C Latombe. *Robot motion planning*. Kluwer Academic Publications, 1991.

[16] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[17] Judith Espinoza León. Estrategias para la exploración de ambientes desconocidos en robótica móvil. Master's thesis, Facultad de Ciencias de la Computación - BUAP, Puebla, México, 2006.

[18] Maja J. Mataric, Martin Nilsson, Kristian T. Simsarian, and Proposed Maja J Matari'c. Cooperative multi-robot box-pushing. pages 556–561, 1995.

[19] Daniel Meier, Cyrill Stachniss, and Wolfram Burgard. Coordinating multiple robots during exploration under communication with limited bandwidth. In *In ECMR*, pages 26–31, 2005.

[20] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (2Nd, Extended Ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[21] Giuseppe Oriolo, Marilena Vendittelli, Luigi Freda, and Giulio Troso. The srt method: Randomized strategies for exploration. In *ICRA*, pages 4688–4694. IEEE, 2004.

[22] Mark H. Overmars, Petr Svestka, Mark H. Overmars, and Mark H. Overmars. A probabilistic learning approach to motion planning. In *In Proc. Workshop on Algorithmic Foundations of Robotics*, page pages, 1994.

[23] Alfredo Toriz Palacios. Estrategias probabilísticas para la exploración cooperativa de robots móviles. Master's thesis, Facultad de Ciencias de la Computación - BUAP, Puebla, México, 2007.

[24] Pamela Renton, Michael Greenspan, Hoda A. ElMaraghy, and Hassen Zghal. Plan-n-scan: A robotic system for collision-free autonomous exploration and workspace mapping. *Journal of Intelligent and Robotic Systems*, 24(3):207–234, 1999.

[25] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *International Journal of Computer Vision*, pages 1067–1073, 1997.

[26] Zheng Sun, David Hsu, Tingting Jiang, Hanna Kurniawati, and John H. Reif. Narrow passage sampling for probabilistic roadmap planning, 2005.

[27] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337. Morgan Kaufmann, 1993.

[28] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 06*, 2006.

[29] Ket ziquel Hernández Guadarrama. Estrategias de coordinación para la exploración con multi-agentes robóticos. Master's thesis, Facultad de Ciencias de la Computación - BUAP, Puebla, México, 2010.

Apéndice A

Estudio de los robots manipuladores

En este capítulo se presenta el estudio de los robots manipuladores. Comenzaremos con la definición de las cadenas cinemáticas, después presentamos un breve estudio de dos técnicas de representación de los robots manipuladores. En concreto se presentará el desarrollo de la cinemática inversa y directa del robot PUMA 560.

A.1. Las cadenas cinemáticas

Como se mencionó el capítulo B, un robot $\mathcal{R}$ se compone de un conjunto de cuerpos rígidos conectados por varias articulaciones. Las articulaciones pueden ser simples como las rotacionales o las prismáticas o más complejas como las articulaciones esféricas o las articulaciones de rótula. La diferencia entre los dos tipos de articulaciones es que en las simples la articulación solo se mueve respecto a un grado de libertad: el ángulo de rotación para las articulaciones rotacionales o la cantidad de desplazamiento para las articulaciones prismáticas. En cambio las articulaciones esféricas o de rótula tienen dos grados de libertad. En este capítulo, asumiremos que tenemos articulaciones con un solo grado de libertad.

Una vez que hemos establecido que cada articulación tiene solo un grado de libertad, se puede describir la acción de cada una de ellas mediante un número real: el ángulo de rotación o el desplazamiento. El objetivo del análisis de la cinemática directa es determinar el efecto *acumulado* de las articulaciones variables. Vamos a explicar a continuación algunas convenciones que proveen un procedimiento sistemático para la ejecución de este análisis. Sin embargo, debemos estar consientes que el análisis de la cinemática directa de un manipulador con n-articulaciones se puede convertir en un problema sumamente complejo.

Un robot manipulador con n articulaciones tiene $n+1$ cuerpos rígidos ya que cada articulación conecta dos de estos cuerpos rígidos. Se enumeran las articulaciones desde 1 hasta n y se enumeran los cuerpos rígidos desde 0 hasta n, comenzando con la base. Por convención, la articulación i se analiza con respecto al cuerpo $i-1$. Cuando la articulación i se cambia, el cuerpo i se mueve.

Vamos a asociar la *i-ésima* articulación con la variable de esa articulación denotada por q_i. En caso de una articulación rotacional, q_i es el ángulo θ_i de rotación y en el caso de una articulación prismática, q_i es la longitud de desplazamiento d_i.

Para realizar el análisis cinemático, se adjunta rigurosamente un sistema de coordenadas para cada cuerpo rígido. En particular, damos $o_i x_i y_i z_i$ para el *i-ésimo* cuerpo. Esto significa que, cualquiera que sea el movimiento que el robot ejecute, las coordenadas de cada punto en el cuerpo i son constantes cuando se expresa en la *i-ésima* coordenada sistema de referencia. Además, cuando la articulación i actúa, el cuerpo i y su sistema de referencia adjunto, $o_i x_i y_i z_i$, experimenta el movimiento resultante. El sistema de referencia $o_0 x_0 y_0 z_0$, que está unido a la base del robot, se conoce como el sistema de referencia inicial. La Figura A.1 ilustra la idea de la unión de los marcos de referencia de manera rígida a los cuerpos en el caso de un manipulador tipo codo.

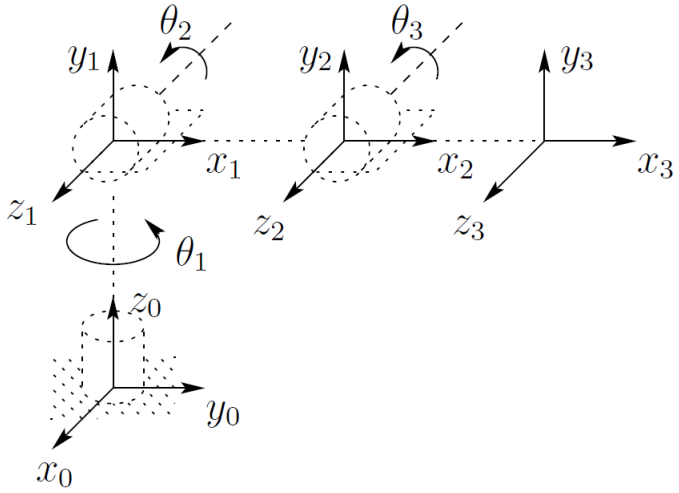


Figura A.1: Sistemas de coordenadas fijados a un robot manipulador tipo codo

Ahora supongamos que A_i es la matriz de transformación homogénea que expresa la posición y orientación de $o_i x_i y_i z_i$ con respecto a $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$. La matriz A_i no es

constante, sino que varía como la configuración del robot se cambia.

La matriz de transformaciones homogéneas que expresa la posición y la orientación de $o_j x_j y_j z_j$ con respecto a $o_i x_i y_i z_i$ se llama, por convención, **matriz de transformaciones**, y se denota por T_j^i. Esta matriz de transformaciones está dada por:

$$T_j^i = A_{i+1} A_{i+2} \cdots A_{j-1} A_j \text{ si } i < j$$
$$T_j^i = I \text{ si } i = j$$
$$T_j^i = (T_j^i)^{-1} \text{ si } j < i$$

Por la manera en que se han fijado los sistemas de referencia a los cuerpos rígidos correspondientes, se deduce que la posición de cualquier punto del efector final del robot, cuando se expresa el marco de referencia n, es una constante independiente de la configuración del robot. El vector O_n^0 denota la posición y orientación del efector final con respecto al sistema de referencia inicial y la matriz de rotación 3 x 3 R_n^0, con ello se puede definir la matriz de transformaciones homogéneas.

$$H = \begin{bmatrix} R_n^0 & O_n^0 \\ 0 & 1 \end{bmatrix}$$

Por lo tanto, la posición y orientación del efector final a partir del sistema de referencia inicial están dados por:

$$H = T_n^0 = A_1(q_1) \cdots A_n(q_n) \tag{A.1}$$

Y cada transformación homogénea A_i es de la forma:

$$A_i = \begin{bmatrix} R_i^{i-1} & O_i^{i-1} \\ 0 & 1 \end{bmatrix}$$

Por lo tanto,

$$T_j^i = A_{i+1} \cdots A_j = \begin{bmatrix} R_j^i & O_j^i \\ 0 & 1 \end{bmatrix}$$

En principio, la cinemática directa consiste en determinar las matrices $A_i(q_i)$, se multiplican entre sí, según sea necesario. Sin embargo, es posible alcanzar una cantidad considerable de simplificación mediante el uso de convenciones y formalismos, como el formalismo Denavit-Hartenberg, el cual se describe a continuación.

A.2. Formalismo Denavit-Hartenberg

El formalismo Denavit-Hartenberg o DH, es una convención comúnmente utilizado para la selección de sistemas de referencia en aplicaciones robóticas. fue propuesta en 1955 por Denavit y Hartenberg. Es una propuesta matricial que permite establecer de manera sistemática un sistema de coordenadas [6].

La representación de DH de un elemento rígido depende de cuatro parámetros geométricos asociados con cada elemento. Estos cuatro parámetros describen completamente cualquier articulación prismática o de rotacional y se definen de la siguiente forma:

- Rotación alrededor del eje z_{i-1} en un ángulo θ_i.
- Traslación a lo largo de z_{i-1} una distancia d_i.
- Traslación a lo largo de x_i una distancia a_i.
- Rotación alrededor del eje x_i en un ángulo α_i.

En esta convención, cada matriz de transformación homogénea A_i está representada

como un producto de cuatro transformaciones básicas como se presenta a continuación:

$$
\begin{aligned}
A &= R_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} R_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} c_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{A.2}
$$

Donde las cuatro variables θ_i, a_i, d_i, α_i son los parámetros asociados con cada cuerpo rígido i la articulación i.

A.2.1. Algoritmo de Denavit-Hartenberg

- **Paso 1**. Localizar y etiquetar los ejes de las articulaciones $z_0, \cdots, z_{n-1}$.

- **Paso 2**. Establecer el marco de referencia base. Ajustar el origen donde indique el eje z_0. Los ejes x_0 y y_0 son elegidos convenientemente para formar un sistema de referencia que siga la regla de la mano derecha. Para $i = 1, \cdots, N-1$, realizar los pasos 3 a 5.

- **Paso 3**. Localizar el origen O_i donde la normal común a z_i y z_{i-1} intersecta z_i. Si z_i intersecta a z_{i-1} se debe localizar O_i en esta intersección. Si z_i y z_{i-1} son paralelos, se debe localizar O_i en cualquier posición a lo largo de z_i.

- **Paso 4.** Establecer x_i a lo largo de la normal común entre z_{i-1} y z_i a través de O_i, o en la dirección normal al plano z_{i-1-z_i} y z_i intersecta.

- **Paso 5**. Establecer y_i para completar el sistema de referencia respecto a la regla de la mano derecha.

- **Paso 6**. Establecer el sistema de referencia del efector final o_n, x_n, y_n, z_n. Suponiendo que la enésima articulación es rotacional, establecer $z_n = a$ a lo largo de la dirección z_{n-1}. Establecer el origen O_n convenientemente a lo largo de z_n, preferentemente en el centro de la pinza del efector final o por lo la punta de cualquier herramienta que el manipulador puede estar llevando. Establecer $y_n = s$ en la dirección del cierre de la pinza y $x_n = \mathbf{n}$ como el producto cruz sxa. Si la herramienta no es un simple conjunto de pinzas, entonces establecer x_n e y_n convenientemente para formar un sistema de coordenadas que cumpla la regla de la mano derecha.

- **Paso 7**. Crear una tabla para los parámetros de cada cuerpo rígido del robot con los parámetros $a_i, d_i, \alpha_i, \theta_i$.

- **Paso 8**. Formar las matrices de transformaciones homogéneas A_i mediante la sustitución de los parámetros en la ecuación A.2.

- **Paso 9**. A partir de $T_n^0 = A_i \cdots A_n$. Esto da la posición y orientación del sistema de referencia de la pinza expresada en base al sistema de referencia inicial.

En la sección de resultados de este apéndice se mostrará la implementación de este algoritmo para hallar la cinemática inversa y directa del robot Puma 560.

A.3. Cinemática inversa y directa

La solución a la cinemática inversa y directa toma como forma de representación del robot el formalismo DH. Esta solución fue propuesta originalmente por C. Lee et al. [4]. Los sistemas de referencia con los que se construyeron las matrices de transformaciones homogéneas se tomaron del robot Puma 560 mostrado en la Figura A.2.

La tabla que resultó después de la aplicación del algoritmo de Denavit-Hartenberg se muestra en el Cuadro A.1.

De acuerdo a los parámetros mostrados en el Cuadro A.1, las matrices de transformaciones para los sistemas de coordenadas del Puma 560 se muestran a continuación:

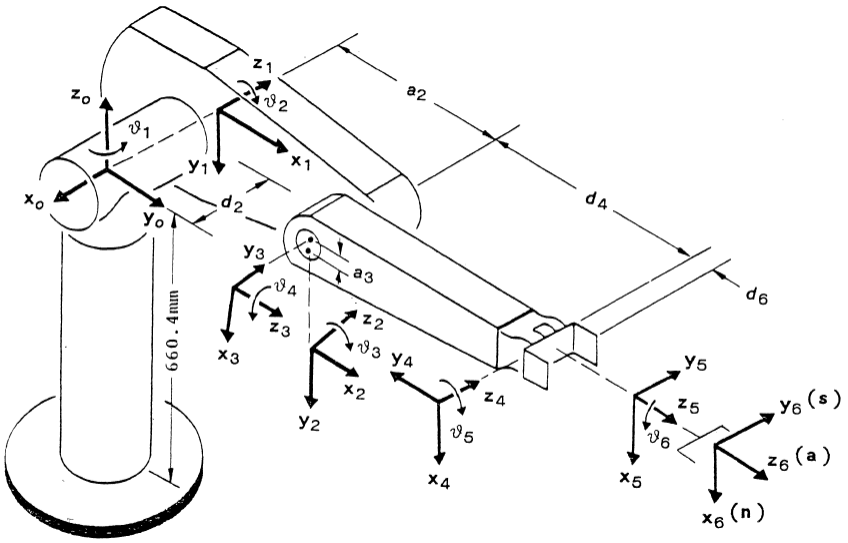


Figura A.2: Sistemas de referencia del Puma 560

Articulación	θ_i	α_i	a_i	d_i
1	90	-90	0.0mm	0.0mm
2	0	0	3.5mm	1.8mm
3	90	90	-0.01mm	0.0mm
4	0	-90	0.0mm	3.5mm
5	0	90	0.0mm	0.0mm
6	0	0	0.0mm	1.0mm

Cuadro A.1: Parámetros para sistema de coordenadas para el Puma 560

$$
\begin{aligned}
A_0^1 &= \begin{bmatrix} c_{\theta_1} & 0 & -s_{\theta_1} & 0 \\ s_{\theta_1} & 0 & c_{\theta_1} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_1^2 &= \begin{bmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & a_2 c_{\theta_2} \\ s_{\theta_2} & c_{\theta_2} & 0 & a_2 s_{\theta_2} \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_2^3 &= \begin{bmatrix} c_{\theta_3} & 0 & s_{\theta_3} & a_3 c_{\theta_3} \\ s_{\theta_3} & 0 & -c_{\theta_3} & a_3 s_{\theta_3} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_3^4 &= \begin{bmatrix} c_{\theta_4} & 0 & -s_{\theta_4} & 0 \\ s_{\theta_4} & 0 & c_{\theta_4} & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_4^5 &= \begin{bmatrix} c_{\theta_5} & 0 & s_{\theta_5} & 0 \\ s_{\theta_5} & 0 & -c_{\theta_5} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_5^6 &= \begin{bmatrix} c_{\theta_6} & -s_{\theta_6} & 0 & 0 \\ s_{\theta_6} & c_{\theta_6} & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{A.3}
$$

Donde $c_{\theta_i} \equiv cos\theta_i$ y $s_{\theta_i} \equiv sen\theta_i$.

A.3.1. Cinemática Directa

Una vez que se han presentado las matrices de transformaciones para el Puma 560, estamos en condiciones de expresar la solución para hallar el modelo cinemático directo. Las ecuaciones X, Y, X representan la posición del efector final dados los parámetros θ_i, con $i = 1 \cdots 6$.

$$\begin{aligned} p_x &= c_{\theta_1}[d_6(c_{\theta_{23}}c_{\theta_4}s_{\theta_5} + s_{\theta_{23}}c_{\theta_5}) + s_{\theta_{23}}d_4 + a_3c_{\theta_{23}} + a_2c_{\theta_2}] - s_{\theta_1}(d_6s_{\theta_4}s_{\theta_5} + d_2) \\ p_y &= s_{\theta_1}[d_6(c_{\theta_{23}}c_{\theta_4}s_{\theta_5} + s_{\theta_{23}}c_{\theta_5}) + s_{\theta_{23}}d_4 + a_3c_{\theta_{23}} + a_2c_{\theta_2}] - c_{\theta_1}(d_6s_{\theta_4}s_{\theta_5} + d_2) \\ p_z &= d_6(c_{\theta_{23}}c_{\theta_5} - s_{\theta_{23}}c_{\theta_4}s_{\theta_5}) + c_{\theta_{23}}d_4 - a_3s_{\theta_{23}} - a_2s_{\theta_2} \end{aligned} \tag{A.4}$$

A.3.2. Cinemática Inversa

El problema de cinemática directa contrasta con el problema de la cinemática inversa, que se ocupa de determinar los valores de las variables de las articulaciones que permitan alcanzar un posición y orientación deseada para el efector final del robot. En nuestro caso es imperante la solución exacta de este problema, ya que las estrategias de exploración parten de la idea de generar configuraciones (posiciones) para el sensor, que se coloca en el efector final del robot manipulador.

A continuación se presenta la solución de la cinemática inversa de las primeras tres articulaciones, ya que con ellas es posible colocar el efector final del Puma 560 en la posición deseada. Esto debido a que las últimas articulaciones están relacionadas con la orientación de la pinza del efector final y por el momento no es necesaria su solución.

Solución para la primer articulación . Se proyecta el vector de la posición $\vec{p}$ en el plano $x_0 - y_0$ y obtenemos la solución de la articulación 1:

$$\theta_1 = atan2\left[\frac{p_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_xd_2}{p_x\sqrt{p_x^2 + p_y^2 - d_2^2} - p_yd_2}\right] \tag{A.5}$$

Solución para la segunda articulación . Para encontrar la solución a la articulación 2, proyectemos el vector de la posición $\vec{p}$ al plano $x_1 - y_1$. La solución está dada por:

$$\theta_2 = atan2\left[\frac{sen(\theta_2)}{cos(\theta_2}\right] \tag{A.6}$$

Donde:

$$sin(\theta_2) = sin(\alpha + \beta) \tag{A.7}$$

$$cos(\theta_2) = cos(\alpha + \beta) \tag{A.8}$$

Y los valores de cos(α), sen(α), cos(β) y sen(β) están dados por:

$$sin(\alpha) = \frac{p_z}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{A.9}$$

$$cos(\alpha) = \frac{\sqrt{p_x^2 + p_y^2 - d_2^2}}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{A.10}$$

$$sin(\beta) = \sqrt{1 - cos^2\beta} \tag{A.11}$$

$$cos(\beta) = \frac{p_x^2 + p_y^2 + p_z^2 + a_2^2 - d_2^2 - (d_4^2 + a_3^2)}{2a_2\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{A.12}$$

Solución para la tercer articulación . Finalmente, para encontrar la solución a la articulación 3, proyectemos el vector de la posición $\vec{p}$ al plano $x_2 - y_2$. La solución está dada por:

$$\theta_3 = atan2\left[\frac{sen(\theta_3)}{cos(\theta_3)}\right] \tag{A.13}$$

Donde:

$$sin(\theta_3) = sin(\phi + \beta) \tag{A.14}$$

$$cos(\theta_3) = cos(\phi + \beta) \tag{A.15}$$

Y para solucionar las ecuaciones anteriores es necesario obtener los valores de cos(ϕ), sen(ϕ), cos(β) y sen(β), que están dados por:

$$sin(\phi) = \sqrt{1 - cos^2\phi} \tag{A.16}$$

$$cos(\phi) = \frac{a_2^2 + (d_4^2 + a_3 2) - R^2}{2a_2\sqrt{d_4^2 + a_3^2}} \tag{A.17}$$

Donde

$$R = \sqrt{p_x^2 + p_y^2 + p_z^2 + d_2^2} \tag{A.18}$$

$$sin(\beta) = \frac{d_4}{\sqrt{d_4^2 + a_3^2}} \tag{A.19}$$

$$cos(\beta) = \frac{a_3}{\sqrt{d_4^2 + a_3^2}} \tag{A.20}$$

A.4. Resultados

En esta sección se presentaran los resultados obtenidos al implementar el robot Puma 560 mediante un modelo de OpenGL. El prototipo que desarrollamos para probar la correcta implementación del modelo cinemático inverso y directo fue programada en C++ y empleando el entorno de desarrollo Qt Creator.

Esta herramienta permite establecer los valores permitidos para las 6 articulaciones (para nuestro robot las seis articulaciones son rotacionales). El robot, empleando las ecuaciones presentadas para la cinemática directa, calcula la posición del efector final

respecto al sistema de coordenadas inicial. El efecto del sistema es colocar una esfera de color amarillo en la posición calculada. En la Figura A.3 se muestra la primer prueba de la solución a la cinemática directa, en la cual los valores de los seis ángulos son: $\theta_1 = 40^o$, $\theta_2 = 30^o$,$\theta_3 = -10^o$, $\theta_4 = 10^o$, $\theta_5 = -30^o$, $\theta_6 = 20^o$. El resultado se observa en la Figura A.3.

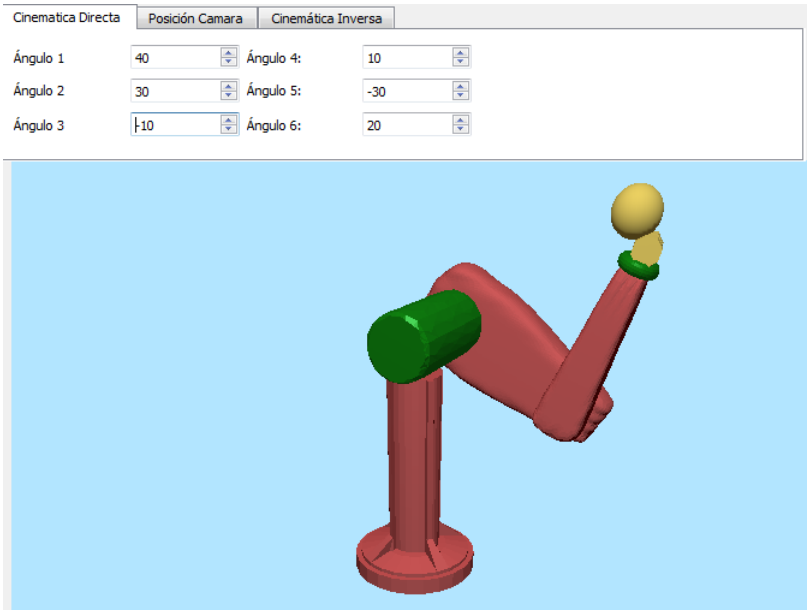


Figura A.3: Prueba de la solución de la cinemática directa del Puma 560

Para realizar las pruebas de la solución de la cinemática inversa, la herramienta se desarrolló con la capacidad de establecer el punto en el cual se desea que se coloque el efector final del robot. Se deben establecer las tres coordenadas x, y y z y, al igual que para la cinemática directa, la esfera amarilla indica la posición donde se ubica el efector final. Una vez establecidas las coordenadas, se calculan los valores de los ángulos (los primeros tres) para aproximar la posición del efector final a la posición deseada. En la Figura A.4 se muestra un ejemplo en el cual se desea que el efector final del robot se ubique en la posición $x = 3$, $y = 1$, $z = 5$.

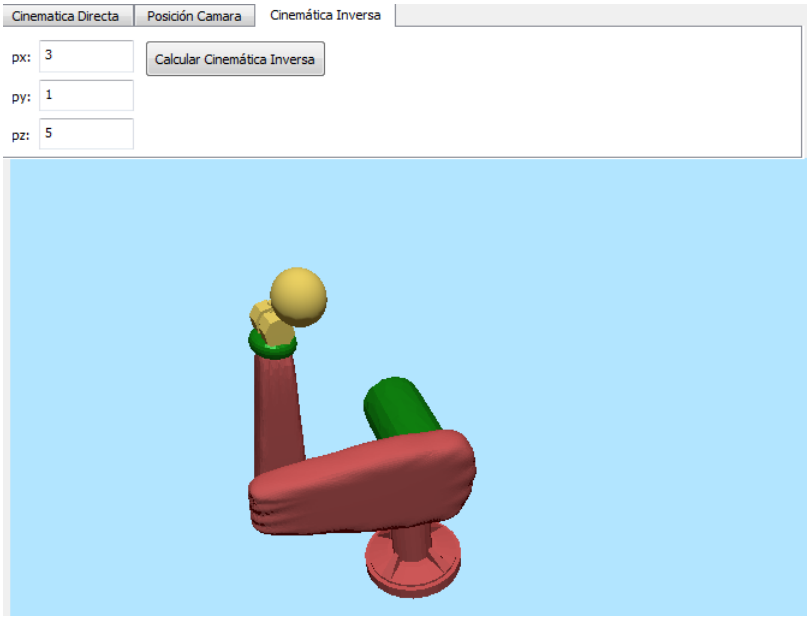


Figura A.4: Prueba de la solución de la cinemática directa del Puma 560

Apéndice B

Formulación del problema

En el contexto de la exploración, el robot *despierta* en un ambiente desconocido poblado de obstáculos. El robot tiene que explorar el ambiente y construir un modelo de ese ambiente. La exploración se realiza mediante el *cubrimiento* del mundo tanto como sea posible con las percepciones del sensor.

Comenzaremos formulación del problema de la exploración para sistemas robóticos generales propuesta por [9].

B.1. Formulación del robot y los modelos de los ambientes

El robot lo denotaremos por $\mathcal{R}$. Un robot consiste de una cadena cinemática de r cuerpos rígidos ($r \geq 1$) intercontectados por articulaciones elementales. Esta descripción incluye: a los manipuladores de base rígida, robots móviles tales como vehículos tipo trailer, robots manipuladores en dos dimensiones, humanoides y móviles manipuladores.

El ambiente $\mathcal{W}$ es un subconjunto compacto de $\mathbb{R}^N$, con $N = 2$ o $N = 3$. El ambiente representa el espacio físico en el cual se mueve el robot y de donde percibe. $\mathcal{W}$ contiene obstáculos estáticos $\mathcal{O}_j, j = 1, 2, \cdots, p$, donde cada obstáculo es un subconjunto compacto conectado de $\mathcal{W}$. La frontera $\partial\mathcal{W}$ se asume que actúa como un límite y es considerada como un obstáculo. La *región de obstáculos* $\mathcal{O}$ es la unión de $\partial\mathcal{W}$ y todos los obstáculos $\mathcal{O}_j, j = 1, 2, \cdots, p$. El *ambiente libre* está denotado por $\mathcal{W}_{libre} = \mathcal{W} \backslash \mathcal{O}$. El *espacio de configuraciones* del robot está denotado por $\mathcal{C}$ y un la configuración de un robot es denotada por q. Sea $\mathcal{R}(q)$ la región compacta de $\mathcal{W}$ ocupada por el robot en cierta configuración q. Se define la *región C-obstáculo* $\mathcal{CO}$ como el conjunto de con-

figuraciones q tales que $\mathcal{R}(q) \cap \mathcal{O} \neq \emptyset$. El *espacio de configuraciones libres* es $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{CO}$.

B.2. Formulación del modelo del sensor

El robot se equipa con un sistema de m sensores exteroceptivos, a continuación se presenta su formulación.

Asumiendo que el robot está en una configuración q, se denota por $\mathcal{F}_i(q) \subset \mathbb{R}^N$ a la región compacta que ocupa el campo de visión del i-ésimo sensor, el cual se asume que tiene forma de estrella con centro respecto al centro del sensor $s_i(q) \in \mathcal{W}$. Por ejemplo, en $\mathbb{R}^2$, $\mathcal{F}_i(q)$ puede ser un sector circular con centro $s_i(q)$, un ángulo abierto α_i y un radio R_i, donde el siguiente es el rango de percepción. El campo de sensado (total) en la configuración q. se define como la unión de todos los $\mathcal{F}_i(q)$, para $i = 1, 2, \cdots, m$.

Dada una configuración q, un punto $p \in \mathcal{W}$, se dice que es *visible desde el i-ésimo* sensor si $p \in \mathcal{F}_i(q)$ y el segmento abierto de linea que une a p con $s_i(q)$ no se intersecta con $\partial\mathcal{O} \cup \partial\mathcal{R}(q)$.

En cada configuración q, el sistema de sensado del robot devuelve:

- la *región libre visible* $\mathcal{V}(q)$, la cual contempla todos los puntos en $\mathcal{W}_{free}$ que son visibles desde al menos un sensor.

- la frontera de obstáculo visible $\mathcal{B}(q) = \partial\mathcal{O} \cup \partial\mathcal{V}(q)$, la cual contempla todos los puntos de $\partial\mathcal{O}$ que son visibles desde al menos un sensor.

B.3. Formulación de la tarea de exploración robótica

El robot explora el ambiente a través de una secuencia de acciones de movimiento planeado a partir de una visión. Cada configuración donde se obtiene una visión es llamada *una configuración de visión*. Sea q^0 la configuración inicial del robot y $q^1, q^2, \cdots, q^k$ la secuencia de configuraciones de visión ejecutadas por el robot en el

k-ésimo paso de la exploración. Cuando la exploración comienza, todo el conocimiento endógeno del robot se puede expresar como:

$$\mathcal{E}^0 = \mathcal{R}(q^0) \cup \mathcal{V}(q^0)$$

Donde $\mathcal{R}(q^0)$ representa el volumen libre que el cuerpo del robot puede ocupar y $\mathcal{V}(q^0)$ es la visión en la configuración q^0. En el paso $k \geq 1$. la *región explorada* es:

$$\mathcal{E}^k = \mathcal{V}(q^k) \cup \mathcal{E}^{k-1}$$

Ya que un requisito evidente es que $\mathcal{R}(q^k) \subset \mathcal{E}^{k-1}$ para cualquier k, tenemos que:

$$\mathcal{E}^k = \mathcal{R}(q^0) \cup \left(\bigcup_{i=0}^{k} \mathcal{V}(q^i) \right)$$

Se define como un punto *explorado en el paso k* a un punto $p \in \mathcal{W}_{free}$ si es contenido en $\mathcal{E}^k$ y en caso contrario se considerara *no explorado*. En cada paso k, la estimación actual del ambiente libre se denota por:$\mathcal{E}^k \subseteq \mathcal{W}_{free}$.

Una configuración q es *segura en el paso k* si $\mathcal{R}(q) \subset \mathcal{E}^k$. La *región segura* S^k contempla todas las configuraciones que son seguras en el paso k. Notemos que $S^k \subseteq \mathcal{C}_{free}$ representa una imagen del espacio de configuraciones de $\mathcal{E}^k$ que también representa el espacio estimado actual de $\mathcal{C}_{free}$. Una ruta en $\mathcal{C}$ es *segura en el paso k* si está completamente contenida en $\mathcal{S}^k$. El objetivo de la tarea de exploración es expandir $\mathcal{E}^k$ tanto como sea posible que k se incremente.