Estrategias de Planificación de Movimientos para Encontrar un Objeto en Ambientes Tridimensionales con un Robot Manipulador Móvil

por

M.C. Judith Espinoza León

Benemérita Universidad Autónoma de Puebla (2006)

Sometida a revisión al Departamento de Ciencias de la Computación
en el cumplimiento parcial de los requisitos para obtener el grado de

Doctor en Ciencias con Orientación en Computación

en el

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS A.C.,

Agosto 2012

Firma del autor .....
Departamento de Ciencias de la Computación
Agosto de 2012

Certificado por .....
Dr. Rafael E. Murrieta Cid
Grupo de Robótica, Investigador Titular B
Director de Tesis

Aceptado por .....
Dr. Rogelio Hasimito Beltran
Coordinador del Postgrado en Ciencias de la Computación

Estrategias de Planificación de Movimientos para Encontrar un Objeto en Ambientes Tridimensionales con un Robot Manipulador Móvil

por

M.C. Judith Espinoza León

Sometida a revisión al Departamento de Ciencias de la Computación
en Agosto de 2012, en el cumplimiento parcial de los
requisitos para obtener el grado de
Doctor en Ciencias con Orientación en Computación

1

Estrategias de Planificación de Movimientos para Encontrar un Objeto en Ambientes Tridimensionales con un Robot Manipulador Móvil

por

M.C. Judith Espinoza León

Sometida a revisión al Departamento de Ciencias de la Computación
en Agosto de 2012, en el cumplimiento parcial de los
requisitos para obtener el grado de
Doctor en Ciencias con Orientación en Computación

Resumen

En el presente trabajo de tesis se aborda el problema de minimizar el valor esperado del tiempo para encontrar un objeto en un ambiente tridimensional. Nosotros consideramos un robot manipulador móvil con siete grados de libertad equipado con un sensor en el efector final "eye in hand". También, suponemos que el robot conoce el ambiente. En particular, nos enfocamos al problema de la búsqueda de un objeto cuya ubicación en el ambiente es desconocida por el robot, pero la cual está modelada por una función de distribución de probabilidad, que se supone conocida.

Proponemos una formulación discreta del problema, utilizamos una división del ambiente usando visibilidad. Presentamos una cobertura convexa del espacio libre y utilizamos muestreo aleatorio para estimar la forma y tamaño de las regiones convexas en ambientes tridimensionales. También, utilizamos las regiones convexas para dirigir el proceso de búsqueda y procuramos encontrar caminos que mantengan un balance entre el movimiento de la base y el movimiento brazo del robot. Proponemos un método práctico para aproximar las regiones de visibilidad en tres dimensiones para un robot equipado con un sensor limitado en campo de vista y rango. La calidad de los caminos generados depende significativamente de las capacidades de sensado del robot. En este trabajo, generamos planes de búsqueda para un robot manipulador móvil equipado con un sensor omnidireccional y con un sensor limitado en campo de vista y rango mínimo y máximo y comparamos los resultados.

Presentamos un plan de búsqueda que consiste principalmente en un conjunto de regiones convexas, un orden para visitar las regiones y los caminos para que el robot viaje a cada región según el orden. Presentamos un método de reparación de planes de búsqueda cuando el ambiente sufre un cambio local, para aprovechar al máximo nuestra estrategía de búsqueda. Implementamos nuestros algoritmos y presentamos resultados en simulación en ambientes realistas.

Las publicaciones obtenidas en esta tesis hasta el momento son las siguientes [13], [15], [14] y [68].

Director de Tesis: Dr. Rafael E. Murrieta Cid
Título: Grupo de Robótica, Investigador Titular B

A mis amados Ángeles, que con su amor impulsan mi vida personal y profesional

Agradecimientos

Gracias a ti Señor, por quererme tanto y estar siempre presente en mi vida, por hacerme las cosas más difíciles para sentir una enorme felicidad al lograr mis metas.

Gracias Dr. Rafael Murrieta Cid, por aceptarme como tesista, por creer que podía desarrollar este trabajo y apoyarme en todo momento en la realización del mismo. Finalmente, gracias por su tiempo y paciencia.

Gracias a mi familia, por apoyarme en mis aspiraciones profesionales.

Gracias a mis amigos de mi ciudad natal, por ser fieles a la amistad aún en la distancia.

Gracias Gaby, por compartir tu vida a mi lado y ser mucho más que compañera y amiga, DEP.

Gracias a mis compañeros y amigos del CIMAT, que hicieron de mi estancia en el posgrado una experiencia memorable.

Gracias Angel, por todo lo vivido en Guanajuato durante mi estancia en CIMAT.

Por último, agradezco el apoyo brindado por el Consejo Nacional de Ciencia y Tecnología (CONACYT) para el inicio, desarrollo y culminación de este proyecto doctoral.

Índice general

Índice de figuras

Índice de Tablas

Lista de Algoritmos

Capítulo 1

Introducción

El presente proyecto de tesis se enfoca al problema general de desarrollar estrategias de movimiento para resolver una tarea que involucra visibilidad. En un entorno real, es difícil esperar que el robot conozca la ubicación de cada objeto, por lo que contar con un módulo de búsqueda de objetos se torna una necesidad crítica para cualquier robot, especialmente cuando se desconoce la ubicación del objeto. Para realizar una búsqueda el robot necesita moverse por el ambiente, explorarlo, pero hacerlo sin una metodología puede derivar en un proceso arduo y/o conllevar a un derroche de recursos, como tiempo y energía; peor aún producir resultados infructuosos, no encontrar el objeto. Si el número de objetos a buscar se incrementa seguramente el problema se tornaría computacionalmente desafiante, aún más si los objetos de interés son movidos, lo que comúnmente suele ocurrir en ambientes reales. Es importante que el robot cuente con un sistema de búsqueda de objetos, ya sea para realizar una tarea de búsqueda de objetos como tarea principal o como subtarea en un proyecto más integral.

1.1. Campo de Estudio y Motivación

El trabajo se ubica en el área planificación de movimientos para robots manipuladores móviles usando sensores. En este campo hay diferencias en como se usa el sensado. Por ejemplo, en el caso general de navegación [44, 1] donde la tarea es mover al robot desde una configuración inicial hasta una final, el sensado usualmente se emplea como herramienta para proveer información al robot. Nosotros estamos interesados en el tipo de tareas donde el objetivo final es sensar el ambiente [78, 25, 33], en este caso, el sensado se integra a la generación del plan como una restricción que debe satisfacerse al ejecutarse el plan de movimiento. Hay una serie de problemas

que yacen en esta categoría como la exploración, persecución-evasión y cobertura de terreno.

El proyecto de tesis está motivado por las múltiples aplicaciones que pueden darse a la búsqueda de objetos con robots móviles, desde encontrar partes en ambientes industriales o localizar obras de arte en un museo, hasta la localización de landmarks para tareas de navegación y estimación de pose o incluso para encontrar personas en brigadas de búsqueda/rescate. Es benéfico desarrollar nuevas estrategias de búsqueda de objetos ya que podrían usarse por sistemas automatizados o por buscadores humanos. Nuestra propuesta va más allá de una solución suficiente, como una estrategia de fuerza bruta que sería computacionalmente prohibitiva para situaciones fuera de lo trivial, en este trabajo tratamos de aportar soluciones eficientes, una estrategia que encuentre un objeto lo más rápido posible en promedio.

1.2. Antecedentes del Problema

Para llevar acabo la tarea principal de búsqueda necesitamos dotar al robot de la capacidad de moverse en un ambiente con obstáculos sin que colisione con ellos. La planificación de movimientos es la disciplina de la robótica que trata con tales problemas.

En planificación de movimientos para robots [37], el problema básico consiste en encontrar un camino libre de colisión para que el robot se mueva de una posición inicial a una posición final en un ambiente con obstáculos (navegación). Este problema también se conoce como el problema de la mudanza del piano [24]. Hay que llevar un piano de un lugar a otro en un ambiente conocido. El problema consiste en encontrar un camino libre de colisión, en un ambiente en tres dimensiones con obstáculos, el requerimiento básico es que el piano no choque con los obstáculos durante su movimiento.

En robótica, típicamente, el cálculo de caminos libres de colisión no se realiza en el espacio de trabajo del robot (donde la dimensión máxima es 3), sino en el espacio de configuraciones[1], donde el robot es caracterizado por las variables que determinan su posición y orientación. El espacio de configuraciones tiene la ventaja de que cualquier robot, no importando su forma o número de grados de libertad, puede ser representado como un punto. Esta formulación permite definir de manera precisa el problema, sin importar la forma geométrica del robot. Dicha formulación, fue propuesta por Tomas Lozano Pérez [46], y ha sido la manera clásica de establecer el problema por casi 30 años. Sin embargo, la complejidad consiste en calcular

[1] Ver apéndice A.

de manera exacta dicho espacio de configuraciones, que en general es de altas dimensiones y cuya caracterización topológica no es siempre euclidiana. Este problema ha sido abordado desde finales de la década de los setentas por investigadores en ciencias computacionales y matemáticas discretas. Los primeros métodos propuestos fueron determinísticos. El mejor planificador determinístico conocido hasta la fecha, fue desarrollado por John Canny en el MIT, es el llamado método silueta [8]. Dicho método encuentra un mapa de caminos en un espacio de configuraciones de altas dimensiones. Desafortunadamente, este planificador tiene complejidad exponencial en función de los grados de libertad del robot, por lo cual no es práctico en robots con más de tres grados de libertad.

Dada esta limitación, desde finales de la década de los ochentas, se han propuesto planificadores probabilísticos [37, 31, 42, 39, 10]. Este tipo de planificadores abandonan la idea de calcular de manera exacta el espacio de configuraciones y concentran sus esfuerzos en explorarlo a través de muestreo, a fin de capturar la topología del mismo por medio de redes de caminos. Entre los planificadores probabilísticos más exitosos podemos mencionar el *planificador probabilístico de mapas de caminos, PRM* [31] (propuesto por el grupo de Jean Claude Latombe en Stanford) y los *árboles aleatorios de exploración rápida, RRT* (propuesto por el grupo de Steven LaValle de la Universidad de Illinois), los cuales, además, permiten incorporar restricciones cinemáticas y dinámicas en las soluciones generadas [42]. Los PRMs y RRTs han mostrado ser muy eficientes para establecer conectividad en espacios de configuraciones de altas dimensiones y geométricamente complejos, solucionando así de manera eficiente, el problema de establecer una ruta libre de colisión para llevar a un robot de una configuración a otra. Sin embargo, dichos planificadores también tienen limitaciones. No son algoritmos completos[2] en el sentido determinístico, es decir no son capaces de establecer para un problema dado si existe una solución o no. Además, su desempeño se degrada significativamente si existen pasajes estrechos en el espacio de configuraciones [63]. Otras limitaciones importantes son que estas técnicas típicamente no toman en cuenta información proveniente de los sensores del robot, dado que suponen ambientes conocidos sin incertidumbre y además suponen la posición de la meta que el robot debe alcanzar como conocida.

1.2.1. Persecución - Evasión

Persecución-Evasión puede definirse como un escenario donde uno o más perseguidores (buscadores) se mueven en un ambiente para establecer o mantener visibilidad de uno o varios

[2] Ver apéndice A.

evasores (blancos). Existen diferentes variantes del escenario general que originan distintos problemas. Tres de esas variantes son:

1. El perseguidor tiene como meta encontrar a los evasores [76, 43, 21, 41, 83, 30].
2. El perseguidor desea mantener visibilidad sobre los evasores [40, 53, 51, 55, 52].
3. El perseguidor tiene como objetivo atrapar a los evasores [28, 27], esto es, el perseguidor tiene que moverse a una configuración en contacto con los evasores o más cerca que una distancia dada.

En general el problema de persecución-evasión típicamente se clasifica en dos categorías: búsqueda de blancos y mantener visibilidad sobre los blancos.

1. *Búsqueda de blancos:* es la tarea de encontrar un objeto estático o móvil con uno o varios perseguidores que cubren el ambiente. En el caso de un blanco móvil se debe tener especial cuidado en el diseño de los algoritmos para que el blanco no regrese a una región del espacio que ya ha sido explorada (a este problema se le conoce como *recontaminación*) [21]. Algoritmos determinísticos [59, 76, 20, 62] y probabilísticos [29, 23, 83] han sido propuestos para resolver este problema.
2. *Seguimiento de blancos:* se trata de mantener en el campo de vista (región de visibilidad) del robot a los evasores móviles que tratan de escapar de esta región, por ejemplo escondiéndose detrás de algún obstáculo [40, 53].

Para los problemas que se definieron previamente, hay características que pueden ser agregadas para hacerlos más generales. Por ejemplo, restricciones cinemáticas o dinámicas al movimiento de los robots [3], incertidumbre en la posición de los robots [54], sensores limitados [79, 55], etc. Las estrategias de movimiento que consideran múltiples robots [70] son generalmente más difíciles de obtener, pero usualmente producen mejor desempeño (comparadas con las generadas para un solo robot). En todos estos problemas, es bastante interesante, pero frecuentemente difícil, desarrollar algoritmos completos que encuentren soluciones óptimas (bajo algún criterio) [53, 3].

A continuación describimos algunos trabajos que consideramos particularmente relevantes sobre búsqueda de objetos.

1.2.2. Búsqueda de blancos

Se han obtenido resultados interesantes en la problemática de encontrar un blanco en un grafo, donde el perseguidor y el evasor pueden moverse de vértice en vértice hasta que el perseguidor garantiza tocar al evasor [49, 59]. Un aspecto importante es determinar el número mínimo de perseguidores que se necesitan para solucionar el problema, este rubro está muy relacionado con otras características del grafo tal como la anchura de corte [48, 50]. También se ha demostrado que ciertos grafos se pueden recorrer monotónicamente, es decir, sin volver a visitar vértices múltiples veces [4, 35].

El problema mencionado arriba fue introducido por Suzuki y Yamashita [76], quienes estaban interesados en la existencia y la complejidad de un algoritmo para un polígono simple P con n segmentos. El objetivo era decidir si hay una solución para un solo perseguidor. De existir una solución el algoritmo debía proponer un plan de búsqueda.

De forma independiente, Icking y Klein [26] definieron el problema del recorrido para dos guardias. Es un problema de búsqueda para dos guardias cuya posición inicial y final son conocidas, y se mueven en un polígono de modo que siempre sean mutuamente visibles. Icking y Klein dieron una solución con un orden de complejidad de $O(n \log n)$, que fue mejorada más adelante por Heffernan [22] por el óptimo $O(n)$. Tseng et al. en [80] ampliaron el problema del recorrido de dos guardias relajando la restricción de que la posición inicial y la final sean conocidas. Ellos presentaron un algoritmo $O(n \log n)$ que decide si un polígono puede ser recorrido por dos guardias, y un algoritmo que da como resultado todas las posiciones iniciales y finales posibles que permiten la búsqueda para esos dos guardias. La problemática de persecución-evasión es frecuentemente formalizada como un problema de teoría de juegos dinámicos no cooperativos [27, 2].

1.2.3. Problema de la galería de arte

En 1973, Klee elaboró esta simple pregunta: ¿Cuántos guardias/cámaras son necesarios para vigilar una galería de arte?. Supone que la galería está modelada por un polígono y el guardia/cámara puede ver infinitamente lejos y en todas direcciones. Es decir, el problema consiste en determinar el número mínimo y la ubicación de los guardias de forma que colectivamente vean toda la galería. Este problema es conocido como el *Problema de la galería*

de arte. Desde entonces esta pregunta ha recibido una gran atención y se han estudiado diversas variantes, como se muestra en [57] y en las revisiones del estado del arte en [74, 82]. [45] prueba que este problema es NP-duro. Diversas variantes incluyen guardias móviles, campo de vista limitado, polígonos rectilíneos e iluminación de conjuntos convexos. Los resultados son tan numerosos y la mayoría son más combinatorios que geométricos. Usando un mapa poligonal bi-dimensional del espacio de trabajo del robot, en [17] se plantea una estrategia de muestreo para encontrar una solución aproximada al problema de la galería de arte; adicionalmente toma en cuenta las limitaciones físicas del sensor, satisfaciendo las restricciones de incidencia y de rango.

Otro problema relacionado es el *Problema del vigilante* que implica calcular un camino continuo de longitud mínima, que al recorrerlo el vigilante pueda ver todo el ambiente. En este problema, lo importante es la longitud del camino -en contraste con el problema de la galería de arte, donde la distancia entre los puntos es irrelevante. El problema del vigilante también es NP-duro cuando el polígono que representa el ambiente no está simplemente conectado como se muestra en [9]. En [74] también se encuentra un compendio de propuestas para solucionar este problema.

Es importante subrayar la relación entre el problema de búsqueda de objetos y el problema de la galería de arte. En el segundo problema la meta es determinar el mínimo número de guardias y sus ubicaciones en el ambiente (galería de arte) tal que todos los puntos del espacio de trabajo sean visibles por lo menos por un guardia. En la búsqueda de objetos la meta es encontrar un agente estático o móvil usando un guardia móvil (también puede verse como una instancia de persecución-evasión). Con respecto al problema del vigilante, el objetivo es encontrar un camino con el cual el guardia pueda ver todo el ambiente, esto garantiza que eventualmente al recorrerlo encontará al objeto, pero no siempre el camino de longitud mínima conlleva a minimizar el tiempo promedio de encontrar un objeto.

Nosotros nos enfocaremos a la variante de persecución-evasión que corresponde a la búsqueda de objetos. Es importante hacer notar que un componente básico en la solución del problema es generar algoritmos de navegación que permitan mover al robot de una configuración a otra. Por lo que también propondremos técnicas de navegación basadas en el sensado adaptadas particularmente a la problemática de búsqueda de objetos.

1.2.4. Estrategias de búsqueda recientes

El problema de búsqueda de objetos también ha sido atacado como una tarea de "búsqueda visual activa", es un tipo de tarea perceptual que requiere del proceso de "atención visual". Esta última es una capacidad presente en animales y seres humanos que permite reducir la cantidad de información que llega a través de los ojos, limitando las regiones del campo visual a analizar a aquellas que resultan más interesantes. En otras palabras, la búsqueda visual involucra un sensado activo del ambiente para discernir objetos o una característica en particular (el objetivo) entre otros objetos o características (los distractores). Se puede tener un mejor panorama sobre esta vertiente en [85, 81, 75].

Lau et al. [38] proponen un algoritmo para robots buscadores de múltiples objetivos en un ambiente conocido. El ambiente es dividido en un conjunto de regiones, se utiliza una matriz de adyacencia para describir la conexión existente entre ellas. El algoritmo se basa en la información disponible de los objetivos: el número de objetivos esperados en una región dada es proporcional a su tamaño. Esta información se actualiza después de cada operación de búsqueda. La tarea es encontrar una secuencia de acciones que minimice el valor esperado del tiempo en encontrar los objetivos. La estrategia de búsqueda utiliza programación dinámica. El principal inconveniente de la propuesta es que al incrementarse el número de regiones, el problema se convierte en uno computacionalmente intratable.

Bourgault et al. [6] emplea un enfoque Bayesiano para coordinar múltiples sistemas de sensores autónomos que buscan un único objetivo no antagónico. La función de densidad de probabilidad del objetivo se utiliza como información a priori y se va actualizando con los datos del modelo del sensor y el movimiento esperado del objetivo. La trayectoria de búsqueda óptima se define como aquella que maximiza la probabilidad acumulada de detección en un período de tiempo limitado. Las suposiciones clave en su estrategia son: las funciones de densidad de probabilidad de las ubicaciones del objetivo son suaves y el espacio de búsqueda está libre de obstáculos que limiten los movimientos de los buscadores, por ejemplo, los vehículos de rescate aéreos.

Sarmiento et al. en [69, 70, 73] aborda el problema de encontrar trayectorias óptimas para la búsqueda de un objeto en un ambiente bidimensional conocido, de tal forma que se minimice el valor esperado del tiempo en encontrar el objeto. Proponen una solución en dos niveles, en el primer nivel usan una descomposición del espacio de trabajo en curvas críticas para establecer una estructura cualitativa sobre la trayectoria solución. En el segundo nivel, se refinan segmentos

individuales de la trayectoria usando optimización numérica local. Después en [72] introducen un esquema basado en muestreo para generar un conjunto inicial de configuraciones de sensado para el robot. Proponen un algoritmo de cobertura convexa usando este muestreo. Utilizan esta descomposición del ambiente para generar un grafo que captura la conectividad del espacio. Finalmente, hacen una búsqueda en el grafo para generar trayectorias que tratan de minimizar el tiempo promedio en encontrar el objeto.

1.3. Contribuciones

Esta tesis presenta una extensión sobre los trabajos en [69, 71, 73, 72]. Las soluciones propuestas se muestran a través de un simulador desarrollado en lenguaje C++ y OpenGL. Diseñamos ambientes de trabajo con cierto toque realista, como una oficina o un conjunto de cuartos que pueden pertenecer a una planta en un edificio o a un museo. Parte de la solución propuesta al problema de búsqueda deriva en un algoritmo rápido y eficiente de cobertura convexa que puede aplicarse a ambientes bidimensionales así como a tridimensionales.

Consideramos tres contribuciones importantes, primero, calcular la región de visibilidad exacta en escenas 3D es una tarea computacionalmente muy demandante, los métodos exactos conocidos actualmente no son aplicables directamente a escenas grandes debido a su complejidad computacional y a problemas de robustez numérica, [60, 12, 11, 5]. Nosotros proponemos un método práctico y rápido basado en muestreo para aproximar lo que el robot percibe del ambiente ya sea con un sensor omnidireccional o un sensor limitado, en otras palabras, proponemos un método que aproxima la región de visibilidad en ambientes 3D para dirigir el proceso de búsqueda.

Nuestra segunda contribución es la implementación de una estrategia de movimientos que selecciona un subgrupo de grados de libertad del robot para realizar un proceso de optimización sobre ellos, el resto de grados de libertad del sistema se consideran variables aleatorias. La optimización sobre un conjunto reducido de grados de libertad es crucial para mantener un balance entre una solución efectiva que reduzca el valor esperado del tiempo en encontrar un objeto y una solución que sea viable con respecto al consumo de tiempo de cómputo.

Una última aportación es una metodología para la reparación de un plan de búsqueda. Una vez que se ha generado un plan de búsqueda para un objeto en un ambiente 3D dado, puede

ocurrir un cambio en un espacio localizado del ambiente, lo que genera una modificación local del mismo. De acuerdo a nuestro método el plan global de búsqueda puede seguir utilizándose realizando también sólo una modificación local.

1.4. Organización de la tesis

En el siguiente capítulo expondremos el problema a tratar, así como un bosquejo de la solución propuesta y la notación utilizada. En el capítulo 3 describiremos brevemente un algoritmo eficiente para encontrar el orden de las configuraciones de sensado. En el capítulo 4 presentamos una estrategia basada en muestreo que descompone el ambiente tridimensional en una cobertura convexa, en otras palabras, obtenemos una representación del espacio libre del ambiente a través de un conjunto de regiones convexas. En el capítulo 5 describiremos las características del sistema robótico. Analizaremos cómo la métrica adoptada para medir el movimiento influye en las trayectorias encontradas para mover el robot en el ambiente. En el capítulo 6 detallaremos las estrategias adoptadas para la búsqueda de un objeto con un manipulador móvil equipado con un sensor limitado tanto en rango como en campo de vista. El capítulo 7, presenta la metodología para la reparación de planes globales de búsqueda cuando se presenta un cambio local en el ambiente. En el capítulo 8 presentamos los resultados obtenidos con nuestro simulador para diferentes ambientes. Finalmente, en el capítulo 9 presentamos nuestras conclusiones y un esbozo sobre futuras direcciones de investigación.

Capítulo 2

Planteamiento del problema y Notación

La meta principal de este trabajo de tesis es desarrollar estrategias de movimiento para solucionar el *problema de encontrar un objeto tan rápido como sea posible en promedio*. Nosotros definimos el problema de la manera siguiente: Dado un ambiente, que es completamente conocido por el robot, y un objeto del cual el robot desconoce su ubicación dentro del ambiente, el objetivo es desarrollar una estrategia de movimiento para que el robot sea capaz de encontrar el objeto en el menor tiempo posible en promedio. En otras palabras, si la búsqueda se realizará varias veces se minimizaría el tiempo promedio requerido para encontrarlo.

Veamos ahora los componentes principales del escenario donde se desarrolla el problema:

1. **Ambiente,** (W).- Trabajaremos con ambientes en el espacio tridimensional. Los ambientes son de tipo interior, generalmente se les denomina así a lugares que se encuentran dentro de edificios, se caracterizan por ser cerrados, con pisos planos con poca pendiente, la gran mayoría de las estructuras que los forman son del tipo geométrico y con ángulos generalmente rectos (por ejemplo las paredes, los marcos de las puertas, esquinas, cruces de pasillos, etc.). Se supone también que en los ambientes de interior es más fácil tener controlados aspectos del ambiente como puede ser la iluminación.

   El ambiente es modelado a través de poliedros, los cuales, generan restricciones de movimiento y visibilidad para el robot; siendo necesario encontrar trayectorias libres de colisión para mover al robot. Se supone que el ambiente es conocido por el robot, es decir,

el robot cuenta con una representación del ambiente, un mapa geométrico.

2. **Robot**.- El robot es un manipulador móvil con siete grados de libertad dotado con un sensor en el efector final.

3. **Sensor**.- El modelo del sensor utilizado para resolver el problema, es un sensor con capacidades limitadas, es decir, tiene un rango de alcance mínimo y un rango de alcance máximo. Así mismo, está limitado en su campo de visión. Es un sensor conocido como "Eye in hand", que puede ser una cámara o un sensor láser de rango (en ingles; laser range finder). Nosotros asumimos que el robot es capaz de reconocer el objeto cuando este se encuentra dentro de la región de visibilidad del sensor.

4. **Objeto**.- En la definición del problema se plantea la búsqueda de un objeto en general, es decir, que no se particulariza a un objeto en especial, por ejemplo, un cesto de basura, un teléfono o una lampara de techo, ya que puede sesgar la búsqueda, un cesto de basura generalmente lo encontramos en el piso, un teléfono sobre superficies horizontales y la lampara en las partes altas del ambiente. Nosotros queremos proponer una solución que posteriormente, con modificaciones menores, pueda ser adaptable a cualquier objeto. Se supone que la ubicación del objeto es desconocida por el robot buscador. Debido a que no existe alguna razón para creer que una ubicación del objeto es más probable que otra, se asignan valores equiprobables a cada ubicación. De hecho, esta suposición es conocida en la literatura y es denominada como el "Principio de razón insuficiente" [36].

   Creemos que la formulación propuesta en este trabajo puede utilizar funciones de densidad de probabilidad (f.d.p.) arbitrarias. En un escenario donde uno o más objetos son buscados varias veces, debería ser posible iniciar la búsqueda asumiendo una f.d.p. uniforme y modificarla de acuerdo a los lugares donde el objeto es encontrado. Es claro que esto requeriría un trabajo adicional, por lo que en lo subsiguiente solamente se desarrollarán soluciones específicas para el caso de una función de densidad de probabilidad uniforme.

2.1. Bosquejo del problema: caso continuo

Dado que suponemos que la ubicación exacta del objeto es desconocida, podemos aproximar una función de densidad de probabilidad de la ubicación del objeto en el ambiente, $f_{xyz}(x, y, z)$.

Por otra parte, establecemos que el robot inicia su movimiento en el tiempo $t = 0$ desde una posición dada y sigue una trayectoria S en el ambiente. Definimos al tiempo que le toma en encontrar el objeto, T, como el tiempo que transcurre desde que parte de la posición inicial hasta que el objeto entra por primera vez en el campo de vista del robot.

La probabilidad de ver el objeto antes de un tiempo dado t, $P(T \leq t)$, es la función de distribución acumulada (FDA) de la variable aleatoria T,

$$F_T(t) = \int_0^t f_T dt = P(T \leq t). \tag{2.1}$$

En particular, la probabilidad de ver el objeto en un momento dado depende de la ruta seguida, S, por lo que, podemos definir la FDA para cualquier ruta como,

$$F_T(t|S) = \int_{V(S,t_f)} f_{xyz}(x, y, z)\, dxdydz. \tag{2.2}$$

donde f_{xyz} es la función de densidad de probabilidad de la ubicación del objeto asociada con el punto (x, y, z) en el ambiente, y $V(S, t_f)$ es el subconjunto del ambiente que el robot vió, sensó continuamente, desde el tiempo $t = 0$, cuando inicia la búsqueda, hasta el tiempo $t = t_f$, instante donde encuentra el objeto. Decimos que en el tiempo t_f el robot *cubrió* la región $V(S, t_f)$, en general usamos el término *cubrir* para denotar que el robot ha sensado el ambiente.

En particular, cuando f_{xyz} es una función de densidad de probabilidad uniforme, la probabilidad de ver el objeto antes de un tiempo dado t es proporcional al volumen del ambiente que se ha visto,

$$\frac{V(S, t)}{Volumen\, Total} = P(T \leq t). \tag{2.3}$$

Para calcular el valor esperado del tiempo T en encontrar el objeto siguiendo una ruta S podemos utilizar F_T para obtener f_T,

$$E[T|S] = \int_0^\infty t \cdot f_{T|S}(t|S)\, dt. \tag{2.4}$$

Estamos interesados en encontrar la trayectoria S que minimiza el valor esperado del tiempo en encontrar un objeto $E[T|S]$. En otras palabras, queremos obtener el camino que, en promedio, haga que el robot encuentre el objeto tan rápido como sea posible. Pero este problema en el continuo es uno de optimización de dimensión infinita cuya solución globalmente óptima

aún es desconocida [73]. Como se mostrará más adelante aún en una versión discreta el problema es NP-duro. En los capítulos posteriores presentaremos algoritmos que buscan 1)minimizar el valor esperado del tiempo para encontrar el objeto pero que 2)también buscan reducir el tiempo de computo para generar la ruta, en general, tratamos de mantener un balance entre estas dos características.

2.2. Bosquejo del problema: caso discreto

Podemos obtener más de una versión del problema de búsqueda de objetos, estableciendo restricciones en el ambiente, en la información de entrada, en el comportamiento del robot y sus capacidades de sensado.

Si restringimos el problema al caso cuando el robot sólo puede sensar el ambiente en ubicaciones específicas, cambia la naturaleza del problema de continuo a discreto, obteniendo la información del ambiente por bloques. Esto introduce el problema de generar un conjunto de configuraciones de sensado "apropiado".

Vamos a denotar una región convexa en un ambiente tridimensional como C_r, donde r indica la etiqueta de la región. Note que en una región convexa cualquier punto $p\,(x, y, z)$ de su interior puede conectarse, a través de un segmento de línea recta que está completamente contenido en la región C_r, a cualquier otro punto $p'\,(x', y', z')$ en su interior. Denotamos el espacio libre de obstáculos del interior de W como $int(W)$. Existe un conjunto $C = \{C_{r_1}, C_{r_2}, \ldots C_{r_n}\}$ de regiones convexas, donde n es el número de regiones, en el cual el espacio libre puede dividirse, de tal forma que la unión de todas las regiones en C cubre todo $int(W)$, $\bigcup_{i=1-n} C_{r_i} = int(W)$.

Como mencionamos, el robot sensa el ambiente en configuraciones discretas q_l, a estas configuraciones también les llamamos *guardias*, por el problema de la galería de arte. Vamos a denotar $V(q_l)$ a la región de visibilidad del sensor asociada con la configuración q_l.

Describimos la ruta seguida por el robot como una secuencia de configuraciones de sensado q_{l_k}, que comienza con la configuración inicial del robot e incluye cada configuración sólo una vez. En la notación q_{l_k} el subindice l se refiere a la etiqueta (identificador) de la configuración mientras que k se refiere al orden que la configuración q_l toma en la ruta. Por ejemplo, supongamos dos configuraciones, q_l y q_m, sea q_l la configuración inicial y q_m una configuración

parte de la ruta, el orden de visita del robot inicia con q_{l_0} después se mueve a otra configuración de sensado, enseguida a otra y así sucesivamente, la configuración q_m se alcanza en la $k-ésima$ visita, por lo que la denotamos como q_{m_k}.

Para cualquier ruta R, definimos el tiempo de encontrar el objeto, T, como el tiempo que usa el robot para moverse de una configuración a otra, - en orden -, hasta encontrar el objeto, es decir, hasta que el objeto entra por primera vez en el campo de vista del sensor.

Nuestra meta es encontrar la ruta que minimiza el valor esperado del tiempo en encontrar el objeto,

$$E\left[T|R\right] = \sum_{j} t_j \cdot P\left(T = t_j\right) \tag{2.5}$$

donde, t_j es el tiempo que transcurre desde que el robot parte de la configuración inicial hasta que recorre j configuraciones de sensado según la ruta R.

$$P\left(T = t_j\right) = \frac{Volumen\left(V(q_{l\_j}) \setminus \cup_{i<j} V(q_{l\_i})\right)}{Volumen\left(int\left(W\right)\right)} \tag{2.6}$$

es la probabilidad de ver el objeto por primera vez desde la configuración q_{l_j}.

Cabe mencionar que nosotros nos enfocaremos al problema en el caso discreto. En [73] los autores proponen una solución aproximada en el continuo para un ambiente en dos dimensiones.

2.3. Valor esperado del tiempo contra el tiempo en el peor caso

Es importante notar la diferencia entre minimizar el valor esperado del tiempo y minimizar el tiempo que le tomaría al robot el encontrar el objeto en el peor caso.

Para minimizar el tiempo en el peor caso, el robot debería obtener el camino más corto que cubra completamente el ambiente. Comúnmente significa que ninguna porción del ambiente tiene prioridad sobre otra y el instante en el que se cubren nuevas porciones es indiferente.

Por otro lado, para minimizar el valor esperado del tiempo, el robot debe obtener la mayor probabilidad de ver el objeto tan rápido como sea posible. Para una FDP uniforme, lo anterior significa, cubrir las porciones más grandes del ambiente tan pronto como sea posible, incluso

si esto significa dedicar más tiempo después para completar el sensado de todo el ambiente. Para funciones de probabilidad distintas a la uniforme, el robot debería ver las secciones más prometedoras primero. Nosotros creemos que esto representa otro paradigma para la tarea de cobertura y búsqueda, donde es importante ganar tanta información nueva como sea posible en el más corto tiempo. Este planteamiento podría ser muy útil en aplicaciones donde el tiempo designado para realizar la tarea está limitado.

Las trayectorias que corresponden a minimizar el valor esperado del tiempo y la distancia Euclidiana que satisfacen los criterios anteriores no son las mismas. En realidad, para un ambiente dado, la ruta que minimiza la distancia viajada puede no minimizar el valor esperado del tiempo de encontrar el objeto al recorrerla.

Consideremos el ambiente de la figura 2.1. Es un ambiente en dos dimensiones integrado por dos cuartos de diferente tamaño y un corredor, el robot se considera como un punto (x, y) y se supone un sensor omnidireccional. Se considera que el objeto siempre está presente en alguno de los dos cuartos y la probabilidad de verlo está relacionada con el tamaño de los cuartos. Cada cuarto tiene una puerta y se puede apreciar su interior completamente desde el umbral. El robot inicia el recorrido en el corredor en la configuración q_0. El cuarto de la derecha es más pequeño, se puede ver su interior desde la configuración q_1, pero está más cercano a la configuración inicial. Mientras que el cuarto a la derecha es más grande, desde q_2 se puede ver su interior, pero está más lejos de la configuración inicial. Existen sólo dos rutas que el robot podría seguir para resolver el problema: Ir al cuarto más pequeño primero $R1 : (q_{0\_0} \rightarrow q_{1\_1} \rightarrow q_{2\_2})$ o primero ir al cuarto grande $R2 : (q_{0\_0} \rightarrow q_{2\_1} \rightarrow q_{1\_2})$. Ahora analicemos ambas rutas, asumimos que el robot se mueve una unidad de longitud por unidad de tiempo.

- **Ruta 1:** Si el robot primero va al cuarto más pequeño y después se mueve al cuarto grande, alcanza q_1 al tiempo $t_1 = 1$ y q_2 al tiempo $t_2 = 7$. El valor esperado del tiempo que toma siguiendo esta ruta es:

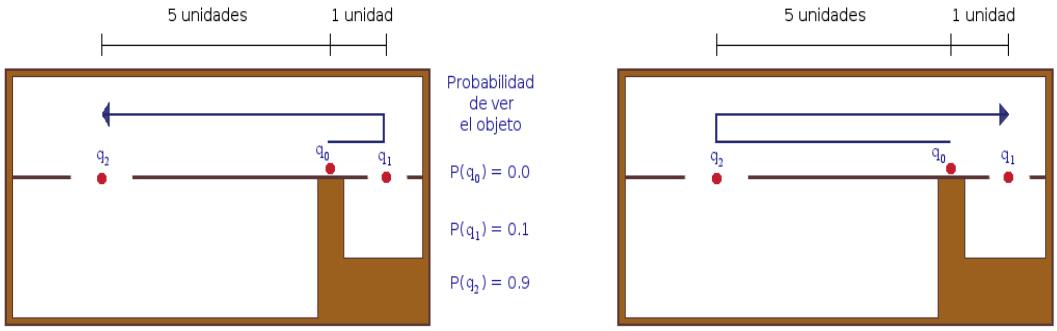
  $$E\left[T \mid (q_{0\_0}, q_{1\_1}, q_{2\_2})\right] = (0.1)(1) + (0.9)(7) = 6.4$$

  el robot completa la búsqueda después de 7 unidades de tiempo.

- **Ruta 2:** Si el robot primero va a la habitación grande y después se desplaza a la habitación pequeña, entonces alcanzará la configuración q_2 al tiempo $t_1 = 5$ y la configuración q_1 al tiempo $t_2 = 11$. El tiempo esperado en este caso es

  $$E[T \mid (q_{0\_0}, q_{2\_1}, q_{1\_2})] = (0.9)(5) + (0.1)(11) = 5.6$$

En el peor caso, le tomará al robot 11 unidades de tiempo encontrar el objeto.



(a) Ruta 1 (b) Ruta 2

Figura 2.1: Valor esperado contra peor caso

Un robot siguiendo la ruta 1 siempre termina la ruta después de 7 unidades, mientras un robot siguiendo la ruta 2 le toma 11 unidades. La ruta 1 minimiza la distancia recorrida. Sin embargo, el tiempo promedio que le toma al robot encontrar el objeto siguiendo la ruta 1 es 6.4 unidades, siendo mayor al tiempo esperado de la ruta 2 que es de 5.6 unidades. Por lo tanto, la ruta 2 minimiza el valor esperado del tiempo para encontrar un objeto.

En conclusión, una trayectoria que es óptima en distancia en el peor tiempo (al recorrerla completamente) no necesariamente minimiza el valor esperado del tiempo para encontrar el objeto deseado.

2.4. Planteamiento general de solución

Una vez que hemos definido el problema veamos un esquema general de nuestra propuesta de solución. Debido a la complejidad computacional de la tarea de búsqueda en ambientes 3-D, proponemos soluciones aproximadas basadas en el muestreo y heurísticas, con el objetivo principal de disminuir el valor esperado del tiempo en encontrar un objeto.

En [69] se mostró que el problema de encontrar un objeto en un ambiente poligonal en 2-D con un robot puntual es NP-Duro. Por tanto, decidimos continuar con la idea básica del uso de heurísticas. Se define una función de utilidad como la razón de una ganancia sobre un costo. Esta función es buena para dirigir un algoritmo voraz en un espacio de búsqueda reducido que es capaz de explorar varios pasos adelante sin incurrir en un alto costo computacional.

En general, el valor esperado del tiempo esencialmente depende de dos factores: 1) el valor de la variable aleatoria, cuantificada en alguna métrica y 2) la probabilidad asociada a la variable aleatoria. En nuestro problema de búsqueda, el valor esperado del tiempo depende de dos aspectos principales: 1) el **costo** de mover el objeto entre dos configuraciones, evaluado con alguna métrica, y 2) la probabilidad de ver el objeto desde la configuración de sensado, lo cual es equivalente a la **ganancia**.

Para nuestro robot manipulador móvil de siete grados de libertad usamos algoritmos conocidos en la literatura para calcular las trayectorias más cortas, con respecto a la métrica elegida, para moverse entre las distintas configuraciones que forman la ruta de búsqueda. Además, generamos trayectorias dependientes de la métrica, es decir, trayectorias que encuentran un compromiso entre mover la base o mover el brazo del robot. Tener diversos medios para cuantificar el costo de alcanzar una configuración, puede ser de utilidad, dependiendo en los requerimientos del usuario, para la minimización de energía, tiempo, desgaste mecánico, etc. Además, proponemos la coordinación de las traslaciones de la base y las rotaciones del brazo para ahorrar tiempo en alcanzar una configuración dada.

Para definir la probabilidad de ver el objeto en ambientes 3D, es necesario calcular la forma y tamaño de las regiones de visibilidad del sensor y sus intersecciones, pero tal cálculo es de muy alta complejidad computacional en un ambiente 3D poblado de obstáculos, [18]. Por esta razón, decidimos usar un enfoque basado en muestro para obtener de forma aproximada la región de visibilidad de cualquier punto. Los algoritmos propuestos no son determinísticos, así que proponemos límites probabilísticos en su desempeño. También utilizamos un algoritmo de cobertura convexa que permite dividir el ambiente en regiones convexas traslapantes. En base a esta descomposición proponemos estrategias de movimiento para el robot manipulador móvil.

Si el robot está equipado con un sensor omnidireccional, nuestro esquema de búsqueda es el siguiente: el robot siempre inicia en una región C_r particular perteneciente a C y visita las otras regiones a medida que el tiempo transcurre. El robot sigue las trayectorias más cortas según la métrica. Cabe notar que, la cobertura convexa proporciona flexibilidad de donde colocar el sensor para cubrir una región convexa, cualquier configuración q_i que coloque el sensor dentro de la región es una candidata válida para el sensor omnidireccional.

Para el caso de un sensor limitado tanto en rango como en campo de vista, primero el ambiente es divido en un conjunto de regiones convexas C, y en seguida cada región convexa

es cubierta con el poliedro que representa el campo de vista limitado del sensor, denotado por $\mathcal{F}$. Generamos un conjunto de configuraciones de sensado que cubran cada región convexa, cada conjunto de configuraciones es ligado a un grafo. Desarrollamos algoritmos que realizan búsquedas en grafos para así obtener una trayectoria de búsqueda en el ambiente para encontrar el objeto. El orden de visita para cada configuración es establecido por medio de nuestra función de utilidad.

En resumen, en el presente trabajo, proponemos una solución al problema global de búsqueda de un objeto, mediante tres procedimientos principales: 1) Aproximamos la región de visibilidad en 3D usando un método de muestreo sobre el espacio de trabajo. 2) Generamos un conjunto de configuraciones de sensado en el espacio de siete dimensiones del robot, para esto también empleamos muestreo. El conjunto de configuraciones tiene dos características: una baja cardinalidad y la distancia que separa a una configuración de otra es pequeña. Ambas propiedades son convenientes para disminuir el valor esperado del tiempo. 3) Determinamos el orden de visita de las configuraciones de sensado usando nuestra función de utilidad. Así encontramos una ruta que usa el robot para disminuir el tiempo promedio en encontrar dicho objeto

Capítulo 3

Selección de un orden para un conjunto de configuraciones

Como mencionamos anteriormente, nuestra estrategia global de búsqueda es descomponer el ambiente en regiones convexas y calcular una trayectoria para sensar todo el ambiente. Para recorrer cada región convexa necesitamos establecer un orden de visita para el conjunto de configuraciones de sensado que las cubra. A estas configuraciones les denominamos guardias, derivado del problema de la galería de arte. Entonces el problema se convierte en uno combinatorio y requerimos un algoritmo que encuentre un orden para que el robot visite cada uno de los guardias.

Para lo que resta del capítulo, no nos ocuparemos de la geometría del robot o sus capacidades de sensado. Por el momento, sólo vamos a considerar un robot puntual con un sensor omnidireccional y un rango infinito. Estas suposiciones ayudan a simplificar el análisis para proporcionar un mejor entendimiento del problema. Ya en los capítulos posteriores abordaremos el problema considerando un robot con geometría y capacidades de sensado limitadas.

Sarmiento et al. en [69, 73], proponen un algoritmo para obtener el orden óptimo de visita sobre un conjunto de ubicaciones de sensado. Este algoritmo encuentra la ruta que minimiza el valor esperado del tiempo en encontrar un objeto, suponiendo un ambiente en dos dimensiones, poligonal.

El algoritmo se apoya en una técnica de *ramificación y poda* cuyo nombre en español proviene de Branch and Bound en el idioma inglés. Esta técnica usualmente se aplica a problemas de

optimización para encontrar una solución óptima, especialmente en optimización discreta y combinatoria. El método fue propuesto por Land and Doig [34] para programación discreta.

3.1. Técnica de ramificación y poda

El método de ramificación y poda realiza una enumeración parcial del espacio de soluciones del problema basándose en la generación de un árbol de expansión [19], donde se quiere que grandes subconjuntos de soluciones candidatas no prometedoras sean descartadas en masa, usando cotas estimadas mínimas y máximas de la cantidad que se optimiza. El método en su versión más sencilla puede hacer un recorrido del árbol en anchura (estrategia LIFO), o en profundidad (estrategia FIFO), o utilizando el cálculo de funciones de costos para seleccionar el nodo que en principio parezca más prometedor a analizar (estrategia del mínimo costo o LC).

Además de estas estrategias, el método de ramificación y poda utiliza cotas para hacer que el podado de las ramas del árbol de expansión conduzcan a la solución óptima. Para ello se calcula en cada nodo una cota del posible valor de aquellas soluciones alcanzables desde ése nodo. Si la cota muestra que es peor que la mejor solución hallada hasta el momento no necesitamos seguir explorando por esa rama del árbol, lo que permite realizar el proceso de poda.

La expansión del árbol depende de la estrategia de búsqueda utilizada. Para determinar en cada momento cual nodo debe ser ramificado, se utilizan diversas estructuras de almacenamiento. Se emplea una pila para almacenar los nodos que se han generado pero no han sido examinados en una búsqueda en profundidad (LIFO). En una búsqueda en anchura se utiliza una cola (FIFO). Por otra parte, si la estrategia elegida es la de mímimo costo, la estructura necesaria es una cola de prioridad para almacenar los nodos ordenados por su costo. En esta última estrategia, la función de costos decide en cada momento que nodo debe explorarse, con la esperanza de alcanzar lo más rápidamente posible una solución más económica que la mejor encontrada hasta el momento.

En ramificación y poda los nodos del espacio de búsqueda se pueden etiquetar de tres formas distintas: nodo vivo, nodo muerto o en expansión.

- Nodo vivo, es un nodo factible y prometedor del que no se han generado todos sus hijos. Un nodo es prometedor cuando la información que tenemos indica que expandiéndolo podemos conseguir una solución mejor a la mejor solución en curso.

- Nodo muerto, es un nodo del que no van a generarse más hijos, ya sea por que se han generado todos sus hijos, o no es factible, o no es prometedor.
- Nodo en expansión es aquél del que se están generando hijos en ese instante.

Básicamente, en un algoritmo de ramificación y poda se realizan tres etapas, selección, ramificación y poda:

1. **La etapa de selección:** Se encarga de extraer un nodo entre el conjunto de nodos vivos que no han sido podados y que tienen la posibilidad de ser ramificados. La forma de elección depende directamente de la estrategia de búsqueda que se decida utilizar en el algoritmo.
2. **La etapa de ramificación:** Se construyen los posibles nodos hijos del nodo seleccionado en la etapa anterior, formando el árbol de expansión.
3. **La etapa de poda:** Se eliminan algunos de los nodos creados en el paso anterior, aquellos cuyo costo parcial sea mayor que la mejor cota mínima calculada hasta el momento se convierten en nodos muertos.

Aquellos nodos no podados pasan a formar parte del conjunto de nodos con posibilidad de ser ramificados, y se comienza de nuevo el proceso de selección. El algoritmo finaliza una vez encontrada la solución del problema o bien cuando se agota el conjunto de nodos vivos.

La contribución de esta técnica es la disminución, en lo posible, del espacio de búsqueda y por tanto de la atenuación de la complejidad en la exploración del árbol de posibilidades de la solución óptima. La dificultad de la técnica radica en encontrar una buena función de costo para el problema, en el sentido de que garantice la poda y que su cálculo no sea muy costoso.

3.2. Algoritmo para obtener un orden óptimo

Ahora presentamos el algoritmo propuesto en [69, 73], para encontrar el mejor orden sobre el conjunto de ubicaciones de sensado para minimizar el valor esperado del tiempo en encontrar un objeto. Notaremos como la técnica de ramificación y poda es sólo un componente de apoyo para la construcción del algoritmo.

Dado un conjunto de ubicaciones, se construye un grafo completamente conectado con pesos, como sigue:

1. El conjunto de nodos del grafo está integrado por los guardias, en otras palabras, se crea un nodo N_j en el grafo para cada ubicación L_j.
2. Para cada par de nodos N_j y N_k se agrega una arista al grafo, asociando un peso variable W_{jk}.
3. El peso W_{jk} es dinámico, es decir, depende de la ruta seguida por el robot antes de alcanzar N_j. Estos pesos son calculados progresivamente. W_{jk} correspondería al incremento en el valor esperado del tiempo que el robot consume por ir de L_j y L_k.

Lo que sigue, una vez definido el grafo, es encontrar un camino hamiltoniano de longitud del número de nodos, n, que parta desde el nodo N_{j_0} hasta el nodo N_{j_n} pasando por cada nodo solo una vez. Más especificamente, encontrar el camino hamiltoniano de mínimo costo que empieza en la ubicación L_{i_0} e incluye sólo una vez a cada una de las restantes ubicaciones de sensado.

Empleando la técnica de ramificación y poda, se elige la estrategia de recorrido por función de costos, por lo que se mantiene una cola de prioridad con los nodos ordenados por su costo acumulado. De esta manera, el próximo nodo a expandirse es el de menor costo hasta el momento. Al expandirse un nodo, solo los nodos que son adyacentes y que no están incluidos ya en el camino son considerados como hijos. El peso W_{jk} de un nodo hijo N_j desde su nodo padre N_k es,

$$W_{jk} = tiempo(N_k) \cdot probabilidad(L_k)$$

$$tiempo(N_k) = tiempo(N_j) + T(L_j, L_k)$$

donde $T(L_j, L_k)$ es el tiempo que le toma al robot para ir de la ubicación L_j a la ubicación L_k, y $tiempo(N_k)$ es el tiempo en alcanzar la ubicación L_k, desde el nodo raíz siguiendo la expansión del árbol. Entonces el costo acumulado para el nodo hijo es:

$$Costo(N_k) = Costo(N_j) + W_{jk}$$

teniendo el costo para el nodo raíz como $Costo(N_{i\_0}) = 0$.

Inicialmente, la cola de prioridad contiene el nodo raíz correspondiente a la ubicación inicial, L_{i_0}. Aplicando el algoritmo de ramificación y poda, expandimos el árbol de soluciones hasta encontrar el camino que contiene todas las ubicaciones. Cuando se encuentra una solución, los nodos que actualmente se encuentran en la cola aún continúan expandiéndose hasta,

1. encontrar una solución de mejor costo que la solución encontrada hasta el momento, en este caso el proceso continúa, o
2. el nodo de mejor costo es peor que la actual mejor solución, en este caso la actual solución es la buscada, es la solución óptima.

Este algoritmo encuentra la solución óptima, aquella que minimiza el valor esperado del tiempo en encontrar un objeto. Desafortunadamente su complejidad en espacio y tiempo no son de orden polinomial. En realidad, el problema por si mismo es intratable, más específicamente es NP-duro. A continuación se muestra este hecho.

Teorema: Encontrar la secuencia de configuraciones de sensado que minimiza el valor esperado del tiempo dado por la ecuación 2.5 es un problema NP-duro.

Prueba: La prueba consiste en reducir el problema del camino hamiltoniano de mínimo peso (del inglés minimum weight hamiltonian path problem, MWHP), el cual se ha mostrado que es NP-duro [16], a una instancia de nuestro problema.

Consideremos un conjunto de ubicaciones $L = \{L_i\}$, $i = 1, ..., n$, tal que sus regiones de visibilidad no se traslapan, es decir, $V(L_i) \cap V(L_j) = \emptyset$ para todo $i \neq j$ y la probabilidad de ver el objeto en cada ubicación de sensado es equivalente, $P(L_i) = \frac{1}{n}$. Ahora, supongamos que $G = (V, E)$ con $|V| = n$ es una instancia del MWHP. La correspondiente instancia de nuestro problema tiene al conjunto $\{L_i\}$ idéntico a V. Y para cualquiera dos ubicaciones de sensado $v_i, v_j \in L$ el peso para la arista asociada a E se define como, el tiempo para ir de L_i a L_j, como su equivalente en distancia.

Ya que hemos definido la probabilidad igual para cada ubicación, el valor esperado del tiempo en encontrar el objeto es determinado solamente por el tiempo que dispone para alcanzar las ubicaciones al recorrer el camino. Dado que el tiempo es proporcional a la distancia, el orden que minimiza el valor esperado del tiempo será exactamente el mismo que minimiza la distancia viajada.

En esencia, si existe un algoritmo en tiempo polinomial que resuelva nuestro problema entonces MWHP puede resolverse igualmente en tiempo polinomial. Y si MWHP es intratable nuestro problema también lo es. ∎

Con lo anterior hemos mostrado que nuestro problema es al menos tan difícil como el problema del camino hamiltoniano de mínimo peso. Por lo cual, proponemos el uso de heurísticas para encontrar soluciones aproximadas.

3.3. Heurística: Función de utilidad

Un algoritmo, de primera intuición, para obtener un buen orden en las ubicaciones de sensado, elegiría como siguiente ubicación a aquella que aporte un menor incremento al cálculo parcial de la función 2.5. Creemos que debido al producto en 2.5, este algoritmo producirá que se visiten primero ubicaciones con baja probabilidad de ver el objeto, lo cual es contrario a lo que se desea, encontrar el objeto lo más rápido posible.

Por esta razón se propone un algoritmo ávido dirigido por una función de utilidad. Esta función evalúa qué tan conveniente es ir a una ubicación desde alguna otra, y se define a continuación:

$$U(L_j, L_k) = \frac{P(L_k)}{Tiempo(L_j, L_k)} \tag{3.1}$$

La función nos indica que la utilidad de ir a la ubicación L_k estando el robot ubicado en L_j es proporcional a la probabilidad de encontrar el objeto e inversamente proporcional al tiempo dedicado a viajar. El algoritmo basado en esta función de utilidad proporcionará al robot la habilidad de determinar su siguiente destino, tendiendo a preferir ubicaciones cercanas y/o que tienen una alta probabilidad de ver el objeto.

La función de utilidad está directamente relacionada con el valor esperado del tiempo en 2.5, como veremos a continuación. Consideremos la definición del valor esperado para una variable aleatoria en el rango de los enteros no negativos,

$$E[X] = \sum_{t=0}^{\infty} P(X > t)$$

Note que:

$$P(X > t) = P(X = t + 1) + P(X = t + 2) + P(X = t + 3) + \cdots$$

entonces

$$\begin{aligned}
\sum_{t=0}^{\infty} P(X > t) &= P(X > 0) + P(X > 1) + P(X > 2) + \cdots \\
&= P(X = 1) + P(X = 2) + P(X = 3) + \cdots \\
&\quad P(X = 2) + P(X = 3) + P(X = 4) + \cdots \\
&\quad P(X = 3) + P(X = 4) + P(X = 5) + \cdots \\
&\quad 1 \cdot P(X = 1) + 2 \cdot P(X = 2) + 3 \cdot P(X = 3) + \cdots \\
&= \sum_{t=0}^{\infty} t \cdot P(X = t) \\
&= E[X]
\end{aligned}$$

Visto gráficamente, figura 3.1, el valor esperado del tiempo para una variable aleatoria la cual toma sus valores en los enteros no negativos, como lo sería para nuestro problema, la variable tiempo cuando se sensa en localidades discretas, es el área por encima de la función de distribución acumulada y por debajo de $y = 1$. Cada término en esta suma puede verse como un rectángulo horizontal con ancho x_j y altura $P(X = x_j)$. Esta suma por partes es análogo al cálculo de la integración. Minimizar el valor esperado es equivalente a minimizar esta área. Una forma de hacerlo es tener intervalos de tiempo tan pequeños y probabilidades tan grandes como sea posible. Esta es la noción que captura nuestra función de utilidad.

La función de utilidad 3.1 es suficiente para definir un algoritmo ávido de un paso adelante, es decir, en cada paso simplemente evaluar la función para todas las ubicaciones disponibles y elegir aquella con el valor más alto. Este algoritmo tiene una complejidad de $O(n^2)$. Sin embargo, podría ser conveniente explorar varios pasos adelante para tratar de escapar de mínimos locales y mejorar la calidad de la solución encontrada. Un inconveniente, al plantear un algoritmo de este tipo, es que comúnmente se incrementa la complejidad por un factor de $O(n)$ por cada paso adelante.

Para reducir este efecto, se propone una segunda heurística, con un efecto similar al algoritmo de ramificación y poda, para reducir el factor de búsqueda en el árbol de las posibles soluciones. La heurística toma en cuenta las variables que participan en el valor esperado del tiempo, la probabilidad y el tiempo empleado para ir de una ubicación a otra.

La heurística expandirá el árbol como sigue, dado un nodo sólo se construyen los nodos hijos que tengan asociados ubicaciónes que no sean estrictamente dominadas respecto a las variables de tiempo y probabilidad.

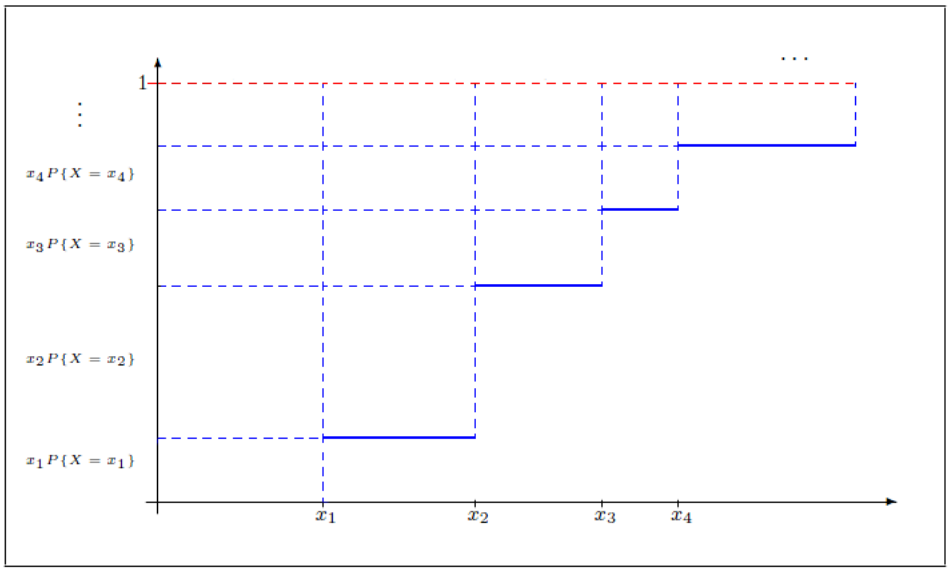


Figura 3.1: Ilustración gráfica del E(X), el valor esperado de X, como el área por encima de la gráfica de la función de distribución acumulada y por debajo de la línea $y = 1$

3.3.1. Ubicaciones estrictamente dominadas

Diremos que una ubicación de sensado domina estrictamente o otra, si su probabilidad de ver el objeto es mayor y el tiempo de alcanzarla es menor. Esto es, si el robot se ubica en L_j, la ubicación L_k domina estrictamente a L_l si se cumplen las siguientes condiciones,

$$P(L_k) > P(L_l)$$

$$Tiempo(L_j, L_k) < Tiempo(L_j, L_l)$$

La figura 3.2 muestra el conjunto de localidades que no han sido exploradas todavía. Graficando las ubicaciones en el plano de tiempo-probabilidad se puede visualizar que aquellas ubicaciones estrictamente dominantes caen en la envolvente convexa del conjunto. Esta envolvente no es cerrada ya que los puntos terminales son ubicaciones indefinidas.

Considerando sólo este subconjunto, las ubicaciones estrictamente dominantes, en cada paso se puede reducir el factor de ramificación, haciendo posible explorar varios pasos adelante sin contravenir el costo computacional. Este algoritmo no preserva la garantía de que la solución óptima se encuentre en este espacio de búsqueda reducido. Los autores en [69, 73] muestran que la heurística tiene un buen desempeño en la práctica, dando ejemplos comparativos para conjuntos manejables de ubicaciones, del orden de 15 ubicaciones de sensado.

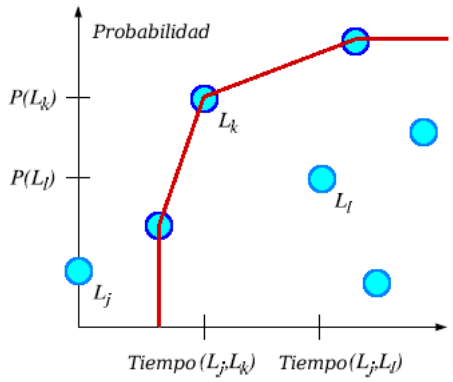


Figura 3.2: Localidades de sensado estrictamente dominadas

3.3.2. Algoritmo: Reduciendo el espacio de búsqueda

El algoritmo completo consiste en explorar de manera iterativa varios pasos adelante, elegir la ruta más prometedora desde un nodo dado, avanzar y volver a empezar.

1. Dada la última localidad, siguiendo la actual ruta solución, se exploran las rutas posibles construyendo un árbol en primera profundidad.

2. Para cada nodo que necesita expandirse, se calcula el conjunto de ubicaciones que no son estrictamente dominadas por otras y se escoge únicamente a esas ubicaciones como nodos hijos.

3. Elegir el mejor nodo hoja basados en la heurística 3.1, se descarta el árbol actual y se vuelve a empezar con el mejor nodo como raíz.

Por lo que respecta a la complejidad, para n ubicaciones de sensado, si el factor de ramificación, número de promedio de nodos hijos por nodo, es B, un árbol con n nodos tiene una altura de $log_B\, n$. De ahí que este subárbol crea una ruta parcial de longitud $log_B\, n$, ya que la ruta solución debe tener una longitud de n, se necesitan $\frac{n}{log_B\, n}$ subárboles para completar la ruta. Esto es lo que describe la figura 3.3, el triángulo grande representa el árbol completo de posibles soluciones, mientras que los pequeños representan los árboles generados con la heurística. Entonces la complejidad del algoritmo es proporcional a explorar un árbol de orden $O(n)$, elegir las configuraciones estrictamente no dominadas con un algoritmo de envolvente

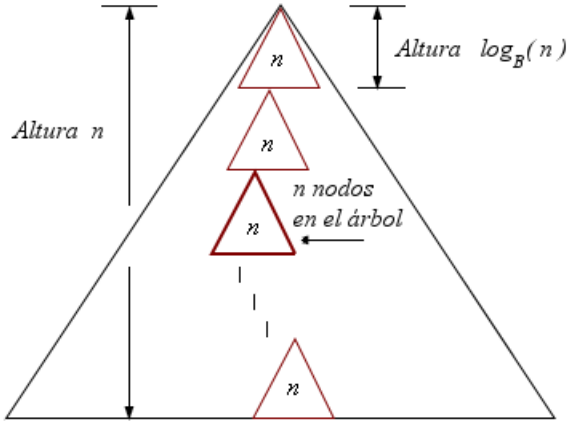


Figura 3.3: Reducción del espacio de búsqueda

convexa en $O(n \log n)$ y repetir el proceso $\frac{n}{\log_B n}$ para obtener la ruta completa, esto es,

$$O\left(n \cdot n \log n \cdot \frac{n}{\log n}\right) = O(n^3)$$

En el peor caso, cuando el factor de ramificación no se reduce del todo, el algoritmo sólo explora un paso adelante y su complejidad es,

$$O\left(n \cdot n \log n \cdot n\right) = O(n^3 \log n)$$

Ahora bien, el problema de minimizar el valor esperado del tiempo para encontrar un objeto en un ambiente en dos dimensiones es un problema NP-duro. El problema para ambientes en tres dimensiones conserva esta complejidad. Y su resolución se torna aún más complicada ya que requerimos calcular la región de visibilidad del sensor en tres dimensiones, el cual es un proceso computacionalmente muy demandante.

Como hemos visto, tratar de encontrar una solución globalmente óptima aún en dos dimensiones en general es un esfuerzo inútil dada la alta complejidad computacional del problema, es por ello que se opta por algoritmos basados en heurísticas para encontrar una buena solución aproximada. Ahora, enfrentar el problema en ambientes tridimensionales, con un robot que ya no sea puntual y que tenga un sensor con capacidades limitadas, se pierde la garantía de minimizar el valor esperado del tiempo. Por ello nuestro trabajo de ahora en adelante, se enfocará a desarrollar estrategias para reducir el valor esperado del tiempo en

encontrar un objeto. Como primera estrategia en el siguiente capítulo describiremos un método para calcular de forma aproximada la región de visibilidad para un sensor acotado tanto en rango como en campo de vista.

Capítulo 4

Estrategia de sensado para ambientes tridimensionales

Como se mencionaba en el planteamiento del problema, capítulo 2, vamos a enfocarnos a desarrollar estrategias de movimiento para un robot que busca un objeto en un ambiente tridimensional. En el capítulo anterior planteamos acotar el problema, a que el robot sólo sense en ubicaciones especificas, trasladando el problema a un marco combinatorio. Entonces uno de los principales subproblemas es definir un conjunto de configuraciones de sensado que colectivamente sensen todo el ambiente. Para obtener este conjunto inherentemente tenemos que calcular la región de visibilidad del sensor. Para ello proponemos una descomposición del espacio libre del ambiente de trabajo en regiones convexas.

A largo del capítulo describiremos el algoritmo de construcción de regiones convexas y en el siguiente detallaremos como empleamos la cobertura convexa para obtener un método rápido para cálcular de forma aproximada la región de visibilidad del sensor.

4.1. Muestro Aleatorio sobre el Ambiente de Trabajo

La descomposición del espacio libre de trabajo, $int(W)$, en regiones convexas se basa en un muestreo aleatorio efectuado en el sistema de coordenadas cartesianas en tres dimensiones, (X, Y, Z).

El algoritmo se inspira en el trabajo presentado en [77]. Ahí, los autores proponen un muestreo aleatorio en el espacio de configuraciones, para construir un roadmap probabilístico.

La técnica utilizada toma ventaja de la noción de visibilidad. Cada configuración libre de colisión generada es un nodo del roadmap sólo si sirve para conectar dos componentes que no estaban conectados o si no son visibles a otra configuración declarada como guardia. La estrategia propuesta construye roadmaps con una pequeña cantidad de nodos y sin requerir de un cálculo explícito de las regiones de visibilidad.

Hay varias diferencias entre el trabajo descrito en [77] y el algoritmo que proponemos: 1) nuestro algoritmo realiza un muestreo aleatorio en el espacio de trabajo y no en el espacio de configuraciones y 2) mantenemos todas las muestras generadas, ya sea que estén en el espacio libre o en el espacio ocupado por un obstáculo; las muestras que colisionan con un obstáculo son almacenadas para determinar el espacio en el ambiente ocupado por el obstáculo, en caso de cambiar su posición original. Esencialmente del trabajo presentado en [77], sólo rescatamos la manera de generar el conjunto de configuraciones llamadas guardias, que para nosotros es un *conjunto de puntos guardias*, es decir, ternas (x,y,z).

4.1.1. Algoritmo de Muestreo

Generamos aleatoriamente un *conjunto de puntos muestra*, S, con el fin de capturar aproximadamente la forma y tamaño del espacio de trabajo, W. Utilizamos una función de distribución uniforme para generar los puntos muestra. Guardamos las muestras en dos subconjuntos, el conjunto S_{free} corresponde a las muestras que caen en el espacio libre $int(W)$, y el conjunto S_{obs} corresponde a las que caen en el espacio ocupado por un obstáculo en el ambiente.

$$S = S_{free} \cup S_{obs} \quad con \quad S_{free} \cap S_{obs} = \emptyset$$

Dadas dos muestras aleatorias s y $s' \in S_{free}$, s y s' son visibles si existe el segmento $\overline{s\,s'} \subseteq int(W)$.

Dada una muestra $s \in S_{free}$ denotamos como $Vis(s)$ al conjunto de muestras en S_{free} que son visibles a s.

Dado el conjunto S_{free} vamos a elegir un subconjunto de puntos, al que le denominamos *conjunto de puntos guardia ocultos, G*, que cumple las propiedades siguientes:

1. Para cualquiera dos elementos g_i, $g_j \in G$ se cumple que no son mutuamente visibles.

$$g_i \notin Vis(g_j) \quad y \quad g_j \notin Vis(g_i) \qquad \forall i \neq j$$

2. Colectivamente los elementos de G ven a todas las muestras en S_{free}, es decir, cada muestra debe ser visible al menos por un guardia.

$$\bigcup_{g_i \in G} Vis(g_i) = S_{free}$$

A continuación describimos el algoritmo que genera el muestreo aleatorio y que selecciona el conjunto de puntos guardia ocultos.

Algoritmo 4.1: Genera Muestras y Guardias

Entrada: Ambiente 3D

Salida: G, conjunto de puntos guardia ocultos; S, Conjunto de puntos muestra aleatorios

```
1  begin
2      G ⟵ ∅
3      S ⟵ ∅
4      n ⟵ 0
5      M ⟵ constante
6      while n ≤ M do
7          s ⟵ random(1)
8          if (s en el int(W) ) then
9              if (s visible por algún g ∈ G) then
10                 S_free ⟵ s
11                 n ⟵ n + 1
12             else
13                 G ⟵ s
14                 n ⟵ 0
15         else
16             S_obs ⟵ s
17     S ⟵ S_free ∪ S_obs
```

Evidentemente, por construcción, el algoritmo obtendrá un conjunto de guardias que ven a todas las muestras restantes generadas en el espacio libre. Cada nuevo guardia insertado al conjunto incrementa la cobertura de $int(W)$, por lo que la probabilidad de generar muestras en espacios que aún no son cubiertos por un guardia se mantiene decreciendo a cada iteración del algoritmo.

Ahora presentaremos una estimación probabilística de la proporción del volumen no cubierto por los guardias. Consideremos al espacio libre de volumen unitario, como se ve en la figura 4.1. Supongamos que la proporción del ambiente que es visible desde el conjunto de guardias es A y tiene tamaño $\mu(A) = 1 - \epsilon$, mientras B es la proporción que aún no es visible a los guardias y que no contiene ninguna muestra, B tiene tamaño $\mu(B) = \epsilon$.

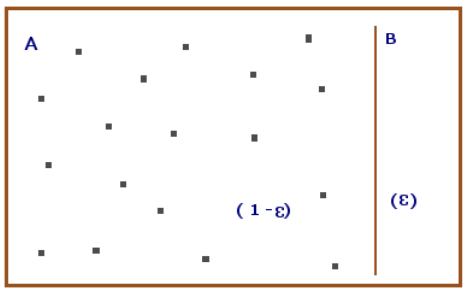


Figura 4.1: Muestreo aleatorio del espacio libre

Si las muestras se generan independientemente la probabilidad que m muestras consecutivas no caigan en la proporción B es $P(A_m) = (1-\epsilon)^m$. Después de generar m muestras consecutivas, con un valor de m pequeño es posible que el tamaño de B aún sea más grande que ϵ. Podemos acotar esta probabilidad con un valor pequeño α, así podemos determinar probabilísticamente un valor para m como sigue:

$$\begin{aligned} (1-\epsilon)^m &\leq \alpha \\ mlog\,(1-\epsilon) &\leq log\,(\alpha) \\ m &\geq \frac{log\,(\alpha)}{log\,(1-\epsilon)} \end{aligned}$$

Por tanto, eligiendo un valor lo suficientemente grande para m, podemos esperar con probabilidad $(1-\alpha)$ que el tamaño de la región B no cubierta por los guardias sea a lo más ϵ.

El algoritmo 4.1 descrito anteriormente regresa un conjunto de guardias con una probabilidad de cubrir la gran mayoría del espacio libre. Cada muestra generada aleatoriamente será vista por al menos un guardia.

Siguiendo nuestro objetivo de generar configuraciones de sensado para cubrir el ambiente, vamos a utilizar el conjunto de puntos muestra, S, y el conjunto de puntos guardia ocultos, G, para descomponer el espacio libre en regiones convexas que se traslapan, es decir, construir una cobertura convexa. Más adelante utilizaremos esta cobertura para aproximar las regiones de visibilidad del sensor limitado en rango y campo de vista.

4.2. Cobertura Convexa

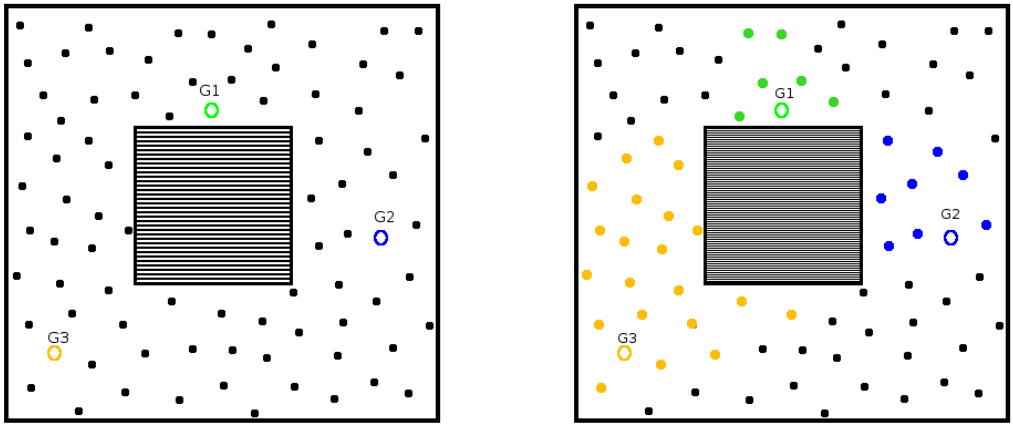
El algoritmo que se propone para la cobertura convexa está basado en la idea de existencia de una dualidad entre *el conjunto máximo de guardias ocultos* y una *cobertura mínima convexa*. Se ha probado que encontrar una cobertura mínima es un problema NP-Duro [56], por tanto el objetivo primordial es generar un algoritmo eficiente que trate de generar tan pocas regiones como sea posible. En la práctica, hemos observado que el algoritmo encuentra la cobertura convexa mínima para algunos ambientes.

Si la densidad de muestras en el espacio de trabajo es adecuada, de acuerdo a la estrategia estocástica que se utiliza para generar los guardias, habrá una vecindad de muestras para cada guardia que solamente él podrá ver, llamamos a este conjunto de muestras el *kernel del guardia*, y lo denotamos como sigue,

$$\forall g_i \in G \qquad Ker(g_i) = Vis(g_i) \setminus \bigcup_{i \neq j} Vis(g_j)$$

En cualquier cobertura convexa mínima C, cada región convexa C_i tiene un conjunto único de puntos contenidos en esa región en particular. De no ser así, la región C_i simplemente podría ser removida y la cobertura ya no sería mínima. A pesar de saber que no hay una equivalencia exacta, utilizamos los kernels de los guardias para aproximar esos conjuntos únicos. Entonces, la idea principal del algoritmo es construir una región convexa con el kernel de cada guardia para generar un conjunto de baja cardinalidad de regiones convexas.

La figura 4.2 presenta un espacio bidimensional el cual contiene un simple obstáculo cuadrado al centro, en 4.2(a) se pueden observar las muestras generadas aleatoriamente en el espacio libre (puntos negros pequeños) y un conjunto de guardias ocultos (circunferencias de color verde G1, azul G2, y amarillo G3) que ven a cada una de las muestras. En 4.2(b) se pueden apreciar los kernels de los guardias, puntos en color verde, azul y amarillo, según corresponde el guardia.



(a) (b)

Figura 4.2: Guardias y kernels

A continuación presentamos la descripción del algoritmo de construcción de la cobertura convexa. En las figuras 4.3 y 4.4 mostramos visualmente el desarrollo del algoritmo.

1. Como primer paso, se elige al guardia con el kernel más grande, $g = argmax_j \{ker(g_j)\}$ y con la ayuda de un algoritmo iterativo de envolvente convexo se construye un envolvente convexo alrededor del correspondiente guardia con las muestras del kernel, $Q_O = convex(Ker(g))$, figura 4.3(a). Para este fin, se inicia con el guardia y se agregan paulatinamente muestras del kernel al actual envolvente convexo usando el algoritmo presentado en [58]. Por cada punto del kernel agregado al envolvente convexo se generan nuevas aristas y por lo que se debe de asegurar que no colisionan con algún obstáculo del ambiente.

2. Si en algún punto, el envolvente convexo instersecta con un obstáculo, $Q_0 \cap W \neq \emptyset$, el guardia seleccionado se mueve fuera del actual envolvente convexo a la ubicación de alguna otra muestra perteneciente al kernel. Dado que el kernel es el conjunto de muestras que sólo el correspondiente guardia puede ver, se conserva la propiedad de ser oculto al resto de los guardias, figura 4.3(b). Al hacer este movimiento algunas muestras pueden quedar sin ser vistas por algún guardia, si esto ocurre se genera un nuevo guardia. Una vez que se tienen cubiertas nuevamente todas las muestras, es decir, que cada muestra es visible a al menos un guardia, se procesan los nuevos guardias para determinar el kernel que les corresponde, figura 4.3(c). Entonces volvemos al paso 1.

3. Si envolvente convexo alrededor del guardia contiene a todo el kernel sin intersectar con

el ambiente, $Q_0 \cap W = \emptyset$, figura 4.3(d), entonces puede continuar creciendo, agregándole puntos de S que son visibles al guardia mientras que las nuevas aristas generadas por esos puntos no produzcan alguna intersección con los obstáculos del ambiente, ver figura 4.4(a). Es decir, se genera el envolvente convexo Q_{i+1} con $Q_i \bigcup \{s\}$ para alguna $s \in S$, sólo sino intersecta con algún obstáculo, $Q_{i+1} \bigcap W = \emptyset$. Cuando el envolvente convexo alcanza su tamaño máximo, se convierte en una nueva región integrante de la cobertura convexa, y se remueve el guardia original g de G.

4. Si todos los guardias han sido procesados el algoritmo termina, 4.4(b), sino volvemos al paso 1.

Este algoritmo, de construcción de cobertura convexa, escala bien con la dimensión del ambiente siempre que se implemente un algoritmo apropiado de envolvente convexo y una rutina para detectar colisión de segmentos y facetas. En particular, adoptamos el algoritmo incremental para envolventes convexos en tres dimensiones de [58] y adaptamos una rutina para intersección de planos para implementar el algoritmo para espacios de trabajo tridimensionales.

A continuación presentamos visualmente los resultados del algoritmo para un ambiente 3D sencillo. El ambiente es un cuarto cerrado con una pared en el interior la cual tiene un hueco que permite ver a través de ella, 4.5(a). La figura 4.5(b) muestra la nube de puntos generados aleatoriamente que cubren el espacio libre del ambiente. Como se puede observar dos guardias serían suficientes para ver el conjunto de muestras pero de acuerdo al algoritmo se construirían dos regiones convexas con los kernels de los guardias las cuales no cubrirían completamente el espacio libre. El algoritmo genera cuatro guardias ocultos, las figuras 4.6(a) y 4.6(b) muestran los guardias ocultos con sus respectivos kernels, los cuadrados grandes representan a los guardias y los puntos pequeños representan a los elementos del kernel.

Los guardias azul y rosa cubren las porciones del ambiente a los lados de la pared, el guardia negro cubre la parte angosta y el guardia amarillo cubre el espacio del hueco, sus kernels se aprecian en el mismo color, azul, rosa, negro y amarillo, respectivamente. El algoritmo construye las correspondientes regiones convexas para cada kernel y las crece hasta alcanzar el tamaño máximo permitido por el ambiente, figura 4.7. También la figura 4.7 ilustra un caso en el cual el algoritmo genera el mínimo número de regiones convexas para cubrir el ambiente.

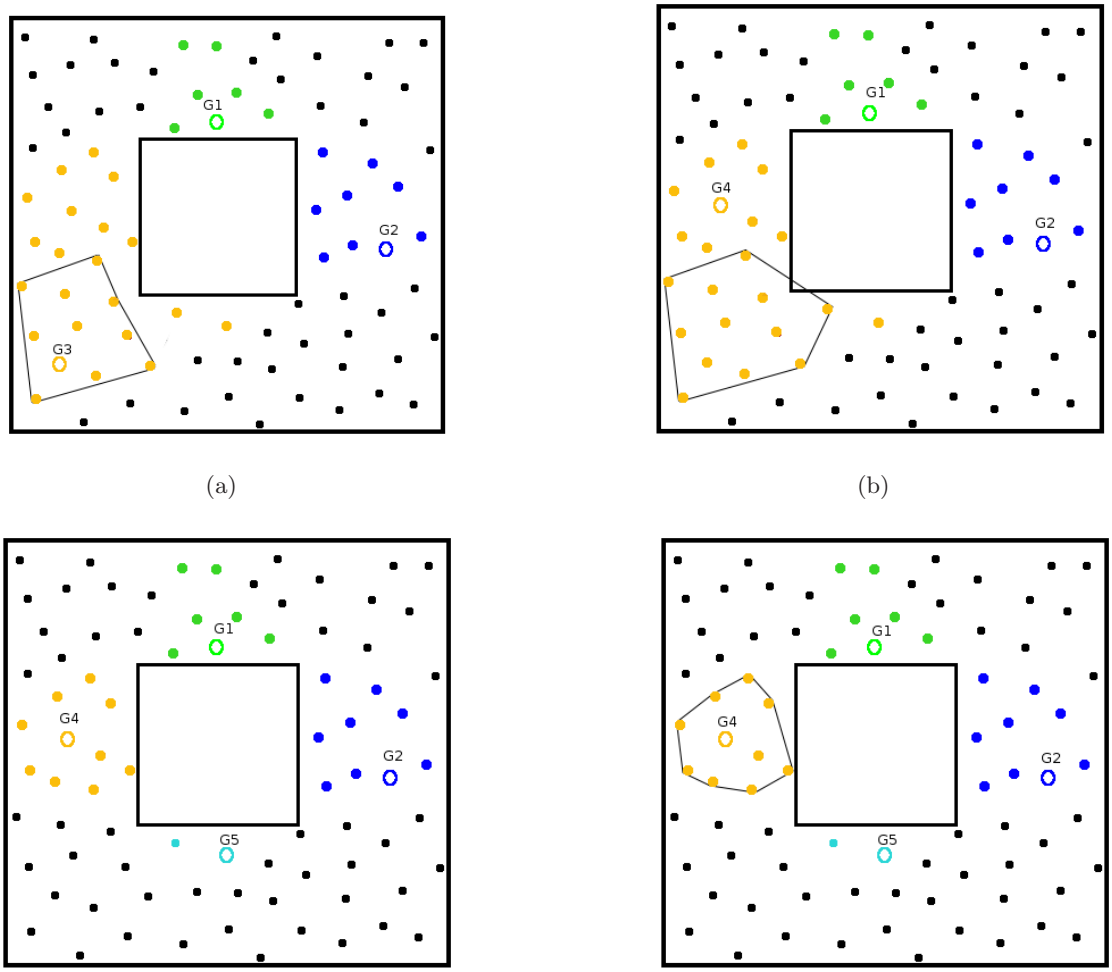
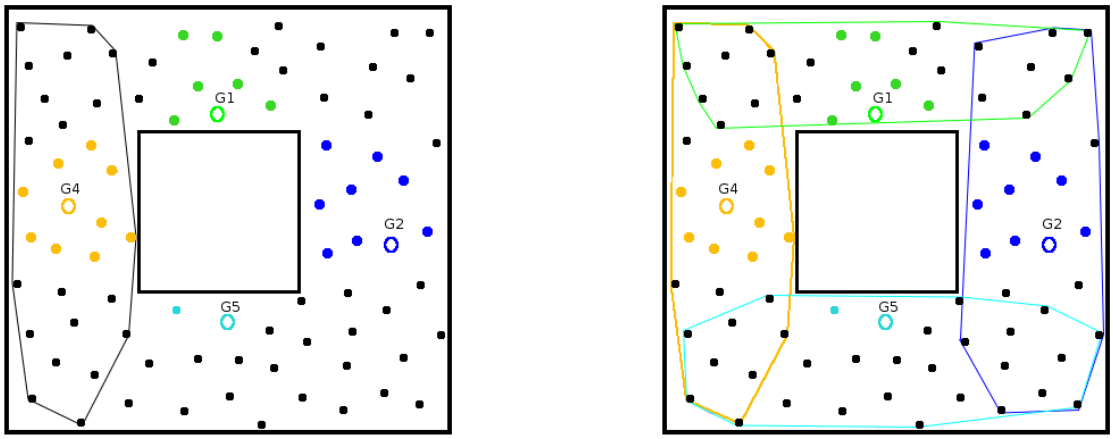
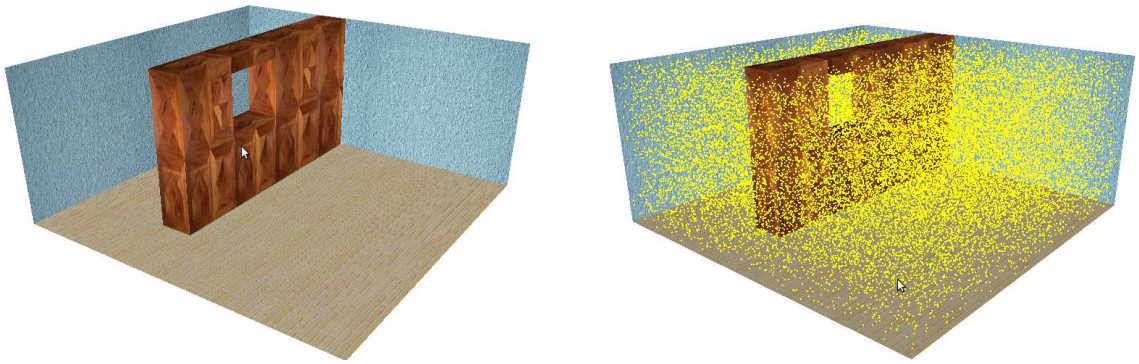


Figura 4.3: Algoritmo: Cobertura convexa del espacio libre



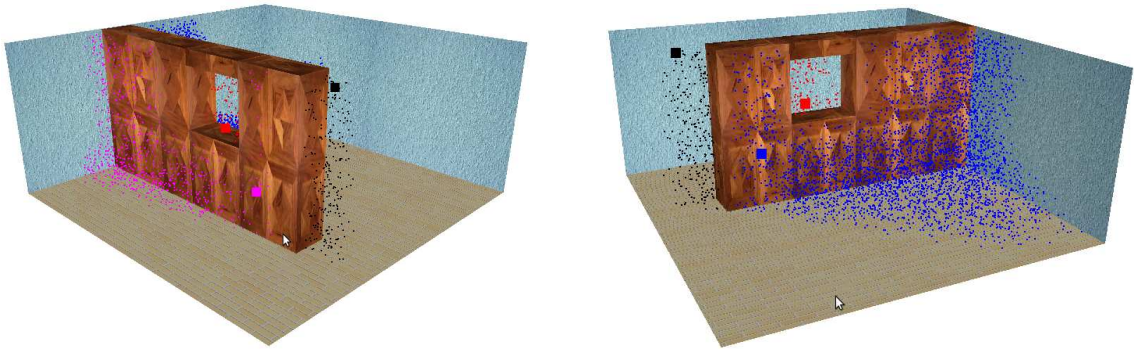
(a) (b)

Figura 4.4: Algoritmo: Cobertura convexa del espacio libre



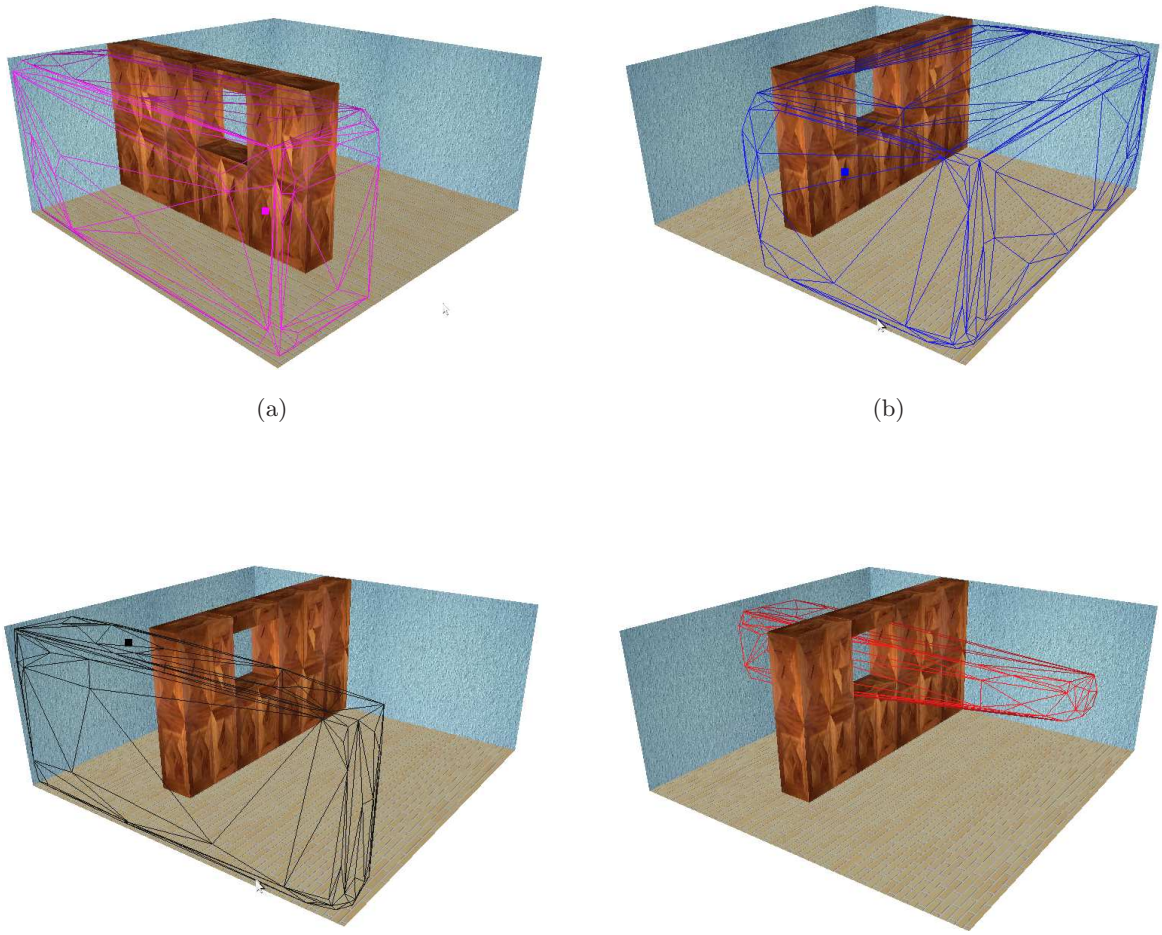
(a) (b)

Figura 4.5: Conjunto de muestras aleatorias en un ambiente 3D



(a) (b)

Figura 4.6: Conjunto de guardias ocultos y sus kernels respectivos



(a) (b)

(c) (d)

Figura 4.7: Conjunto de regiones convexas en un ambiente 3D

Capítulo 5

Trayectorias Dependientes de la Métrica

5.1. Robot Manipulador Móvil

En esta tesis consideramos un robot manipulador móvil para la ejecución del método de búsqueda. El robot se compone de una base móvil y un "brazo" y consta de siete grados de libertad, (g.d.l.), dos de ellos correspondientes a la posición de la base en el plano (x, y), un grado de libertad para la rotación de la base, θ_1 y cuatro grados de libertad para la rotación de las articulaciones del brazo, $(\theta_2, \theta_3, \theta_4, \theta_5)$, ver figura 5.1. Suponemos un sistema robot-sensor denominado *eye-in-hand*, lo cual significa que el sensor se encuentra en el efector final, la figura 5.1 también muestra el sistema de coordenadas donde reside el sensor. El sensado se realiza en configuraciones discretas.

Cabe notar que para colocar el sensor en alguna configuración en el espacio de trabajo tridimensional, se deberían determinar 6 grados de libertad, 3 para la posición del sensor y 3 para la orientación del sensor (ángulos). Sin embargo, sólo consideramos 5 g.d.l., tres para las coordenadas del espacio de trabajo (x,y,z) y dos para la orientación, los ángulos para rotar alrededor del eje Z y del eje Y; la rotación alrededor del eje X no se considera. Entonces, teniendo en cuenta estos grados de libertad, nuestro robot es un sistema redundante para la tarea de colocar el sensor en una configuración; el robot tiene dos g.d.l. extra con respecto a los 5 grados requeridos para definir una configuración del sensor en el espacio de trabajo. En [61], los autores proponen una técnica para mover un manipulador móvil asegurando estabilidad

dinámica; la redundancia en el sistema se usa para alcanzar estabilidad cuando el sistema es inestable. En este trabajo, los grados de libertad extra nos permiten sensar una misma porción del ambiente desde diferentes configuraciones del robot.

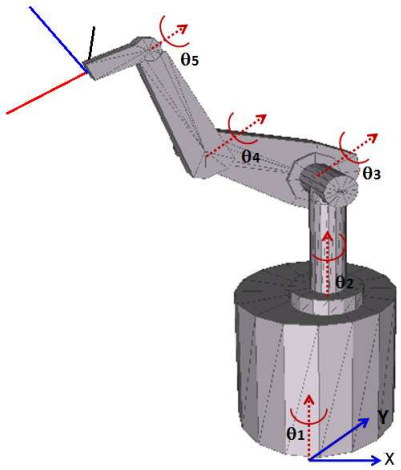


Figura 5.1: Robot manipulador móvil

Como mencionamos en el planteamiento del problema, estamos considerando que el robot eventualmente encontrará el objeto cuando sense el espacio libre del ambiente, la descomposición en regiones convexas nos proporciona una herramienta para dirigir al robot en esta búsqueda. El robot sensará una región convexa tras otra hasta cubrir todo el ambiente. En caso de un sensor omnidireccional, la descomposición en regiones nos otorga flexibilidad en los lugares donde colocar el efector final para sensar cada región, cualquier punto (x, y, z) dentro de una región convexa es un candidato válido. Por una parte, esto simplifica el problema de planificación, ya que tenemos un conjunto de configuraciones meta contrario a una sola. Sin embargo, por otra parte, esa flexibilidad induce un problema de decisión, dado que tenemos más opciones de donde colocar el sensor, ¿cuál es el mejor camino para alcanzar cada una de las regiones?. Nosotros no estamos interesados solamente en alcanzar las regiones sino en encontrar la mejor manera de hacerlo.

Cuando suponemos un sensor limitado en rango y campo de vista utilizamos el algoritmo propuesto en el capítulo 3 para encontrar un orden de visita tanto para las regiones convexas como para las configuraciones de sensado. Pero no hay que perder de vista el valor esperado del tiempo en encontrar el objeto, ya que uno de nuestros objetivos es su minimización o al menos una reducción; el cálculo del valor esperado del tiempo depende de dos factores principales: 1) el costo de mover al robot entre configuraciones de sensado y 2) la masa de

probabilidad relacionada con el sensado del objeto (ganancia). Por lo que necesitamos encontrar las trayectorias más cortas para mover al robot manipulador de 7 g.d.l. entre configuraciones de sensado.

Las trayectorias más cortas dependen de la métrica utilizada para calcular la distancia o costo. En general, una forma de definir la distancia entre dos configuraciones X y Y en un espacio de dimensión D, esta dado por la siguiente ecuación,

$$\|X - Y\|_{\Lambda} \equiv (X - Y)^T \Lambda (X - Y)$$

donde Λ es una matriz diagonal con valores positivos (costos), $\lambda_1, \lambda_2, \ldots, \lambda_d$, asignados a los diferentes grados de libertad. En general, la matriz Λ es necesaria porque las articulaciones podrían no ser equivalentes (eslabones cortos contra largos) o porque los g.d.l. podrían medir diferentes tipos de movimiento, comparar rotación contra traslación.

En nuestro proyecto, asignando costos diferentes a cada g.d.l. podemos proveer de prioridades a los dos componentes principales del sistema: la base móvil y el brazo robótico. Dividimos los g.d.l. del manipulador móvil en dos grupos, uno contiene los dos g.d.l. para trasladar al robot, y el otro incorpora los cinco g.d.l. para la rotación del sistema. Este agrupamiento se hace sin perdida de generalidad ya que los costos pueden ser ajustados individualmente.

En este punto, el problema es encontrar los caminos más cortos a un grupo de configuraciones. Debido a que, inherentemente, es un problema más complicado que el problema clásico de planificación de caminos, el cual sólo considera un camino y no necesariamente el de menor costo, hemos decidido implementar un algoritmo de expansión de frente de onda, llamado NF1, [37]. Este algoritmo es exponencial cuando, D, el número de grados de libertad no está acotado. Pero si nos restringimos a un robot en particular, como el manipulador móvil de siete grados de libertad, $D = 7$, la complejidad es polinomial.

5.2. Algoritmo de expansión de frente de onda

El algoritmo de expansión de frente de onda, NF1 (del inglés; Navigation Function 1), está basado en los métodos de navegación de campos de potencial. En general, el algoritmo discretiza el espacio de configuraciones en una malla. La función de navegación se calcula con una técnica de propagación de onda que empieza en la celda meta asociada con la configuración meta. El algoritmo etiqueta cada celda de la malla con la distancia L_1 o distancia Manhattan,

tomando en cuenta las obstrucciones causadas por los obstáculos en algunas celdas. Así, todas las celdas en la frontera de la onda tienen la misma distancia a la celda meta.

A continuación, describimos el algoritmo para dos dimensiones, el plano x, y.

- Construir una malla rectangular fina, MC_{libre}, sobre el espacio de configuraciones, discretizando los rangos de valores permitidos para cada g.d.l.
- Dos puntos de la malla q y q' son $1-$vecinos si difieren exactamente en un incremento del valor de la malla, (a lo sumo 1 unidad).
- (x, y) es $1-$vecino de $(x \pm 1, y)$ y $(x, y \pm 1)$.
- U es una función de navegación simple, en este caso la distancia Manhattan, L_1.

Algoritmo 5.1: Navigation Function 1, NF1

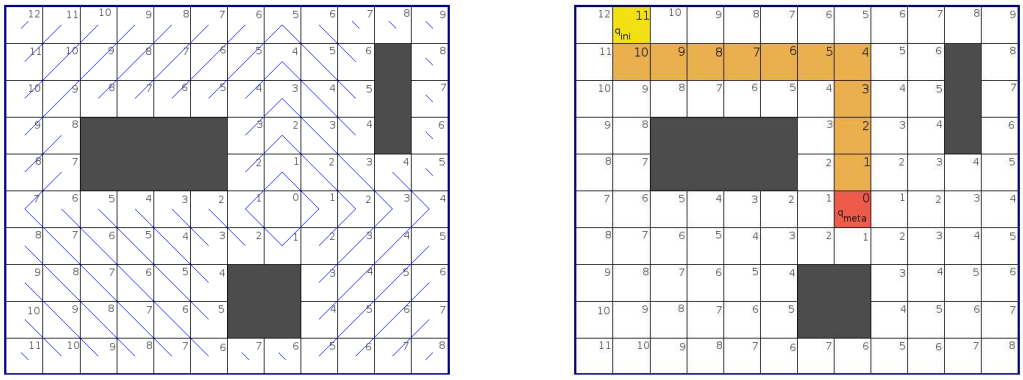
Entrada: Malla en el espacio de configuraciones.

Salida: Malla evaluada por la función de navegación.

```
1  begin
2      Lq ⟵ ∅      // Lista de configuraciones
3      Lq_i ⟵ ∅
4      foreach q en MC_libre do
5          U(q) ⟵ −1      // Etiqueta una celda no evaluada
6      U(q_meta) ⟵ 0
7      Agrega (Lq, q_meta)
8      while Lq ≠ ∅ do
9          foreach q_i en Lq do
10             foreach 1-vecino q'_i de q_i en MC_libre do
11                 if U(q'_i) = −1 then
12                     U(q') ⟵ U(q_i) + 1
13                     Agrega (Lq_i,q')
14         Lq ⟵ Lq_i
```

En la figura 5.2(a) mostramos un ejemplo de la ejecución del algoritmo. El algoritmo NF1 utiliza una técnica de propagación de onda que empieza desde la celda meta, la cual tiene el

valor 0, toma los 1−vecinos y evalúa la función de navegación asignando el valor de 1, después se evalúa la función de navegación en cada una de las celdas de la primera onda para generar la segunda onda, celdas con valor 2, y así sucesivamente hasta evaluar todas las celdas de la malla. Como se muestra en la figura 5.2(a) las celdas contendrán valores ascendentes como se alejen de la celda meta. Para recuperar el camino desde la celda asociada a la configuración inicial a la celda ligada a la configuración meta, se siguen los valores descendentes asignados por la función de navegación de cada celda hasta llegar al valor mínimo, 0, ver la figura 5.2(b).



(a) Propagación de onda

(b) Recuperación del camino

Figura 5.2: Función de Navegación

5.2.1. Modificación del algoritmo NF1

Haciendo modificaciones al algoritmo NF1 podemos encontrar un conjunto de configuraciones de sensado y los caminos para alcanzar las regiones convexas preservando la optimalidad del camino. Los caminos encontrados para alcanzar las regiones convexas serán óptimos en distancia L_1. Las modificaciones al algoritmo 5.1 se resumen a continuación:

- Como la configuración meta es desconocida iniciamos la expansión de la onda a partir de la configuración inicial. Asignamos el valor 0 a la celda asociada a la configuración inicial. $U\left(q_{ini}\right) \longleftarrow 0$. Línea 6.
- Mientras se expande la onda agregamos un mecanismo para detectar si la configuración evaluada por la función de navegación alcanza una nueva región. Línea 12.
- El algoritmo termina cuando se tiene un conjunto de celdas, configuraciones, pertenecientes a la malla MC_{libre} que colocan al efector final dentro de cada región convexa. Línea 8.

- El camino para ir de la configuración inicial a la configuración que alcanza a cada región, al igual que en el ejemplo de la figura 5.2, lo recuperamos siguiendo los valores descendentes asignados por la función de navegación a cada celda hasta llegar al valor mínimo, 0.

Asignar un costo mayor a uno de los grupos de grados de libertad, ya sea al grupo de traslación o de rotación, se verá reflejado en los caminos más cortos. Es evidente que esos caminos tendrán un valor menor cuando los g.d.l. que más se muevan sean aquellos de menor costo. Ahora, podemos seleccionar los grados de libertad que nos gustaría que se movieran más (o menos), tratando de encontrar trayectorias que mantengan un compromiso en el movimiento del sistema, por ejemplo, entre mover la base y el brazo del robot para efectuar la búsqueda.

Por construcción el algoritmo NF1, avanza de 1-vecino en 1-vecino no importando el costo del movimiento. Pero si consideramos la métrica adoptada, nuestro algoritmo de expansión de frente de onda usa los costos ($\lambda_1, \lambda_2, \ldots, \lambda_d$ de los g.d.l.) para expandir la onda en las diferentes dimensiones. El algoritmo sesgará la expansión en dirección de los g.d.l. con menor costo. Regresando a nuestro ejemplo en dos dimensiones, en la figura 5.2(a) los g.d.l. en X y Y tienen igual costo, por lo que la expansión es simétrica en ambas direcciones. Cuando asignamos un menor costo a uno de los grados de libertad, por ejemplo en la coordenada x la propagación de la onda se sesgara. En la figura 5.3 podemos ver que la expansión es mayor en la dirección x, siendo este grado de libertad el de menor costo.

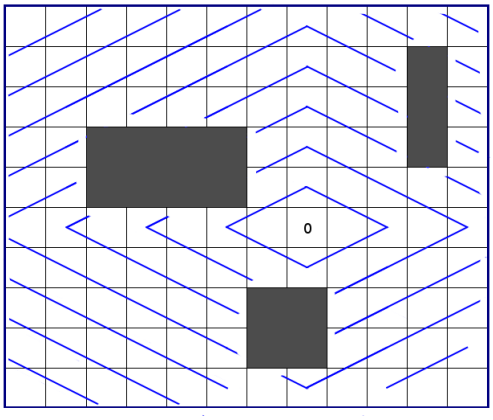


Figura 5.3: Sesgo en la expansión de frente de onda por la métrica

Conjuntando el algoritmo modificado de expansión de frente de onda, como se describio en la sección 5.2.1, con la cobertura convexa obtenemos los resultados deseados para ambientes en tres dimensiones. Esto es, podemos encontrar un conjunto de configuraciones de sensado y los caminos más cortos para alcanzarlas dependientes de la métrica en distancia L_1 .

Capítulo 6

Sensor limitado y selección de grados de libertad a optimizar

En el capítulo 3, desarrollamos un algoritmo para definir un orden con el cual el robot visite un conjunto de configuraciones de sensado y así cubrir el espacio libre de trabajo en búsqueda de un objeto. En este capítulo nos enfocaremos a describir como elegimos ese conjunto de configuraciones de sensado y la estrategia global para encontrar el objeto.

Hemos extendido nuestras estrategias de movimiento, para la búsqueda de un objeto con un robot manipulador móvil equipado con un sensor limitado tanto en rango de alcance mínimo como máximo, así como en campo de vista; la región de visibilidad del sensor limitado tiene la forma de un frustum, $\mathcal{F}$, ver la figura 6.1. Proponemos un método para aproximar la región de visibilidad en tres dimensiones para el sensor limitado. El objetivo del método es determinar la porción del ambiente que el sensor cubre, usando el conjunto de muestras aleatorias S en el espacio libre de trabajo.

Ahora, describimos nuestra estrategia de movimiento para la búsqueda: primero, establecemos un orden de visita para cada región convexa C_r. Este orden sería el mismo si utilizáramos un sensor omnidireccional. Una vez planeado este orden, generamos otro orden para visitar las configuraciones de sensado, q_{l_k}, que colectivamente cubrirán el espacio interno de una región convexa dada. Ambos ordenes son calculados como se indica en la sección 3.3. La selección de las configuraciones de sensado, para cubrir cada región convexa, considera las porciones del ambiente que ya han sido cubiertas mientras se sensaban otras regiones previamente, según el orden definido.

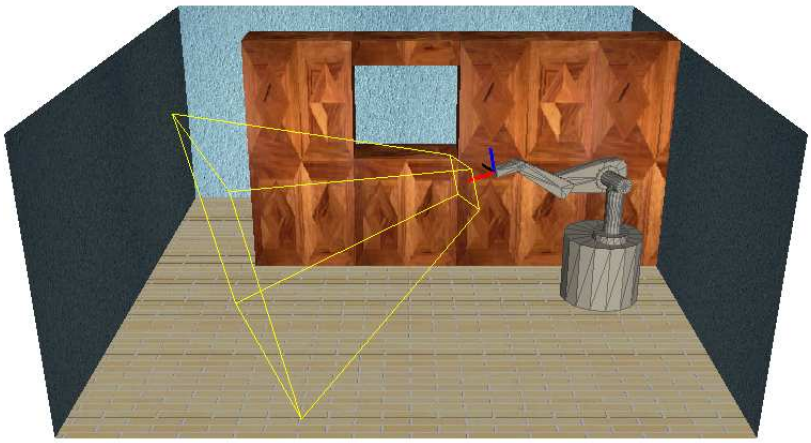


Figura 6.1: Sensor limitado en rango mínimo y máximo y en campo de vista.

6.1. Aproximación de la región de visibilidad

Nuestro método para aproximar las regiones de visibilidad del sensor en 3D utiliza la cobertura convexa C, que ya hemos descrito en el capítulo 4. Las regiones convexas facilitan el cálculo de las regiones de visibilidad para el sensor limitado, evitando el uso de estructuras de datos complejas para determinar cuales porciones del ambiente son cubiertas por el sensor. De acuerdo con el orden de visita de las regiones, el robot irá sensando una región tras otra. Una región es cubierta o sensada si las muestras utilizadas para aproximar su volumen han sido sensadas o equivalentemente, si la unión de los frustums pertenecientes a las configuraciones de sensado han cubierto todas las muestras que se encuentran dentro de la región convexa. Con un sensor omnidireccional, las regiones de visibilidad en 3D se podrían aproximar burdamente con las regiones convexas, esto es, generando una configuración de sensado que coloque el efector final dentro de una región convexa dada, se garantiza que cualquier parte en el interior de la región es visible al sensor. Ahora, para un sensor limitado, la región de visibilidad la aproximamos con las muestras cubiertas por el frustrum del sensor y las regiones convexas donde se encuentra situado el efector final.

Cada región convexa dada, $C_r \in C$, tiene asociada un conjunto de muestras aleatorias S_r que originalmente se usaron para calcular el volumen de la región, $S_r = \{s \in S \mid s \text{ está en el interior de } C_r\}$. Recordemos que, cada muestra s está en el espacio 3D y se define como un punto tridimensional $s(x, y, z)$. Usamos el conjunto S_r para determinar la cobertura del espacio de la región C_r por el sensor limitado. Para cubrir cada región convexa C_r, generamos un conjunto de configuraciones de sensado Q_r. Todas las configuraciones de sensado q_{l_k} que usamos para

cubrir una región convexa tienen la propiedad de colocar el efector final del robot, donde reside el sensor, dentro de la misma región convexa. Gracias a esta propiedad podemos aproximar la región de visibilidad sin complejos cálculos de visibilidad 3D. Entonces el conjunto $Q_r = \{q_{l\_k} \mid q_{l\_k}$ coloca el efector final en el interior de $C_r\}$. Cabe notar que en la notación C_r, S_r, y Q_r, el subíndice r corresponde al identificador asociado a la región.

Dada una configuración de sensado q_{l_k} usamos los puntos muestra que caen en el frustum para aproximar la región de visibilidad $V(q_{l\_k})$ asociada a esa configuración. Entonces, la región de visibilidad del sensor limitado en la configuración q_{l_k} se aproxima como el conjunto de muestras que caen en el poliedro generado por la intersección del frustum $\mathcal{F}$ del sensor y la región convexa C_r que estamos sensando.

$$V(q_{l\_k}) = \{s \in S_r \mid s \text{ está en el interior de } (\mathcal{F} \cap C_r)\}$$

En la figura 6.2, podemos ver una región convexa y una configuración de sensado que coloca el efector final al interior de la región. Resaltamos con puntos amarillos las muestras que aproximan la región de visibilidad correspondiente a la configuración.

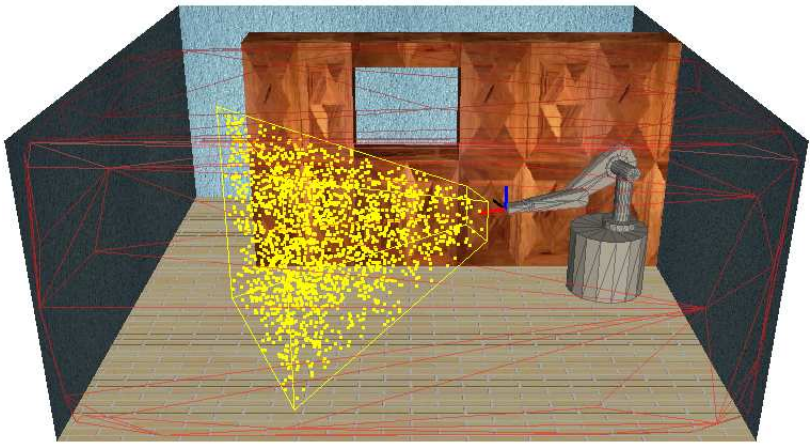


Figura 6.2: Primera aproximación de la región de visibilidad

Note que al estar cubriendo la región C_r la configuracion q_{l_k} puede colocar el sensor en el interior de varias regiones a la vez, en tal caso, la región de visibilidad la aproximamos como el conjunto de muestras que están simultáneamente en la intersección del frustum con las regiones convexas donde reside el sensor. Sea $R = \{R_1, R_2, \ldots, R_m\}$ el subconjunto de regiones convexas, $R \subseteq C$, donde se encuentra el efector final dada la configuración q_{l_k}. Lógicamente C_r es igual a algún R_i, ya que es la región que estamos sensando. Entonces aproximamos la región de

visibilidad como:

$$V(q_{l\_k}) = \left\{ s \in S \;\middle|\; s \text{ está en el interior de } \bigcup_{R_i \in R} (\mathcal{F} \cap R_i) \right\} \tag{6.1}$$

Por tanto, siguiendo el orden de visita para cubrir cada región convexa, mientras sensamos la región C_r también marcamos como sensadas y lógicamente removidas las muestras pertenecientes a las regiones en R que intersectan con el frustum del sensor. Así se garantiza que estas muestras están libres de oclusión desde la configuración q_{l_k}. En la figura 6.3 representamos con puntos amarillos las muestras que corresponden a la intersección del frustum del sensor y la región convexa que estamos sensando y con puntos morados las muestras que están en otra región donde también se localiza el efector final.

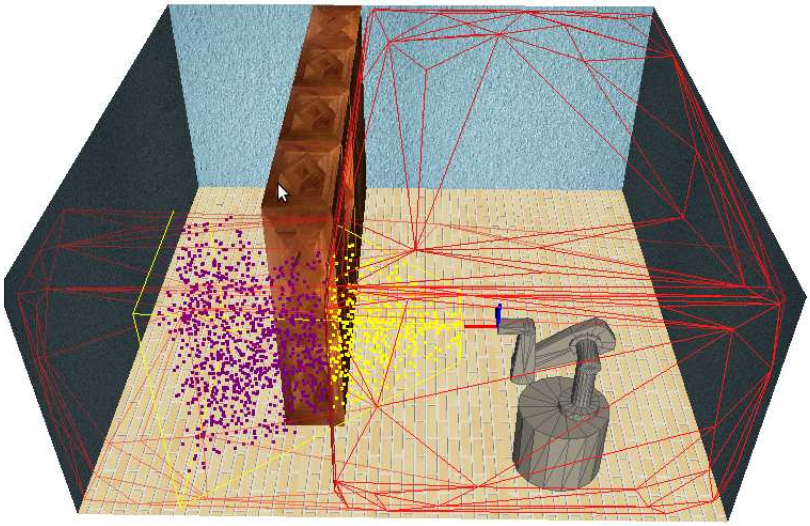


Figura 6.3: Aproximación de la región de visibilidad

6.2. Selección de las configuraciones de sensado

En nuestro método, para cubrir las regiones convexas generamos un conjunto de configuraciones de sensado con una función de distribución de probabilidad uniforme sobre los siete grados de libertad del robot, las coordenadas (x, y) para la posición de la base y los cinco ángulos, $\theta_i,\ i = 1 - 5$, para la rotación de la base y las uniones del brazo. Cada configuración de sensado está asociada a una región convexa en particular. De acuerdo con el orden de visita

de las regiones convexas, generamos aleatoriamente un conjunto de configuraciones de sensado Q_r para cubrir la región C_r.

Existen diversos problemas que involucran encontrar un conjunto de configuraciones efectivas para el robot, por ejemplo, en [84] los autores requieren configuraciones para determinar una postura efectiva del robot para agarrar un objeto. En nuestro problema de búsqueda, requerimos configuraciones de sensado que disminuyan el valor esperado del tiempo de encontrar un objeto. Por esta razón, determinamos que nuestro conjunto de configuraciones de sensado debe cumplir dos propiedades: una baja cardinalidad de configuraciones y una distancia pequeña entre configuraciones.

Es importante notar que un conjunto con la cardinalidad mínima puede no ser el óptimo para nuestro problema, ya que el costo en distancia puede ser grande para alcanzar las configuraciones de sensado si se encuentran alejadas unas de otras. Por lo cual, nuestro problema es diferente a aquel de elegir únicamente el mínimo número de configuraciones que cubran por completo al ambiente: el problema de la galería de arte. Dado que el problema de la galería de arte es NP-Duro [74], es dudoso que nuestro problema pueda ser resuelto de forma eficiente con un algoritmo determinista. Por lo tanto, decidimos usar un esquema de muestreo aleatorio para generar un conjunto de configuraciones de sensado Q_r, y después obtener un conjunto reducido $Q'_r \subseteq Q_r$. Al final, el conjunto de configuraciones de sensado Q'_r es el que utilizamos para cubrir la región convexa C_r.

A continuación describimos el algoritmo para obtener nuestro primer conjunto de configuraciones de sensado Q_r, (algoritmo 6.1). En cada paso, generamos aleatoriamente una configuración de sensado q, la cual es aceptada para formar parte del conjunto Q_r si ubica el efector final dentro de la región convexa C_r y si en la configuración q el robot está libre de colisión con los obstáculos del ambiente y consigo mismo. También debemos evitar oclusiones producidas por el cuerpo del robot por lo que rechazamos la configuración q, si con ella el frustum del sensor está en colisión con el robot. Una vez hecha esta validación, la función *nuevaMuestra(q)* se encarga de calcular la región de visibilidad del sensor como se describió en la sección anterior. Además, *nuevaMuestra(q)* se encarga de determinar si el robot está sensando una nueva porción de la región convexa, es decir, regresa un valor verdadero si detecta al menos una muestra que no haya sido cubierta con algún frustum de una configuración generada previamente. El algoritmo continuará generando configuraciones de sensado hasta que se hayan agotado M intentos de encontrar un punto muestra que no haya sido cubierto hasta ese momento, en otras palabras, intentará M veces ver una nueva porción de la región convexa C_r. El valor de M es elegido utilizando la cota descrita en la sección 4.1.1.

Algoritmo 6.1: Conjunto de configuraciones de sensado Q_r

Entrada: Región convexa C_r, ambiente 3D

Salida: Q_r, Conjunto de configuraciones de sensado aleatorias

```
1  begin
2      Q_r ⟵ ∅
3      M ⟵ log(α)/log(1 − ϵ)
4      nInt ⟵ 0
5      while (nInt ≤ M) do
6          generaConfiguracion(q)
7          nuevaPorcion ⟵ falso
8          if ( dentroC_r(q) ∩ noColision(q) ∩ noOclusion(q) ) then
9              agrega(Q_r,q)
10             nuevaPorcion ⟵ nuevaMuestra(q)
11         if ( nuevaPorcion ) then
12             nInt ⟵0
13         else
14             nInt ⟵ nInt + 1
```

Una vez hecho el muestreo del espacio de configuraciones, para generar el conjunto Q_r de configuraciones de sensado candidatas, seleccionamos un subconjunto $F \subseteq Q_r$ de baja cardinalidad. Nuestro algoritmo para seleccionar F está inspirado por el trabajo presentado en [17]. El trabajo esta enfocado a ambientes 2D, los cuales son representados como polígonos. Los autores requieren encontrar un buen conjunto de ubicaciones para obtener un sensado efectivo del ambiente. Más específicamente, buscan el conjunto mínimo de guardias dentro de un espacio de trabajo polígonal W, tal que todo el borde ∂W del espacio de trabajo sea visible a los guardias. La idea básica de su solución es usar muestreo aleatorio para transformar el problema en uno de cobertura de conjuntos o en su dual, el problema de intersección de conjuntos, (del inglés set cover problem y hitting-set problem, respectivamente). Para la instancia del problema de cobertura de conjuntos aplican un algoritmo voraz (greedy) y para el problema de intersección de conjuntos utilizan el algoritmo propuesto por Brönnimann y Goodrich [7]. Para nuestro problema elegimos el enfoque de cobertura de conjuntos junto con el algoritmo voraz, ya que desde el inicio hemos optado por un muestreo aleatorio sobre el espacio de trabajo W y esta técnica se adapta muy bien a nuestras necesidades como se mostrará a continuación.

Para sensar una región convexa C_r, primero enumeramos cada muestra $s \in S_r$ y las agrupamos en el conjunto $S_r = \{s_1, s_2, \ldots, s_l\}$ donde l es el número de muestras en la región C_r. Después construimos la colección de subconjuntos $T = \{T_1, T_2, \ldots, T_m\}$ donde $T_i = \{s \in S_r \mid q_i \textit{ cubre } s\}$ es el subconjunto de elementos de S_r que son cubiertos por la configuración de sensado $q_i \in Q_r$. Por construcción, cada elemento de S_r pertenece al menos a un conjunto de T, $S_r = \bigcup_{(T_i \in T)} T_i$. El sistema (S_r, T) es la instancia conseguida. Ahora el problema de encontrar un conjunto de configuraciones de sensado, usando un sensor limitado en rango de alcance mínimo y máximo y campo de vista, que cubran la región convexa C_r en un ambiente 3D puede tratarse a través de la instancia del problema de cobertura de conjuntos (S_r, T). El objetivo es encontrar un conjunto de tamaño mínimo $F \subseteq T$ cuyos elementos cubran todo S_r, $S_r = \bigcup_{(T_i \in F)} T_i$. El problema de cobertura de conjuntos es NP-duro, y ha sido ampliamente estudiado. El algoritmo voraz es preferido por su simplicidad y dar una solución aproximada logarítmica.

Veamos un ejemplo del algoritmo, supongamos que estamos sensando la región convexa C_r y que el conjunto S_r que cubre la región consta de 24 muestras. Tenemos la colección de conjuntos $T = \{T_1, T_2, T_3, T_4, T_5, T_6\}$ generados por las configuraciones en Q_r. En la figura 6.4 el conjunto mínimo que cubre las muestras es $F = \{T_3, T_4, T_5\}$. En cada paso, el algoritmo voraz eligirá el conjunto $T_i \in T$ que cubre el número más grande de elementos que aún no son cubiertos. Para el ejemplo, el algoritmo eligirá primero el conjunto T_1 que cubre 12 elementos. Entonces,

escogerá a T_4 que tendrá 5 elementos sin cubrir, después eligirá T_3 y T_5 los cuales tienen 4 y 3 elementos que no han sido cubiertos, respectivamente. En este punto, cada elemento de S_r estará cubierto. El algoritmo voraz dará la solución $F = \{T_1, T_4, T_3, T_5\}$.

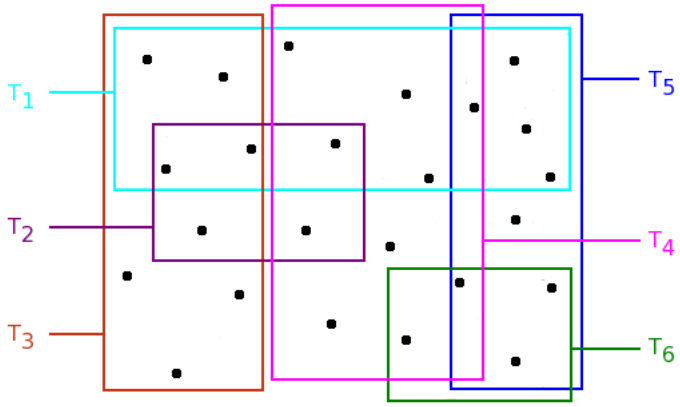


Figura 6.4: Una instancia del problema de cobertura de conjuntos, para la selección de configuraciones de sensado.

Algoritmo 6.2: Conjunto reducido de configuraciones de sensado. Algoritmo voraz.

Entrada: S_r, T
Salida: F

```
1 begin
2     X ⟵ S_r
3     F ⟵ ∅
4     while (X ≠ ∅) do
5         seleciona T_i ∈ T que maximiza |T_i ∩ X|
6         X ⟵ X\T_i
7         F ⟵ F ∪ {T_i}
```

El algoritmo 6.2 en cada paso simplemente elige el conjunto de T que cubra tantos elementos que aún no han sido cubiertos como sea posible, almacena el conjunto en F y remueve los elementos de X. Cuando todos los elementos de X son cubiertos, F tiene una subfamilia de T que cubre a S_r completamente y el algoritmo termina.

Similar a [7] seleccionamos un conjunto reducido de configuraciones de sensado con alta

cardinalidad de puntos muestras, eliminando aquellas que producen redundancia de muestras cubiertas.

El procedimiento de selección de configuraciones de sensado descrito anteriormente se repite varias veces para cada región convexa C_r, produciendo diversos conjuntos F. Uno de estos conjuntos será seleccionado, siguiendo el procedimiento descrito en la siguiente sección.

6.2.1. Conexión de configuraciones de sensado

Debido a que deseamos tener opciones de configuraciones de sensado entre las cuales mover al robot y además queremos reducir el valor esperado del tiempo para encontrar un objeto, conectamos las configuraciones de sensado de cada conjunto F en un grafo completamente conectado con ne aristas.

A cada arista le asignamos un peso $w_{(q_i,q_j)}$ de acuerdo a la métrica del costo del movimiento del robot entre las configuraciones de sensado q_i y q_j. Considerando el valor promedio de los pesos de las aristas del grafo,

$$E(F) = \frac{1}{ne} \sum_{q_i,q_j \in F} w_{(q_i,q_j)}$$

seleccionamos el conjunto que minimiza ese valor, es decir,

$$Q'_r = argmin_{_F}\ E(F)$$

Al final, obtenemos el conjunto Q'_r con ambas propiedades baja cardinalidad de configuraciones de sensado y distancias pequeñas entre las configuraciones.

Podemos usar el algoritmo de expansión de frente de onda descrito en el capítulo 5 para calcular el costo del movimiento del robot entre las configuraciones de sensado. Sin embargo, la ejecución del algoritmo requiere mucho tiempo cuando el número de configuraciones es grande. Para reducir el tiempo de planificación de caminos para cubrir la región convexa con el sensor limitado, consideramos que el costo de mover al robot entre las configuraciones de sensado corresponde a una línea recta en el espacio de configuraciones. En otras palabras, para reducir el tiempo de computo para cubrir el ambiente con el sensor limitado, estimamos el costo del movimiento del robot entre las configuraciones de sensado como un movimiento en línea recta en el espacio de configuraciones. Para cubrir la región convexa C_r el robot utiliza el conjunto de configuraciones de sensado en Q'_r; realiza un recorrido que comienza y termina en la misma

configuración de sensado, es decir, forma un ciclo simple cuando visita las configuraciones de sensado según el orden establecido. Que el robot viaje ciclos para resolver una tarea ha sido utilizado en otros trabajos, por ejemplo [65] y [64].

Por tanto, para lidiar con el problema de generar caminos libres de colisión entre las configuraciones usamos una técnica de examinación tardía de colisiones (en inglés, lazy collision checking) [66] y [67]. Debido a la cual, conectamos las configuraciones de sensado en un grafo sin probar si existe un camino libre de colisión entre ellas.

Dado que el robot visita una región convexa tras otra, cuando cubrimos una región dada es probable encontrar caminos lineales (en el espacio de configuraciones) libres de colisión entre las diversas configuraciones de sensado. Con frecuencia, una región pequeña requerirá poco movimiento del robot para cubrir la región y en una región grande el robot tendrá más espacio y tenderá a moverse más. Posponemos la prueba de colisión hasta que establecemos un orden de visita de las configuraciones. Si algún camino en la ruta causa colisión entonces generamos un nuevo suborden usando otras aristas en el grafo. La nueva subruta comienza en la última configuración conectada hasta el momento e incluye las configuraciones restantes. Evidentemente, algunas veces el grafo se desconecta dando lugar a dos componentes conectados; si eso pasa, usamos un planificador RRT [42] para encontrar un camino libre de colisión que conecte ambos componentes. En este último caso, el costo asignado a la arista del grafo, que liga a las configuraciones de sensado asociado a los componentes, se obtiene del costo del camino obtenido por el RRT. En nuestros experimentos hemos observado que esta estrategia reduce considerablemente el tiempo para obtener caminos libres de colisión entre las diversas configuraciones de sensado y así cubrir cada región convexa. El camino usado para llevar a cabo el orden de visita de las regiones convexas es de costo óptimo, este camino lo calculamos con el algoritmo de expansión de frente de onda descrito en 5.2.

La descomposición del espacio libre del ambiente 3D en regiones convexas es muy útil, incluso la usamos para trabajar con un robot equipado con un sensor limitado. Hay cuatro beneficios principales de la cobertura convexa: primero, evitamos usar estructuras de datos complejas para mantener actualizadas las porciones del ambiente que sensa el robot. Recordemos que una región convexa es sensada, si las muestras usadas para aproximar su tamaño son sensadas o equivalentemente, si la unión de los frustum asociados a las configuraciones de sensado cubren todas las muestras pertenecientes a la región convexa. El segundo beneficio es que podemos retardar la prueba de colisión hasta establecer un orden para las configuraciones de sensado. Al cubrir una región convexa, es posible encontrar un camino libre de colisión entre las configuraciones de sensado, simplemente moviendo al robot en línea recta en el espacio de

configuraciones. Tercero, cuando el robot está sensando el ambiente 3D no se mueve a otra región convexa hasta que termina de cubrir una región dada, esto evita desplazamientos innecesarios del robot para ir de una configuración de sensado a la siguiente según el orden programado. Creemos que nuestra estrategia de búsqueda es análoga a la forma de organizar el pintado de una casa, primero se elige el orden para pintar las habitaciones (establecer una búsqueda por regiones). Y después se va pintando pared por pared hasta terminar un cuarto para continuar con el siguiente, esto es equivalente a visitar las configuraciones de sensado, cuyos frustum asociados cubren por completo la región convexa. Por último, una vez que hemos generado un plan global si el ambiente cambia localmente, entonces podemos modificar localmente ese plan, tomando en consideración sólo aquellas regiones convexas que se ven afectadas por el cambio. Describiremos a detalle de este último beneficio en el capítulo siguiente.

En resumen, para nuestra estrategia de búsqueda de un objeto reduciendo el valor esperado del tiempo en encontrarlo, primero dividimos el ambiente en un conjunto de regiones convexas C_r, de tal manera que el interior del ambiente $int(W)$ este representado por las regiones convexas. En otras palabras, la unión de las regiones convexas cubren el espacio libre del ambiente 3D, $\bigcup_r C_r = int(W)$. En seguida, establecemos un orden de visita para las regiones convexas. Utilizamos un algoritmo de expansión de frente de onda para encontrar caminos de costo óptimo para mover al robot de una región a otra. Después, generamos aleatoriamente un conjunto Q de configuraciones de sensado para cubrir cada región convexa. El conjunto es filtrado para obtener un subconjunto reducido $Q' \subseteq Q$ de "buenas" configuraciones de sensado que colectivamente cubran toda la región. Las configuraciones de sensado de cada región son ligadas a un grafo y desarrollamos una búsqueda en él para establecer un orden de visita de las configuraciones. Para establecer ambos ordenes, el de visita de las regiones convexas y de las configuraciones de sensado, utilizamos una función de utilidad que nos indica que tan conveniente es ir a determinada región o configuración. La función de utilidad hace una evaluación del costo de mover al robot y de la ganancia de encontrar al objeto. Hacemos énfasis que nuestro algoritmo reduce la magnitud del tiempo computacional en comparación con el algoritmo necesario para encontrar una solución óptima, la cual es exponencial debido a que el problema de optimización es NP-duro.

6.3. Optimización de un número reducido de grados de libertad

Como lo hemos mencionado en los capítulos anteriores, los caminos para establecer un orden de visita de las regiones convexas son de costo óptimo. Para encontrar el camino de menor

costo entre una región y todas las demás usamos el algoritmo de expansión de frente de onda propuesto en 5.2. Note que el problema de encontrar un camino local de costo óptimo (entre dos configuraciones) para un robot con varios grados de libertad puede tomar un tiempo significativo, en particular, cuando requerimos un camino global que incluya un gran número de configuraciones de sensado intermedias.

De nuestras simulaciones hemos detectado que el cuello de botella para la estrategia de búsqueda corresponde al cálculo de los caminos óptimos cuando el número de grados de libertad es grande (4 o más). Mitigamos este problema, seleccionando sólo algunos g.d.l. para llevar a cabo el proceso de optimización y el resto de ellos los determinamos aleatoriamente.

En términos de energía y tiempo, para un robot manipulador móvil, es más costoso trasladar el sistema que rotar la base y las articulaciones del brazo. De hecho, si el planificador es capaz de coordinar la traslación de la base y las rotaciones de las articulaciones tal que ambos movimientos ocurran a la vez, entonces el costo de mover el brazo en términos del tiempo transcurrido es cero, ya que ambos movimientos, traslación y rotación se realizan simultáneamente. Por lo general, el ambiente donde el robot busca el objeto es amplio y tiene que trasladarse considerablemente. Por tanto, para nuestro sistema de siete grados de libertad, consideramos que las coordenadas (x, y) que definen la posición de la base del robot determinan el costo de mover al sistema completo, en consecuencia elegimos estos dos g.d.l para optimizar. Entonces, un camino para mover un robot entre las regiones convexas es una secuencia de configuraciones donde algunos g.d.l. son planificados óptimamente y los restantes se determinan aleatoriamente para que no produzcan colisión entre el robot y los obstáculos.

Capítulo 7

Reparación de planes de búsqueda de un objeto en ambientes 3D

En los capítulos anteriores describimos nuestra estrategia de búsqueda de un objeto estático ubicado en algún lugar de un ambiente 3D. Hemos trabajado con un robot manipulador móvil equipado con un sensor limitado en rango de alcance máximo y mínimo, así como en el campo de vista. Ahora, trabajaremos el caso cuando hemos generado un plan global de búsqueda y debido a un cambio en el ambiente tenemos la necesidad de replanificar la estrategia de búsqueda.

7.1. Reparación de un plan global de búsqueda

En general el algoritmo propuesto para la búsqueda recibe como entrada un mapa 3D del ambiente y regresa un plan de búsqueda. Una vez que ese plan es generado lo podemos usar varias veces para encontrar al objeto, mientras el ambiente no sufra alguna modificación. Le llamamos plan global al conjunto de regiones convexas y configuraciones de sensado, junto a sus respectivos ordenes de visita y los caminos generados para seguir ambos ordenes.

El hecho de que el costo de planificación es amortizado a través de varias instancias de un problema da una justificación de emplear tiempo en la generación del plan. En nuestro escenario cada instancia del problema corresponde a una ubicación diferente del objeto que se está buscando. Sin embargo, cuando se altera el ambiente, el plan también debe ser modificado.

La motivación de elaborar una técnica que repare un plan de búsqueda ya generado, viene de la siguiente observación, cuando el ambiente cambia sólo localmente, el plan se puede

reparar localmente, por lo que no hay necesidad de desechar por completo el plan. Por ejemplo, imaginemos el siguiente escenario: necesitamos encontrar un objeto dentro de una casa y hemos generado un plan para encontrarlo, pero después de elaborar el plan algunos muebles de la casa cambian de lugar, son movidos de una habitación a otra o dentro de la misma habitación, y el resto de los muebles y la estructura de la casa permanecen intactos. Entonces, si necesitamos buscar nuevamente el objeto podemos seguir el plan que ya teníamos en la parte de la casa que no se ha alterado y adecuar el plan localmente para los lugares donde sufrió un cambio.

Creemos que en este tipo de situaciones, donde los cambios a un ambiente son locales y no son tan grandes o tan drásticos que se perciba como un nuevo ambiente, nuestro método es útil y aplicable. Por tanto, el método que proponemos reparará un plan calculado previamente para lidiar con los cambios locales en un ambiente 3D. Basamos este procedimiento en la descomposición en regiones convexas. Básicamente, reparamos el plan generando un nuevo subconjunto de configuraciones de sensado y un nuevo orden para visitarlas, sólo considerando las regiones convexas relacionadas con la modificación del mapa del ambiente 3D.

La ventaja principal de reparar un plan en vez de generar uno nuevo es que el tiempo necesario para la reparación es, en general, significativamente más pequeño que el tiempo necesario para generar un plan completo otra vez. Esto es más evidente cuando el ambiente 3D es grande y complejo. El tiempo empleado en la reparación depende del porcentaje del ambiente que ha cambiado.

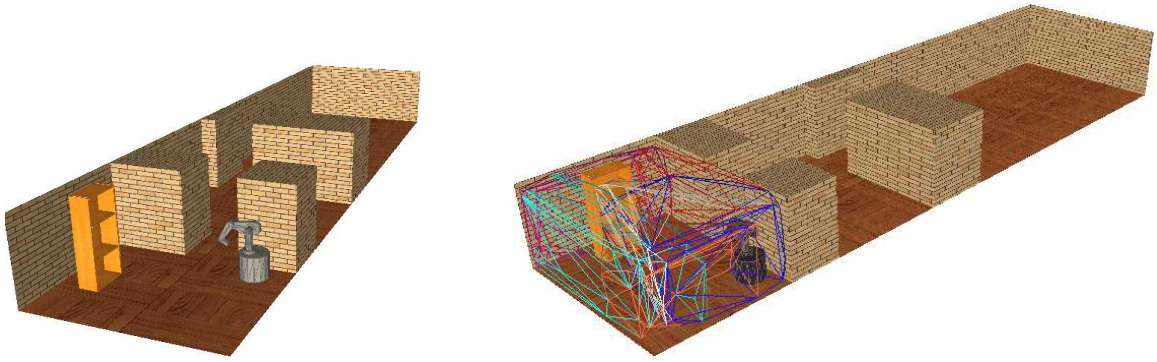
7.1.1. Modificación de la cobertura convexa debido a un cambio en el ambiente 3D

Recordemos, que para construir la cobertura convexa generamos aleatoriamente muestras en el interior del ambiente 3D, ver imagen 4.5. Un subconjunto S_{lib} de muestras caen en el espacio libre del ambiente y el resto caen donde se ubican los obstáculos, el subconjunto S_{obst}. Estas últimas muestras las reservamos y tomamos en cuenta hasta el momento en que se efectúa un cambio en el ambiente. Del conjunto S_{lib} elegimos un subgrupo G de muestras que actúan como guardias ocultos (del problema de la galería de arte). Los miembros de G, colectivamente ven todas las demás muestras en S_{lib}, con la condición de que cada guardia $g \in G$ no puede ver al resto de los guardias, es decir, g está oculto a los otros miembros del conjunto G. Cada guardia g tiene un subconjunto de muestras en S_{lib} que sólo él puede ver, llamamos a este conjunto el kernel del guardia, ver figura 4.6. Usamos las muestras del kernel para construir un envolvente convexo alrededor de cada guardia g. Después expandimos el envolvente agregando muestras

de S_{lib} tanto como sea posible, sin intersectar con los obstáculos y preservando la convexidad, hasta construir una región convexa. De esta manera, construimos una colección C de regiones convexas que dividen el espacio libre del ambiente 3D, imagen 4.7.

Detectamos un cambio en el ambiente a través de la cobertura convexa original C. Esto es, un cambio de ubicación de un obstáculo en el ambiente produce las siguientes modificaciones en la cobertura convexa C.

- Hay un subconjunto de regiones convexas $\{C_i\}_t$ con $C_i \in C$ relacionado con la *ubicación original del obstáculo* (figura 7.1), dichas regiones tienen que ser modificadas.
- Hay un subconjunto de regiones convexas $\{C_i\}_{t+1}$ con $C_i \in C$ relacionado con la *nueva posición del obstáculo* (figura 7.2), estás regiones también deben ser modificadas.



(a) Posición inicial del librero

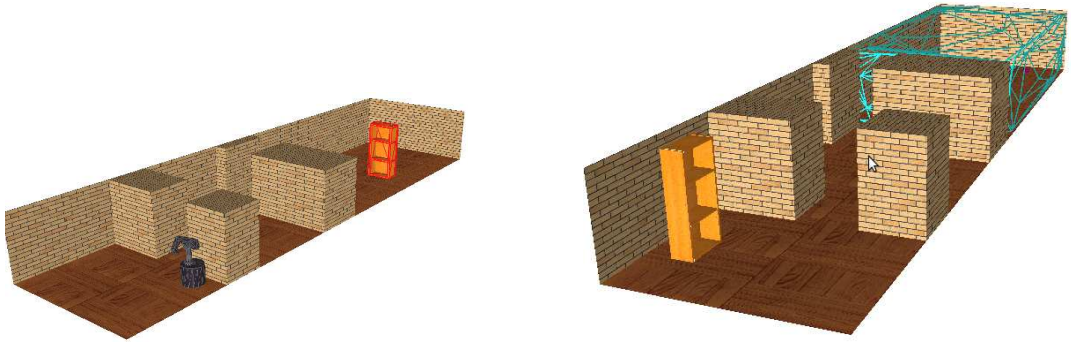
(b) Regiones circundantes, $\{C_i\}_t$

Figura 7.1: Regiones asociadas a la posición inicial del obstáculo

Para determinar cuales regiones son miembros de $\{C_i\}_t$ es necesario detectar las regiones que son adyacentes al obstáculo en su posición original. Un procedimiento (entre otros) para llevar a cabo esto, es medir la distancia entre las regiones convexas y el obstáculo. Todas las regiones que estén más cerca que una distancia dada ϵ del obstáculo serán miembros de $\{C_i\}_t$.

Y para definir los elementos de $\{C_i\}_{t+1}$, lo podemos lograr de manera sencilla. Sólo examinamos si existe colisión del obstáculo en su nueva ubicación con cada región convexa, si una región choca con el obstáculo entonces formará parte del conjunto $\{C_i\}_{t+1}$.

Tanto el conjunto de regiones convexas $\{C_i\}_t$ como $\{C_i\}_{t+1}$ tienen asociado un grupo de muestras. Vamos a denotar a S_t como el conjunto de muestras de S_{lib} que están en las regiones convexas miembros de $\{C_i\}_t$, $S_t = \{s \in S_{lib} | \ s \ está \ en \ \{C_i\}_t\}$, figura 7.3(a). Análogamente,



(a) Posición final del librero

(b) Regiones afectadas, $\{C_i\}_{t+1}$

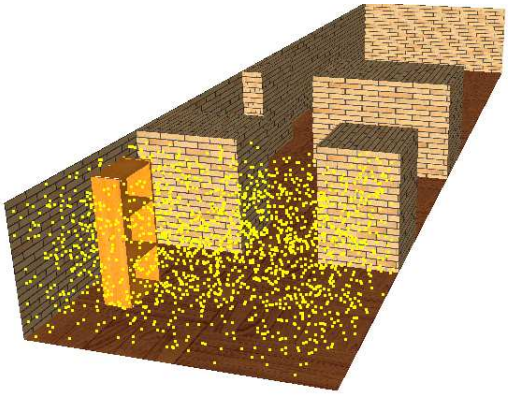
Figura 7.2: Regiones asociadas a la posición final del obstáculo

denotamos como S_{t+1} al conjunto de muestras en S_{lib} que están en las regiones convexas de $\{C_i\}_{t+1}$, $S_{t+1} = \{s \in S_{lib} |\ s\ está\ en\ \{C_i\}_{t+1}\}$, figura 7.3(b). También requerimos determinar las muestras que caen en el espacio ocupado por el obstáculo en su original y nueva posición, figura 7.4. Vamos a llamar como So_t y So_{t+1} al conjunto de muestras de S_{obst} que están en colisión con el obstáculo en la ubicación original y en la nueva posición, respectivamente.

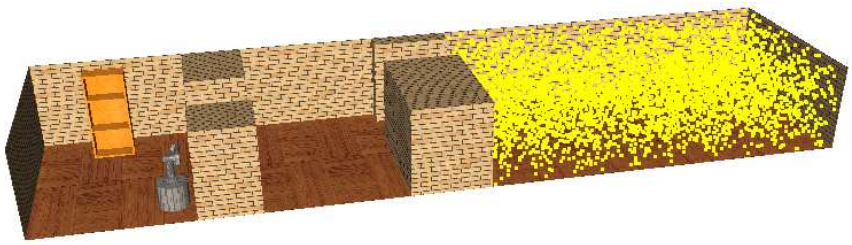
La idea clave para la reparación del plan es tomar en cuenta únicamente el espacio del ambiente 3D circundante al obstáculo que cambia de posición. Utilizamos la cobertura convexa para aproximar este espacio. Las muestras asociadas a las regiones convexas en la vecindad del obstáculo en la nueva y antigua posición; así como las muestras que se encuentran en el espacio ocupado por el obstáculo antes y después del cambio, representarán (aproximarán) el espacio del ambiente que debe considerarse para la reparación del plan. Vamos a denotar con S_Δ a este conjunto de muestras. Entonces, S_Δ es la unión de las muestras en S_t, relacionadas al conjunto de regiones convexas en $\{C_i\}_t$, con las muestras en So_t, asociadas al espacio ocupado por el obstáculo en la posición original; con la unión de las muestras en S_{t+1}, asociadas con $\{C_i\}_{t+1}$, menos las muestras en S_{t+1}, que están en colisión con el obstáculo en la nueva posición; $S_\Delta = (S_t \cup So_t) \cup (S_{t+1} \backslash So_{t+1})$.

La representación poliédrica del ambiente que toma en cuenta la nueva posición del obstáculo y el conjunto de muestras S_Δ son los datos de entrada del algoritmo para la construcción de las regiones convexas, descrito en el capítulo 4. El algoritmo regresa un nuevo conjunto de regiones convexas que dividen el espacio del ambiente relacionado al cambio. Denotamos a este conjunto como C_Δ.

De la cobertura convexa original puede existir un conjunto de regiones convexas $\{C_i\}_{t_0}$ que



(a) Muestras relacionadas a la posición inicial, S_t



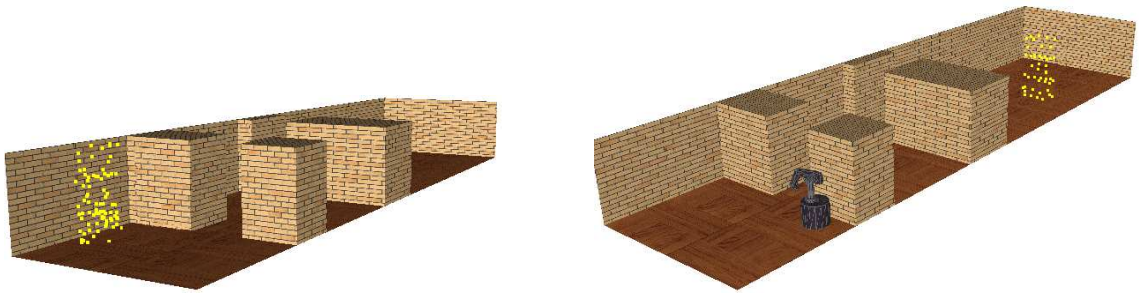
(b) Muestras en las regiones de la posición final, S_{t+1}

Figura 7.3: Muestras asociadas a las regiones convexas a modificarse

permanecen intactas después del cambio de posición del obstáculo. Entonces, la colección C de regiones convexas original está dividida en las regiones que cambian y las que se mantienen intactas, $C = \{C_i\}_t \cup \{C_i\}_{t+1} \cup \{C_i\}_{t_0}$. Y la nueva cobertura convexa N generada por el cambio es la unión de las regiones convexas que permanecen sin modificación con las nuevas regiones convexas, $N = \{C_i\}_{t_0} \cup C_\Delta$. Las regiones convexas en N representan el espacio libre del ambiente 3D considerando al obstáculo en su nueva posición.

7.1.2. Modificación del orden de visita de las regiones convexas y configuraciones de sensado

Como se mencionó anteriormente, el problema de generar un orden de visita de las regiones convexas y de las configuraciones de sensado que colectivamente cubren el ambiente 3D, fue resuelto en dos fases distintas. Primero ordenamos las regiones convexas, auxiliándonos en el



(a) Muestras en la posición inicial, So_t

(b) Muestras en la posición final, So_{t+1}

Figura 7.4: Muestras que caen en el espacio ocupado por el obstáculo

algoritmo de expansión de frente de onda y la función de utilidad. Y segundo, para cada región convexa, establecemos un orden de visita de las configuraciones de sensado asociadas a la región, apoyándonos en un grafo completamente conectado y la función de utilidad. Originalmente esta estrategia la propusimos para reducir el tiempo de cómputo al generar una ruta de búsqueda para el robot. Ahora, vamos a ver cómo esta estrategia de dividir el problema en pequeños subproblemas, también nos facilita el trabajo de reparar un plan previamente calculado.

Tenemos varias opciones para reparar un plan global de búsqueda calculado previamente, por ejemplo, podríamos únicamente calcular el orden de visita de las regiones convexas en N y preservar tanto como sea posible el orden de visita de las configuraciones de sensado en cada $N_r \in N$. Sin embargo, bajo la suposición de que el cambio en el ambiente es local, tiene sentido modificar el plan global sólo localmente, preservando también, tanto como sea posible, el orden de visita de las regiones convexas que no fueron modificadas. Este enfoque tiene una ventaja, que usualmente el tiempo de ejecución para reparar el plan es más pequeño en referencia al tiempo para calcular nuevamente el plan global completo, agregando reactividad al proceso de planeación.

Para reparar el plan, tomamos en cuenta la ubicación espacial de las regiones convexas originales que sufren cambio y las nuevas regiones. Ya que las nuevas regiones convexas en C_Δ aparecerán en los espacios del ambiente ocupados por las regiones pertenecientes a $\{C_i\}_t \cup \{C_i\}_{t+1}$. El objetivo del algoritmo 7.1 es calcular un nuevo orden para la cobertura convexa N, preservando tanto como sea posible el plan global original.

Vamos a llamar O_C al conjunto de regiones convexas originales ordenas. Es decir, O_C es el conjunto C ordenado según el algoritmo del capítulo 3. Y denotamos como O_N al conjunto ordenado de todas las regiones convexas después del cambio. Vamos a tratar de diferenciar las regiones convexas de la colección original C de la nueva colección N. C_{r_j} denota la región

convexa r en el lugar $j-ésimo$ de acuerdo al orden del conjunto O_C; y N_{t_k} denota la región convexa t, elemento del conjunto ordenado O_N, en el lugar $k-ésimo$ del orden. Por último C_v representa una región convexa recién creada, $C_v \in C_\Delta$.

El algoritmo 7.1 recibe como entrada las regiones convexas originales asociadas al espacio que se modificó, es decir, los conjuntos $\{C_i\}_t$ y $\{C_i\}_{t+1}$; las regiones convexas que se originaron tras la modificación, el conjunto C_Δ y las regiones convexas originales ordenadas, O_C. Escencialmente, cada región convexa $C_{r\_j} \in \{C_i\}_t \cup \{C_i\}_{t+1}$ se elimina del nuevo plan, mientras se está determinando el nuevo orden para N. Cada región convexa $C_{r\_j} \in (\{C_i\}_t \cup \{C_i\}_{t+1})$ se verifica para determinar con que región convexa C_v recién creada colisiona. Una región C_{r_j} puede estar en colisión con más de una región C_v; cada región C_v que colisiona con C_{r_j} se almacenan en el conjunto auxiliar $\{C\}_{aux}$, línea 7.

El método *Orden_Local*, línea 16, se usa para establecer un orden de visita para las regiones en $\{C\}_{aux}$; para después agregarlas al conjunto de regiones convexas O_N. *Orden_Local* trabaja similar al algoritmo del capítulo 3. Suponemos que el robot se ubica en la región convexa N_{t_k}, la última hasta el momento según el orden en O_N. Evaluamos la función de utilidad un paso adelante sobre las regiones en $\{C\}_{aux}$ y obtenemos el siguiente elemento para O_N, es decir, la región N_{t_k+1}, la cual máximiza la función de utilidad. Después, suponiendo que el robot está en la región N_{t_k+1} evaluamos nuevamente la función de utilidad en las restantes $n-1$ regiones de $\{C\}_{aux}$, y así obtenemos la siguiente región en el orden, la región N_{t_k+2}. Continuamos de esta manera hasta que todas las regiones en $\{C\}_{aux}$ estén ordenadas.

Por último, si la región C_{r_j} de la cobertura convexa original queda fuera del cambio, es decir, si $C_{r\_j} \notin \{C_i\}_t \cup \{C_i\}_{t+1}$, entonces la región C_{r_j} está en $\{C_i\}_{t_0}$; por lo que se agrega al conjunto O_N conservando la secuencia del orden del plan original, líneas 20 - 23. Hacemos notar que cada nueva región $C_v \in C_\Delta$ se incluye sólo una vez en el nuevo plan. La ubicación de la primera región C_{r_j} que intersecta con la región C_v establece el orden de la región C_v, una vez que se incluye en el nuevo plan, se elimina del conjunto C_Δ, (línea 13). Esto evita redundancia en el nuevo plan. El nuevo orden, O_N, se completa cuando todas las regiones originales son consideradas.

Para mover al robot entre las nuevas regiones convexas y entre una región anterior, que permanece en el nuevo plan, calculamos caminos libres de colisión con el algoritmo de expansión de frente de onda descrito en 5.2. Estos caminos son una secuencia de configuraciones de sensado donde los grados de libertad de traslación (x, y) son planificados optimamente y el resto los determinamos de tal manera que no produzcan colisión con el robot y con los obstáculos, como lo describimos en la sección 6.3. También usamos esta estrategia para calcular los caminos entre

Algoritmo 7.1: Nuevo orden de visita de las regiones en N

Entrada: $\{C_i\}_t$, $\{C_i\}_{t+1}$, $\{C_i\}_{t_0}$, O_C, C_Δ

Salida: O_N

```
1  begin
2      k ⟵ 0
3      n ⟵ 0
4      for (j = 1 to |O_C|) do
5          C_r_j ⟵ {O_C}
6          if (C_r_j ∈ ({C_i}_t ∪ {C_i}_{t+1})) then
7              {C}_aux ⟵ ∅
8              for ( i = 1 to |C_Δ|) do
9                  C_v ⟵ {C_Δ}
10                 if ( C_r_j ∩ C_v ≠ ∅) then
11                     agrega(C_v, {C}_aux)
12                     n ⟵ n + 1
13             C_Δ ⟵ C_Δ \ {C}_aux
14             while (n > 0 ) do
15                 k ⟵ k + 1
16                 N_t_k ⟵ Orden_Local({C}_aux , n)
17                 agrega(N_t_k, O_N)
18                 n ⟵ n − 1
19         else
20             if (C_r_j ∈ {C_i}_{t_0}) then
21                 k ⟵ k + 1
22                 N_t_k ⟵ C_r_j
23                 agrega(N_t_k, O_N)
```

las regiones convexas del plan original.

Para terminar con la reparación del plan global necesitamos generar nuevos planes locales para sensar las nuevas regiones convexas. Esto lo hacemos siguiendo la estrategia descrita en el capítulo 6. Sólo hay que tener en cuenta lo siguiente, mientras sensamos la región C_r con el sensor limitado, respetamos la definición 6.1 que determina la región de visibilidad del sensor limitado. De acuerdo a la definición 6.1 las muestras s_p pertenecientes a otra región diferente a C_r son marcadas como sensadas y lógicamente removidas del conjunto S_{lib}. Entonces, no sólo necesitamos planes para cubrir las nuevas regiones en C_Δ, sino también para aquellas regiones en $\{C_i\}_{t_0}$ del plan original que intersectaban regiones que fueron eliminadas y que estaban antes en el orden original.

En el próximo capítulo mostraremos los resultados de nuestra estrategia de reparación de planes globales para diversos ambientes 3D, en ellos resaltaremos el atributo principal de esta estrategia: si el ambiente cambia localmente nuestro plan también puede ser reparado sólo localmente.

Capítulo 8

Resultados

Los resultados presentados en este capítulo tienen la finalidad de mostrar el desempeño de las estrategias propuestas para la búsqueda de un objeto en ambientes 3D. Nuestros resultados se muestran en simulación. Utilizamos una computadora convencional, bajo el sistema operativo linux, con procesador pentium IV y 1 GB de memoria RAM.

8.1. Ambientes 3D

Esta sección describe los ambientes 3D que serán utilizados en las siguientes secciones. Los ambientes fueron diseñados para que no puedan proyectarse en un plano en el cual pudiera utilizarse algoritmos de búsqueda orientados a 2D, note las construcciones y muebles incluidos en cada ambiente.

8.1.1. Ambiente 1: Cuarto simple

El primer ambiente, el más simple de todos, es un cuarto con una mesa rectangular. La figura 8.1 presenta la mesa situada al centro del cuarto con una notable rotación en el plano XY. Dicho ambiente produce el menor número de regiones en comparación del resto de los ambientes utilizados en los diversos experimentos. Las dimensiones del ambiente son de 20 unidades de largo (eje x), 20 unidades de ancho (eje y) y 10 unidades de altura (eje z).

Figura 8.1: Ambiente 1: Cuarto simple

8.1.2. Ambiente 2: Dormitorio

Este ambiente pretende simular un dormitorio sencillo. El ambiente 2 es más complejo que el anterior ya que incluye 4 obstáculos que representan una cama, un buro, una mesa y una silla. Los obstáculos incluidos en el ambiente y sus detalles pueden observarse en la figura 8.2.

La relativa simplicidad del ambiente nos auxiliará en la introducción de las metodologías descritas a lo largo de este trabajo. Las dimensiones del dormitorio son de 20 unidades de largo, 20 unidades de ancho y 10 unidades de altura.

Figura 8.2: Ambiente 2: Dormitorio

8.1.3. Ambiente 3: Oficina

El ambiente presentado en la figura 8.3 simula una oficina o cuarto de estudio con una división parcial del espacio. Las dimensiones del ambiente son de 26 unidades de largo, 20 unidades de ancho y 10 unidades de altura.

Aunque las dimensiones del ambiente son casi comparables con los ambientes 1 y 2, tiene mayor complejidad derivada de los obstáculos. Éste ambiente incluye un librero de 3 repisas, 1 mesa redonda, 1 mesa rectangular y 1 silla. El muro de división posee una ventana o hueco que comunica ambos lados del cuarto. Los obstáculos de este ambiente serán removidos parcial y totalmente con la finalidad de comparar la complejidad y tiempos de ejecución de la metodología propuesta.

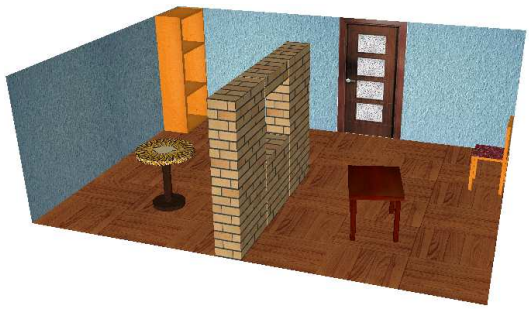


Figura 8.3: Ambiente 3: Oficina

8.1.4. Ambiente 4: Tres cuartos

El ambiente esta conformado por 3 cuartos consecutivos comunicados a través de 2 pequeños pasillos. En la figura 8.4 se observa un librero con 3 repisas situado en el primer cuarto.

Este ambiente fue utilizado en el capítulo 7 para ilustrar nuestra propuesta de reparación en los planes de búsqueda. Las dimensiones del ambiente son de 80 unidades de largo, 20 unidades de ancho y 10 unidades de altura.

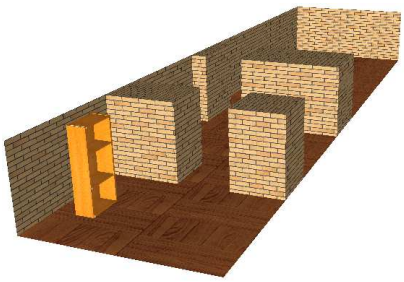


Figura 8.4: Ambiente 4: Ambiente con tres cuartos

8.1.5. Ambiente 5: Bar

En este ambiente 3D queremos representar un bar. Por un lado tenemos una barra donde se ubican las bebidas y al otro extremo un cuarto para clientes. La figura 8.5 muestra una ventana como único medio de comunicación al cuarto de clientes. El ambiente 4 incluye dos obstáculos que representan mesas para clientes. Las dimensiones del ambiente son de 40 unidades de largo, 30 unidades de ancho y 10 unidades de altura.

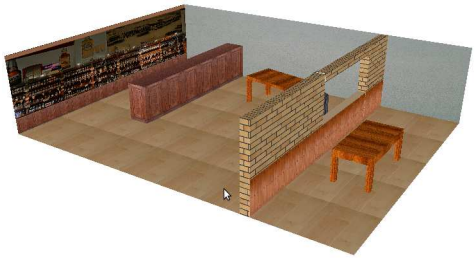


Figura 8.5: Ambiente 5: Analogía de un bar

8.1.6. Ambiente 6: Planta

Por último, presentamos el ambiente de mayor tamaño con respecto a los ambientes anteriores. El objetivo es simular la complejidad de una planta con varios cuartos en un edificio. Las dimensiones del ambiente son de 80 unidades de largo, 50 unidades de ancho y 10 unidades de altura. En la figura 8.6 podemos observar 6 cuartos, 2 pasillos transversales y 1 escalera suponiendo el acceso a una planta alta. Además, el ambiente está poblado con algunos muebles.

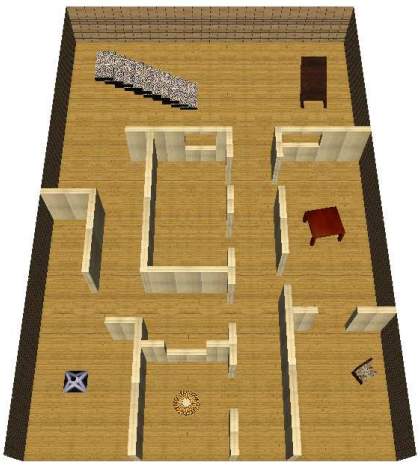


Figura 8.6: Planta

8.2. Construcción de regiones convexas

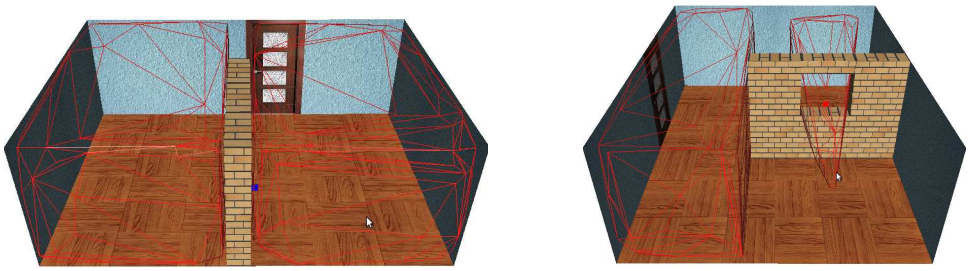
En esta sección describiremos las características observadas en el proceso de construcción de las regiones convexas. Utilizaremos el ambiente 3, oficina, descrito en la sección anterior, figura 8.3.

Para este análisis se realizaron 3 tipos de experimentos en la oficina: 1) sin muebles, 2) con librero y mesa rectangular y 3) con los 4 muebles de la figura 8.3. Para cada experimento se realizaron 30 corridas. Las figuras 8.7(a), 8.7(b) y 8.7(c) muestran, respectivamente, algunas regiones convexas del conjunto C para los experimentos 1, 2 y 3. Las regiones se generaron a partir de 21,000 muestras aleatorias con distribución uniforme para cada uno de los 3 experimentos. Este número constante de muestras lo utilizamos a fin de medir el tiempo de cálculo en la generación de regiones convexas en asociada con la complejidad del ambiente, establecida en función del número de obstáculos (muebles en el ambiente).

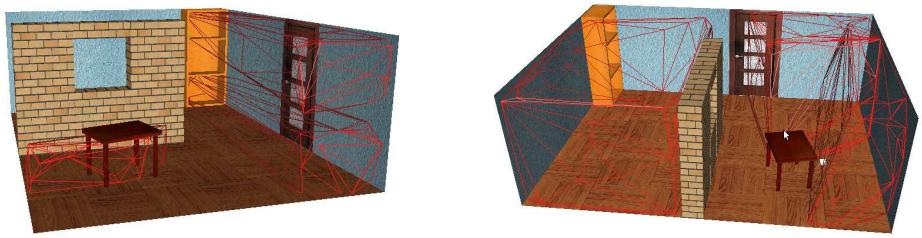
La figura 8.8(a) presenta un gráfico de barras del número de regiones convexas construidas por el algoritmo en cada una de las simulaciones. En color rojo, verde y azul, se presentan los resultados de los experimentos 1), 2) y 3), respectivamente. Se puede observar que para el experimento 1) se construyeron con mayor frecuencia 4 regiones, el cual es el número óptimo para este ambiente. Para el experimento 2) se crearon más frecuentemente 10 y 11 regiones convexas. Por último, cuando el ambiente tiene todos los obstáculos se construyeron en su mayoría entre 20 y 23 regiones.

La figura 8.8(b) muestra la relación entre los tiempos de construcción y el número de regiones para cada conjunto de regiones convexas. La correlación estimada entre el tiempo de construcción y el número de regiones es de 0.98. En la figura 8.8(b), se pueden distinguir fácilmente 3 clusters correspondientes a los 3 tipos de experimentos planteados para este análisis. El tiempo promedio de ejecución del algoritmo, para las 30 corridas, en cada tipo de experimento fue de 16.85, 121.35 y 186.61 segundos con una desviación estándar de 2.39, 16.93 y 10.38 segundos, respectivamente.

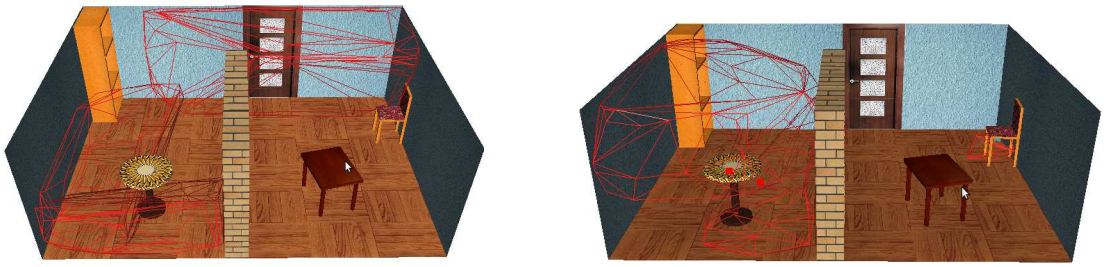
Partiendo del hecho de que las simulaciones fueron realizadas en el mismo ambiente y con el mismo número de muestras, podemos tomar el tiempo de ejecución para evidenciar lo comentado anteriormente sobre la complejidad del ambiente; decimos que nuestros ambientes son más complejos cuando el número de obstáculos se incrementa y por tanto tenemos un número mayor de regiones convexas.



(a) Experimento (1)



(b) Experimento (2)

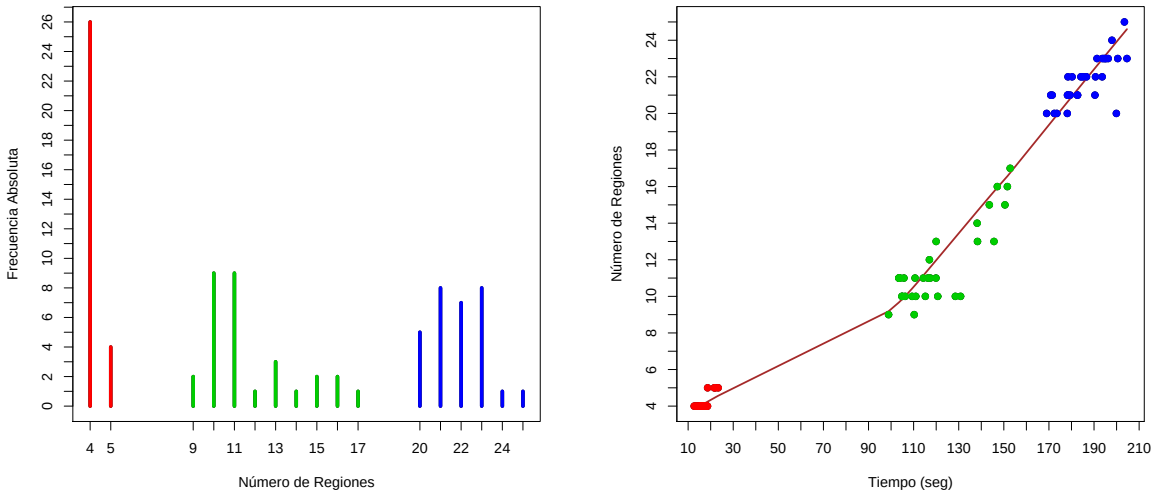


(c) Experimento (3)

Figura 8.7: Ejemplos de regiones convexas en el ambiente 3, oficina

8.3. Trayectorias dependientes de la métrica

En esta sección, empezaremos con los resultados que nos reporta el algoritmo de expansión de frente de onda del capítulo 5, donde las trayectorias dependen de los diferentes costos asignados a los grados de libertad del sistema. Para evidenciar los resultados dividimos los g.d.l. en dos grupos: 1) los cinco g.d.l. relacionados con la rotación de la base y de las articulaciones del brazo y 2) los dos grados relacionados con la traslación del sistema. Primero, asignaremos un menor costo a un grupo que al otro, por ejemplo, el grupo de rotación tendrá un menor costo que el grupo de traslación. Y después será al contrario, es decir, el grupo de traslación tendrá menor costo que el de rotación.



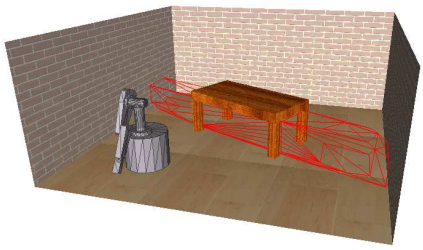
(a) Número de regiones

(b) Tiempos de construcción vs número de regiones

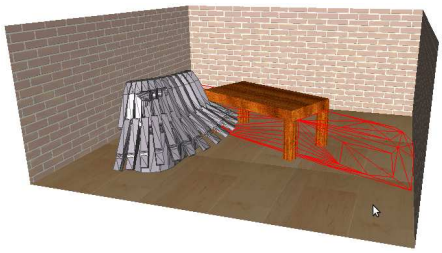
Figura 8.8: Construcción de regiones convexas en el ambiente 3, (30 corridas para cada experimento)

Utilizamos el ambiente 1, cuarto simple, para resaltar las diferencias en las trayectorias según el costo asignado a los grupos de grados de libertad. En la figura 8.9 podemos observar el comportamiento del sistema para alcanzar la región debajo de la mesa. En la figura 8.9(a) el robot se encuentra en la configuración inicial a una distancia no muy lejana de la mesa, en 8.9(b) vemos el camino encontrado cuando los g.d.l relacionados con la rotación tienen un menor costo y en la figura 8.9(d) los g.d.l. para la traslación tienen menor costo, consiguiendo un camino muy diferente. A pesar de que la tarea es la misma, colocar el efector final en la región convexa debajo de la mesa, ver figuras 8.9(c) y 8.9(e), los caminos encontrados difieren. Cuando asignamos menor costo a los grados de rotación el sistema principalmente rota, ver figura 8.9(b) y cuando asignamos menor costo a los grados de traslación el robot sólo se traslada, sin rotar ni la base ni el brazo, hasta alcanzar un extremo de la región, ver figura 8.9(d).

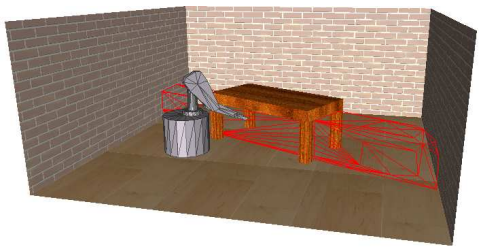
En el siguiente experimento utilizaremos el ambiente del bar, presentado en la figura 8.5. Queremos alcanzar las regiones convexas detrás de la barra y en el cuarto, partiendo de la posición inicial del robot que se muestra en la imagen 8.10. Primero analizaremos cómo alcanzar la región detrás de la barra, las figuras 8.11 y 8.12 muestran el comportamiento del



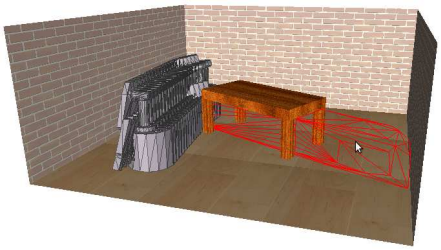
(a) Configuración inicial



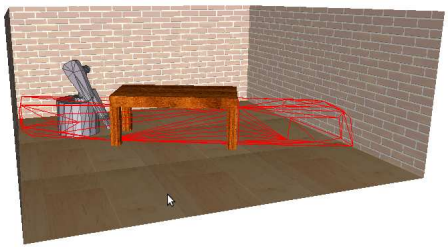
(b) Menor costo a los grados de rotación



(c) Posición final. Rotación



(d) Menor costo a los grados de traslación



(e) Posición final. Traslación

Figura 8.9: Trayectorias diferentes para alcanzar la región convexa debajo de la mesa

robot cuando asignamos diferentes costos a los g.d.l. En la figura 8.11(a) vemos la trayectoria conseguida cuando los g.d.l. de rotación tienen un menor costo, en la figura 8.11(b) se muestra la configuración final que alcanza la región. La figura 8.12 muestra la trayectoria y la configuración final para alcanzar la región convexa cuando los g.l.d. de traslación tienen un menor costo. En este ejemplo, al igual que en el ejemplo de la mesa, el robot tiene flexibilidad en el movimiento y las trayectorias resultantes son muy diferentes. Cuando es menos costoso rotar, el robot gira la base y las diferentes articulaciones del brazo para elevar el efector final por encima de la barra y se traslada sólo lo indespensable, en línea recta, para alcanzar la región. En contraste, cuando se prefiere los g.d.l. de traslación la mayoría de los movimientos son de este tipo; como se puede ver en la imagen 8.12(c), el robot no rota ninguna articulación del brazo, únicamente gira lo necesario la base para hacer que el efector final quede dentro de la región convexa; en la trayectoria la mayoría de los movimientos son de traslación, con el fin de ir al extremo de la

barra para alcanzar la región.

Ahora, veamos como se comporta el robot para alcanzar la región en el cuarto. En la imagen 8.10 podemos ver que la única forma de alcanzar la región convexa es a través de la ventana o hueco en la pared que divide el ambiente. En este caso, el robot no tiene la misma flexibilidad de movimiento para alcanzar su meta como en los ejemplos anteriores. En las figuras 8.13(a) y 8.14(a) vemos trayectorias muy similares que meten el efector final por la ventana para alcanzar la región convexa. En la figura 8.13 la trayectoria se produce al aplicar menor costo a los grados de rotación a diferencia de la trayectoria de la figura 8.14, donde se les asigno un mayor costo. Sin embargo, viendo las trayectorias desde otra perspectiva notamos una sutil diferencia entre ellas, en las figuras 8.13(b) y 8.13(c) mostramos un acercamiento de la trayectoria y de la configuración final, en ellas vemos como el brazo del robot termina estirado cuando damos menor costo a los g.d.l. de rotación, lo cual significa que hubo una mayor rotación de las articulaciones; mientras que en las figuras 8.14(b) y 8.14(c) vemos que el brazo del robot termina doblado al alcanzar la región, lo cual refleja que efectivamente los grados de rotación tienen un maayor costo.

Los ejemplos presentados en esta sección ilustran el comportamiento de nuestro robot en relación al costo que se define para cada grupo de g.d.l.: rotación y traslación. En las siguientes secciones, el objetivo será optimizar los g.d.l. en distancia Manhatan (distancia L_1), por lo que utilizaremos el mismo costo para todos los g.d.l.

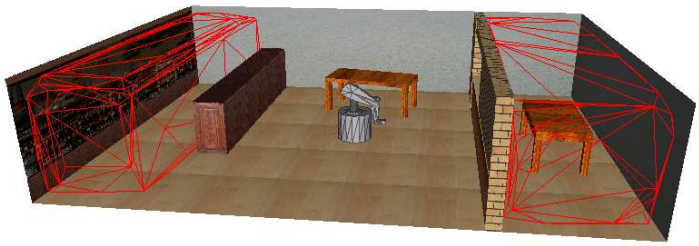
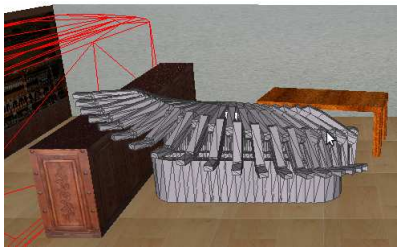


Figura 8.10: Configuración inicial en un ambiente 3D tipo bar. Dos regiones convexas

(a) Trayectoria a la región convexa

(b) Posición final

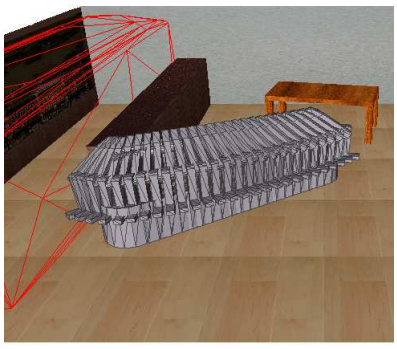


(c) Acercamiento de la trayectoria

Figura 8.11: Menor costo a los grados de rotación

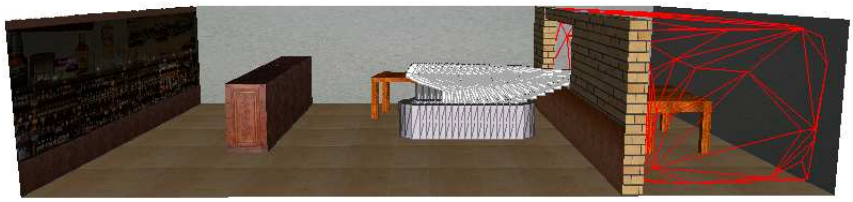
(a) Trayectoria a la región convexa

(b) Posición final



(c) Acercamiento de la trayectoria

Figura 8.12: Menor costo a los grados de traslación. Región convexa detrás de la barra

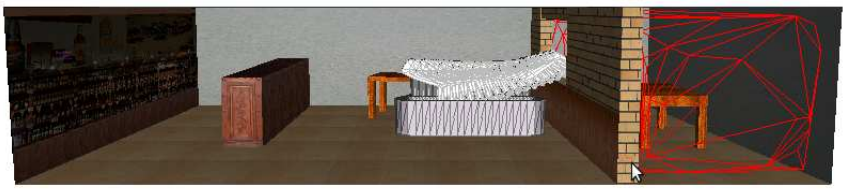


(a) Trayectoria a la región convexa

(b) Acercamiento de la trayectoria

(c) Posición final

Figura 8.13: Menor costo a los grados de rotación



(a) Trayectoria a la región convexa

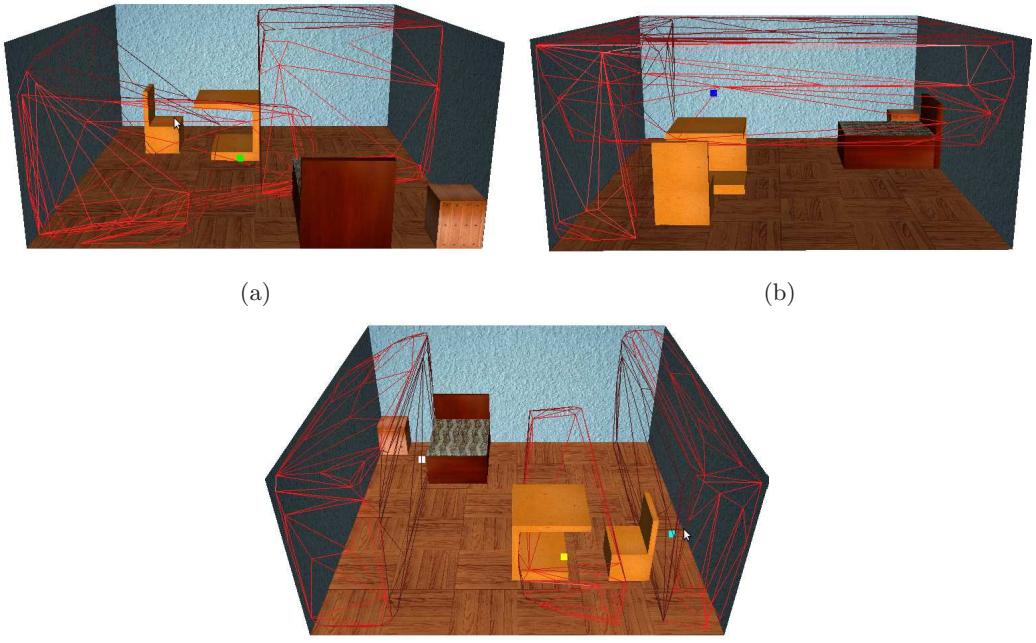
(b) Acercamiento de la trayectoria

(c) Posición final

Figura 8.14: Menor costo a los grados de traslación. Región convexa en el cuarto

8.4. Búsqueda con sensor omnidireccional

En esta sección presentamos resultados arrojados por el algoritmo de expansión de frente de onda que calcula los caminos para visitar las regiones convexas, ver capítulo 5. El algoritmo en su forma original calcula los caminos optimizando los 7 g.d.l. que posee el robot, y nuestra propuesta es optimizar sólo un grupo reducido de g.d.l. y el resto definirlos aleatoriamente como se explicó en la sección 6.3. Por sus dimensiones y relativa sencillez, el ambiente del dormitorio, (figura 8.2), es conveniente para ejecutar el algoritmo, optimizando todos los g.d.l. y/o un subgrupo de ellos. La figura 8.15 muestra un subconjunto de regiones convexas construidas en el espacio libre del ambiente 2.



(a) (b)

(c)

Figura 8.15: Regiones convexas en el ambiente 2, dormitorio

A manera de recordatorio, el algoritmo reporta que ha alcanzado una región convexa en particular cuando el efector final está dentro de la región. Evaluamos nuestra heuristica de la función de utilidad para establecer un orden de visita de las regiones. Como expusimos en la sección 6.1, cuando contamos con un sensor omnidireccional basta encontrar una configuración que coloque al efector final del robot, donde radica el sensor, en el interior de una región para sensar por completo la región convexa. De esta manera, podemos evaluar el valor esperado del tiempo para encontrar el objeto suponiendo un sensor omnidireccional.

8.4.1. Optimización con 2 grados de libertad

Primero, realizamos el análisis del algoritmo cuando optimizamos sólo 2 grados de libertad y los restantes los definimos aleatoriamente. Después analizaremos como se comporta el algoritmo optimizando los 7 g.d.l. de todo el sistema. Entonces, los dos grados de libertad que optimizamos son los de traslación en el plano (x, y) y tomamos los g.d.l. de rotación de la base y de las articulaciones del brazo aleatoriamente.

Presentamos un análisis por región de los tiempos de ejecución del algoritmo para 10 corridas realizadas. Para cada corrida se registro 1) el número de regiones, que competían para ser elegidas por la función de utilidad como siguiente región a visitar, 2) la distancia máxima recorrida desde la región actual hasta la región más alejada de las restantes y 3) el tiempo de ejecución del proceso.

La figura 8.16(a) presenta un histograma de los tiempos de ejecución para cada región. El tiempo promedio de ejecución fue de 3.03 segundos, para las 115 regiones obtenidas en las 10 corridas, con una desviación estándar de 1.56 segundos.

La figura 8.17(a) muestra un histograma de la distancia máxima entre la región actual y las regiones que restan por visitar. La distancia máxima promedio fue de 110.30 unidades con una desviación estándar de 28.46 unidades.

La figura 8.18(a) muestra la relación entre la distancia máxima y el tiempo de ejecución. Se observa una relación lineal con una correlación entre dichas variables de 0.58. El análisis respalda el comportamiento esperado del algoritmo, que a mayor distancia recorrida para visitar las regiones restantes resulta en un mayor tiempo de ejecución del proceso.

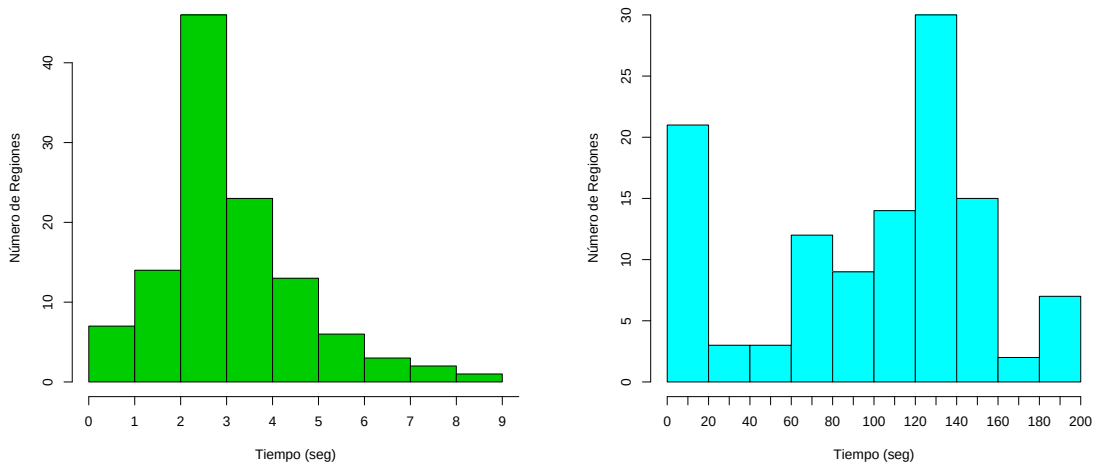
8.4.2. Optimización con 7 grados de libertad

Ahora, presentamos el mismo análsis de la sección anterior pero aplicado a los resultados de 10 corridas realizadas optimizando los 7 g.d.l. del robot. La figura 8.16(b) presenta un histograma de los tiempos de ejecución para cada región. El tiempo promedio de ejecución fue de 97.45 segundos, para las 116 regiones obtenidas en las 10 corridas, con una desviación estándar de 55.49 segundos.

La figura 8.17(b) muestra un histograma de la distancia máxima entre la región actual y las regiones que restan por visitar. La distancia máxima promedio fue de 71.39 unidades con una desviación estándar de 15.83 unidades.

La figura 8.18(b) muestra la relación entre la distancia máxima y el tiempo de ejecución.

En la imagen se observa que existe una relación lineal, según el análisis estadístico encontramos una correlación entre dichas variables de 0.53. Al igual que el análisis anterior, esté análisis respalda el comportamiento esperado del algoritmo, que la distancia recorrida para visitar las regiones restantes es proporcional al tiempo de ejecución del proceso.



(a) Optimización de 2 g.d.l.

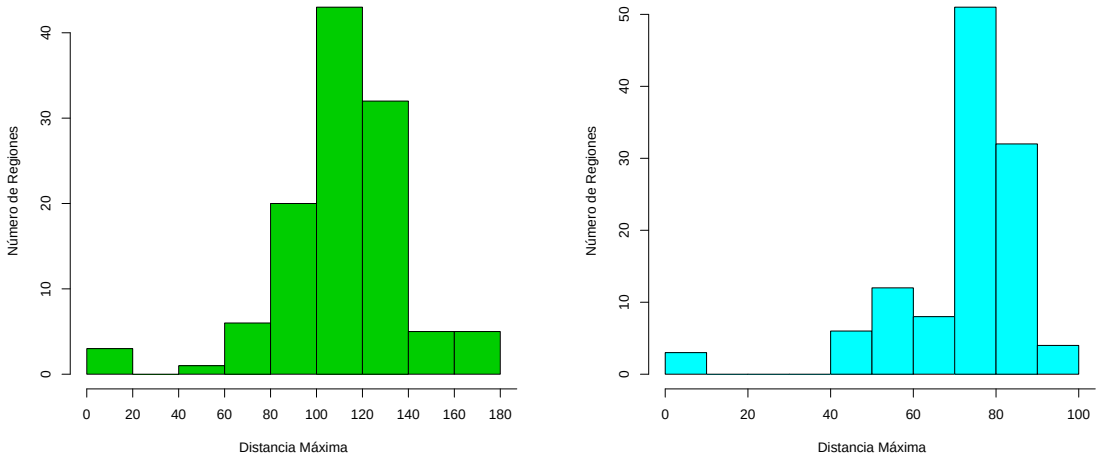
(b) Optimización de 7 g.d.l.

Figura 8.16: Tiempos de ejecución por región en el ambiente 2, dormitorio

8.4.3. Comparación

Las dos subsecciones anteriores muestran la diferencia en la optimización de 2 g.d.l. y 7 g.d.l. del robot. Es evidente la diferencia que existe entre los tiempos promedios estimados optimizando 2 g.d.l. y 7 g.d.l., 3.03 y 97.45 segundos, respectivamente. La diferencia en los tiempos promedio de ejecución por región es de al menos 90 segundos menor optimizando 2 g.d.l. que optimizando 7 g.d.l. Lo anterior nos indica que, para el ambiente del dormitorio con 11 regiones convexas, la diferencia en los tiempos de ejecución optimizando 2 g.d.l. y 7 g.d.l. sería de al menos 990 segundos, equivalente a 16.5 minutos más de tiempo de ejecución optimizando los 7 g.d.l. del robot.

Ahora comparemos las distancias máximas, los resultados obtenidos en esta sección denotan una diferencia en la distancia máxima recorrida desde una región dada a la siguiente región a visitar (incluir) en el camino global. Se observa una diferencia en la distancia máxima promedio



(a) Optimización de 2 g.d.l.

(b) Optimización de 7 g.d.l.

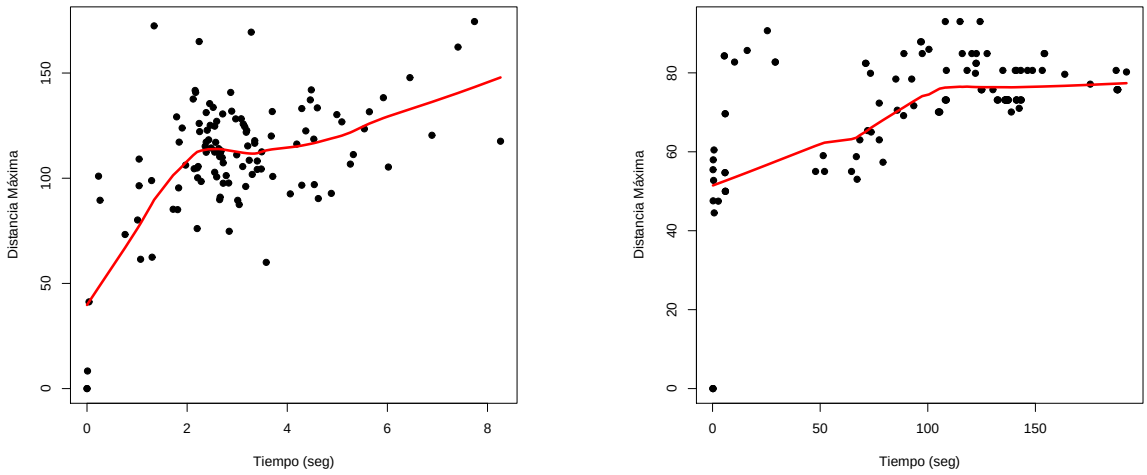
Figura 8.17: Distancia máxima entre región actual y regiones restantes

optimizando 2 g.d.l. y 7 g.d.l. de 110.30 y 71.39 unidades, respectivamente. Por lo tanto, la diferencia en las distancias máximas promedio es de al menos 40 unidades menor optimizando 7 g.d.l. que optimizando 2 g.d.l. del robot.

Este resultado impacta directamente en el valor esperado del tiempo, E[T], ya que una mayor distancia recorrida para cada región convexa incrementa el tiempo de búsqueda del objeto. No es posible estimar la diferencia que existe en el E[T] optimizando 2 g.d.l. y 7 g.d.l. debido a que el cálculo del valor esperado del tiempo involucra la distancia recorrida para alcanzar cada región y el volumen de cada región, siendo variable en cada simulación del ambiente.

No obstante lo anterior, se registraron los valores esperados del tiempo de las 10 simulaciones realizadas optimizando 2 g.d.l. y 7 g.d.l. para proporcionar una estimación del desempeño del algoritmo. La figura 8.19(a) presenta un histograma del valor esperado del tiempo optimizando 2 g.d.l. (barras verdes) y 7 g.d.l. (barras cian). Se puede observar que el E[T] de las simulaciones optimizando 2 g.d.l. es mayor y posee más variabilidad. La figura 8.19(b) muestra un diagrama de caja y brazo de los E[T] de cada grupo de simulaciones. El promedio del valor esperado del tiempo de las 10 simulaciones fue de 36.78 y 19.17 optimizando 2 g.d.l. y 7 g.d.l., respectivamente.

Por último, las imágenes 8.20 y 8.21 muestran un ejemplo de los caminos globales para



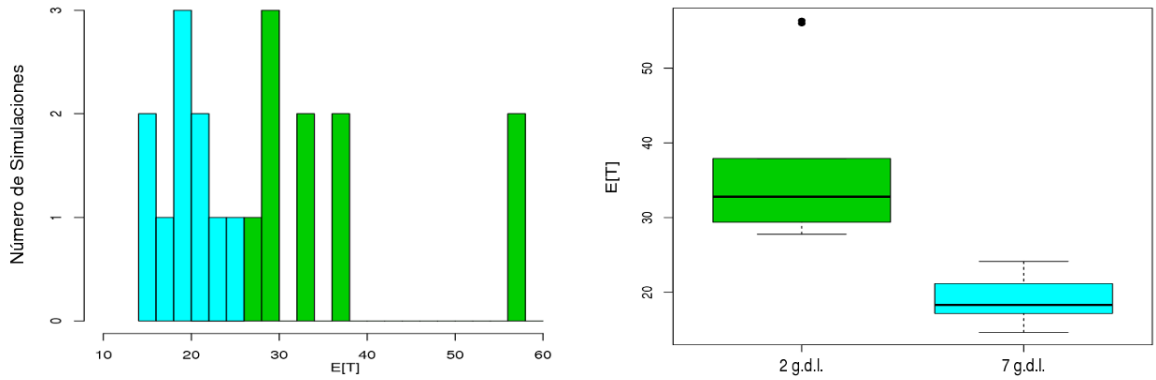
(a) Optimización de 2 g.d.l.

(b) Optimización de 7 g.d.l.

Figura 8.18: Distancia máxima vs. Tiempo de Ejecución

alcanzar el conjunto de regiones convexas, opmizando 2 g.d.l. y 7 g.dl. respectivamente. La figura 8.20(a) muestra el camino global por la parte superior para apreciar el movimiento del brazo y la figura 8.20(b) muestra el camino por la parte inferior para visualizar el desplazamiento de la base, cuando optimizamos 2 g.d.l. La figura 8.21(a) y 8.21(b) muestran la vista superior e inferior del camino, respectivamente, cuando optimizamos 7 g.d.l. En las imágenes 8.20(a) y 8.21(a) podemos notar diferencias en el movimiento del brazo debido a la optimización; en la figura 8.20(a) el brazo tiene mayor movimiento, ya que elegimos aleatoriamente los grados de rotación de las uniones. En las figuras 8.20(b) y 8.21(b) se muestra como la base se desplaza en líneas rectas usando ángulos rectos para girar en el plano (x, y), este desplazamiento es resultado del algoritmo de expansión de frente de onda que hace una optimización en distancia Manhatan en los grados de libertad de traslación.

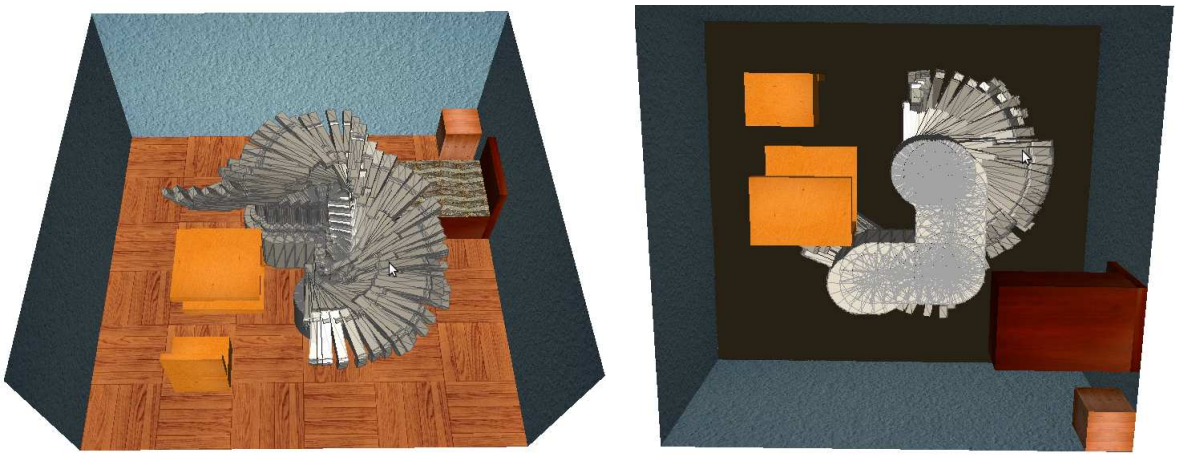
Debido al tiempo de ejecución empleado para generar los caminos para visitar las regiones convexas optimizando los 7 g.d.l., sólo presentamos un ejemplo más. Utilizamos el ambiente tipo oficina (ambiente 3) para correr ambos algoritmos, optimizando 2 y 7 g.d.l. En la figura 8.7 tenemos un ejemplo de las regiones convexas creadas en este ambiente. La tabla 8.1 muestra los datos obtenidos para cada caso de optimización y en la figura 8.22 podemos ver los correspondientes caminos globales para alcanzar todas las regiones. Como puede apreciarse, el tiempo de ejecución para optimizar los 7 g.d.l. es mucho mayor que el tiempo para optimizar sólo 2 g.d.l. Mientras el valor esperado del tiempo es ligeramente menor optimizando los 7 g.d.l.



(a) (b)

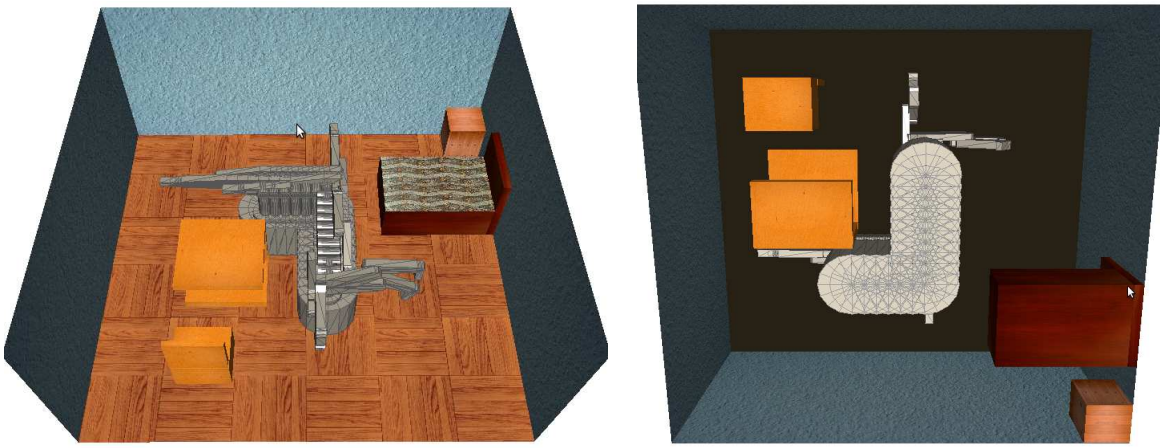
Figura 8.19: E[T] optimizando 2 g.d.l.(verde) y 7 g.d.l. (cyan) del robot

Esta información obtenida sobre el ambiente tipo oficina evidencia el cuello de botella de nuestra estrategia de búsquedad, el tiempo de ejecución. Y deja ver la poca ventaja obtenida respecto al valor esperado del tiempo al optimizar todos los g.d.l. Por lo tanto, en la subsecuente generación de planes de búsqueda utilizamos el algoritmo que óptimiza sólo los grados de traslación del robot.



(a) Vista superior (b) Vista inferior

Figura 8.20: Camino global optimizando 2 g.d.l.



(a) Vista superior (b) Vista inferior

Figura 8.21: Camino global optimizando 7 g.d.l.

	R. convexas	Tiempo de ejecución	Valor esperado del tiempo
2 g.d.l.	23	11 min. 40 s.	41.28
7 g.d.l	23	10 hrs. 51 min.	36.43

Tabla 8.1: Tiempo de ejecución optimizando 2 y 7 g.d.l. Ambiente 3, oficina

(a) Optimización de 2 g.d.l. (b) Optimización de 7 g.d.l.

Figura 8.22: Camino global. Ambiente 3

8.5. Búsqueda por regiones convexas utilizando un sensor limitado

En la sección anterior, presentamos un análisis utilizando un sensor omnidireccional, con el cual sólo se requiere encontrar una configuración de sensado que coloque el efector final del robot dentro de una región convexa para cubrirla completamente. En esta sección presentamos los resultados considerando un sensor limitado tanto en campo como en rango de alcance máximo y mínimo. Con un sensor limitado es necesario un conjunto de configuraciones de sensado, seguramente de más de un elemento, para cubrir completamente una región convexa.

Entonces, una vez que hemos definido el orden de visita de las regiones convexas, el robot va ha sensar cada región, una tras otra según el orden establecido. En cada región convexa el robot manipulador móvil, equipado con el sensor limitado, se moverá entre las configuraciones de sensado y cubrirá con el frustrum del sensor el espacio libre del ambiente perteneciente a la región actual y el espacio que alcance de otras regiones aún no sensadas.

A continuación mostramos los resultados de nuestras simulaciones. Utilizamos el ambiente 3, tipo oficina de la figura 8.3. Para este ejemplo, primero, se crearon 19 regiones convexas, en un tiempo de 3 minutos y 32 segundos. En seguida, se obtuvo el orden de visita en 8 minutos y 33 segundos, siendo el orden el siguiente:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C_6	C_{20}	C_3	C_{18}	C_7	C_{22}	C_9	C_{13}	C_{11}	C_1	C_{12}	C_{15}	C_8	C_{16}	C_{21}	C_2	C_4	C_{19}	C_5

Después, con el algoritmo 6.1 se generaron 827 configuraciones aleatorias para cubrir el espacio libre de trabajo. Entonces, se aplicó el algoritmo 6.2 para encontrar un conjunto reducido de configuraciones necesario para sensar el espacio libre del ambiente, el número conseguido de configuraciones fue de 39, el cual es una reducción significativa de la cantidad de configuraciones de sensado iniciales. Por último, se genero un orden de visita para éste conjunto de configuraciones de sensado y los caminos para ir de una a otra configuración. El tiempo para generar las configuraciones de sensado aleatorias, encontrar el conjunto reducido, calcular el orden y los caminos para moverse entre ellas fue de 64.57 segundos. Entonces el tiempo de ejecución para generar el plan completo fue de 13 minutos y 11 segundos. El valor esperado del tiempo para encontrar el objeto con un sensor limitado siguiendo este plan es de 515.837 unidades.

Ahora, veamos algunas imágenes del proceso de sensado. En la figura 8.7 mostramos un

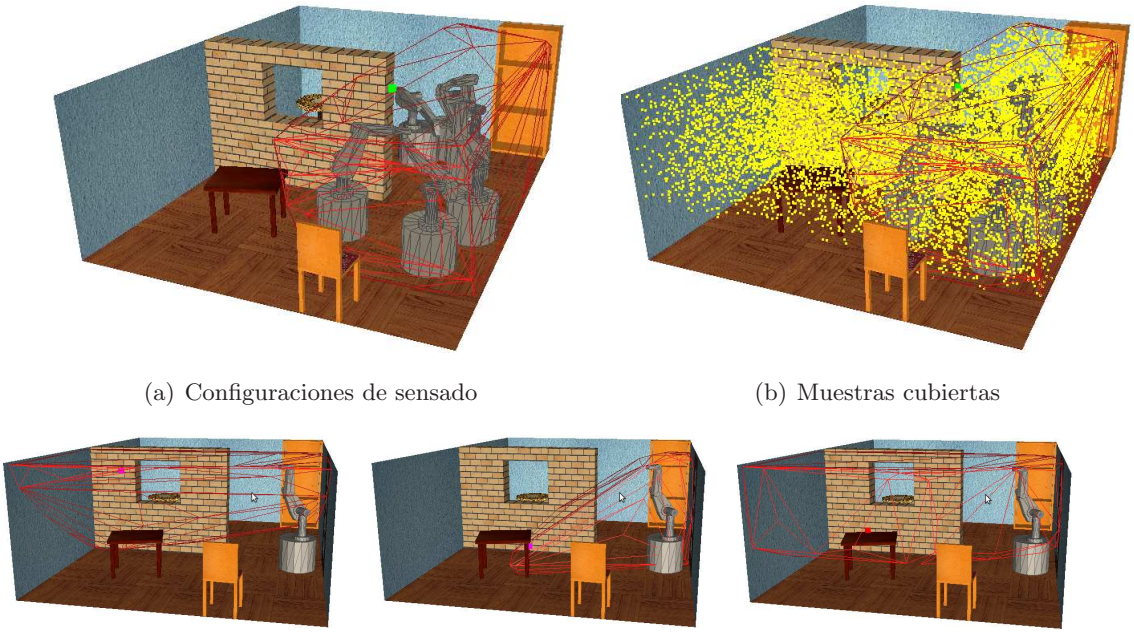
subconjunto de las regiones convexas creadas para este ambiente. De las regiones convexas construidas tomamos la región C_6 para seguir el proceso de sensado. La región C_6 se muestra en malla color rojo en la figura 8.23(a). También, en la figura 8.23(a) se puede ver el conjunto de configuraciones de sensado que cubren la región. En la figura 8.23(b) tenemos las muestras que son cubiertas por los frustums del sensor dentro de la región y también vemos las muestras que pertenecen a otras regiones, éstas muestras son sensadas y logicamente removidas por estar en regiones donde radica el sensor. Las figuras 8.23(c), 8.23(d) y 8.23(e) muestran un ejemplo de una configuración de sensado asociada a la región C_6, que simultáneamente coloca el efector final del robot dentro de las regiones C_3, C_{18} y C_{20} que aún no son sensadas. Con esta simulación podemos verificar que el proceso de sensado efectivamente respeta la definición de región de visibilidad de la sección 6.1.

En la figura 8.24 vemos otro ejemplo de sensado, en este caso se cubre la región del entrepaño superior del librero, la configuración de sensado en la imagen 8.24(a) cubre las muestras en el entrepaño y también las que se sitúan al lado del librero pertenecientes a otra región diferente. En la figura 8.24(b) vemos las regiones convexas donde se encuentra el efector final para producir este resultado de sensado.

La figura 8.25 muestra como el robot sensa el espacio debajo de la silla y la mesa, así mismo, podemos ver las configuraciones de sensado necesarias para cubrir las regiones en este espacio y el camino para ir de la mesa a la silla. Como se puede ver la estrategia de búsqueda será eficaz siempre que las habilidades del robot y geometría de los objetos lo permitan.

Utilizamos también el ambiente 6, figura 8.6, para ejecutar la estrategia de búsqueda. En la figura 8.26(a) podemos ver dos regiones consecutivas en el orden de visita, C_1 al fondo y C_2 al frente. También, la figura 8.26(a) muestra el camino para moverse de la región C_1 a la región C_2. La figura 8.26(b) visualiza las configuraciones de sensado necesarias para cubrir la región C_2. En la figura 8.26(c) se tienen dos configuraciones de sensado pertenecientes a la región C_2 y el camino para ir de una configuración a otra, también podemos observar que el camino entre ellas es lineal en el espacio de configuraciones de 7 dimensiones. Finalmente, la figura 8.26(d) muestra el camino global para que el robot visite todas las regiones convexas, visto de otro modo, el camino que hace posible que el sensor cubra todo el espacio libre del ambiente para buscar el objeto. El camino global es obtenido optimizando sólo los grados de libertad que definen la posición (x,y) de la base del robot, como se explicó en capítulos anteriores.

Para este ambiente se construyeron 47 regiones convexas y se necesitaron en total 512 configuraciones de sensado para cubrirlo completamente. El tiempo de ejecución de nuestro software para calcular el plan de búsqueda con el sensor limitado fue de 1 hora y 50 minutos



(a) Configuraciones de sensado (b) Muestras cubiertas

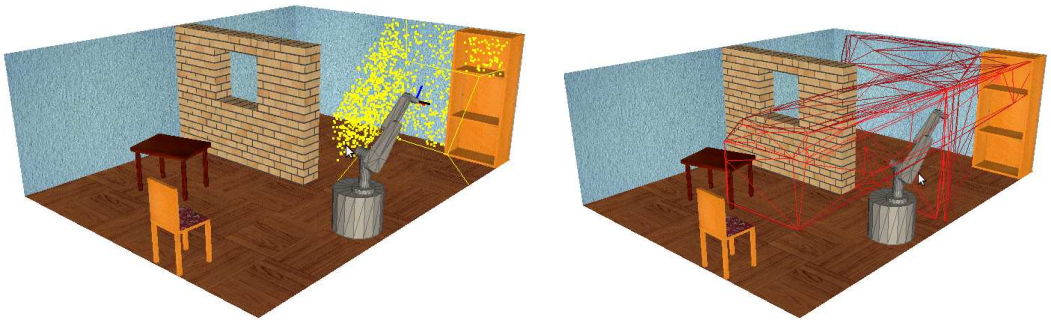
(c) (d) (e)

Figura 8.23: Sensado al interior de una región convexa

aproximadamente y el valor esperado de tiempo del plan fue de 100.44 unidades. Estos tiempos de computo dependen significativamente de la discretización de los grados de libertad del robot.

Por último ejecutamos el algoritmo sobre el ambiente 2, tipo dormitorio. La tabla 8.2 resume los resultados obtenidos por la estrategia de búsqueda optimizando 7 y 2 g.d.l. La tabla esta dividida por las principales bloques de la estrategia, como son: la construcción de regiones convexas, el orden de visita de las regiones y el sensado de las regiones. Se realizarón 30 corridas para obtener los datos. La información obtenidad refuerza el análisis que hemos presentado.

Antes de pasar a la última sección de resultados, mostramos en la tabla 8.3, los tiempos de ejecución de los caminos para visitar las configuraciones de sensado. En este caso especifico, la estrategia de búsqueda aprovecha la técnica de retraso en la examinación de colisiones (del inglés; lazy colision cheking) que pueden ocurrir en el camino propuesto, es decir, se pospone la examinación de colisiones hasta que se planea todo el camino. Sin la técnica, se checa que no haya colisión, entre el robot y los obstáculos y también entre las mismas partes del robot, mientras se planea el camino. Entonces en la tabla, se reportan los tiempos de ejecución, con y sin la técnica, de los caminos para visitar las configuraciones de sensado sobre el ambiente 3, tipo oficina. Se realizaron 10 corridas del algoritmo para obtener los datos, se puede ver en la



(a) (b)

Figura 8.24: Ejemplo del proceso de sensado del ambiente

información obtenida que utilizar la técnica reduce el tiempo de ejecución de la estrategia de busqueda como se reporto en [67].

Hay que resaltar que en nuestra metodología, el subproceso que consume mayor tiempo de ejecución para generar el plan global de búsqueda del objeto, es la visita de las regiones convexas, particularmente, la generación de los caminos globales para visitar las regiones, como se puede ver en la tabla 8.2. En consecuencia, nosotros optamos por la metodología que optimiza sólo 2 g.d.l. y que utiliza la técnica de retraso en la examinación de colisiones. Si bien esta elección afecta al valor esperado del tiempo en encontrar el objeto, incrementándolo, también tiene el beneficio de generar el plan de búsqueda en un tiempo menor.

Promedio

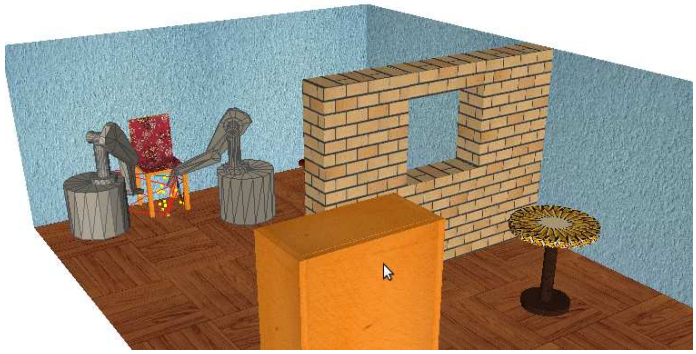
	Construcción de regiones convexas				Visita de regiones			Sensado de regiones (Sensor limitado)					Plan Global
Experimento	M. Libres	M. en R.C.	Tiempo (s)	Núm. R.C.	Distancia	V.E.T.	Tiempo (s)	Distancia	V.E.T.	Tiempo (s)	C.S. Aleatorias	C.S.	Tiempo total (s)
2 gdl	17490.43	17216.17	45.71	12.70	393.82	35.86	76.12	1995.91	462.15	13.77	223.53	31.70	135.60
7 gdl	17500.83	17245.37	44.56	12.87	228.91	18.19	1297.34	1726.98	460.88	11.56	228.33	33.40	1353.46

Desviación estándar

	Construcción de regiones convexas				Visita de regiones			Sensado de regiones (Sensor limitado)					Plan Global
Experimento	M. Libres	M. en R.C.	Tiempo (s)	Núm. R.C.	Distancia	V.E.T.	Tiempo (s)	Distancia	V.E.T.	Tiempo (s)	C.S. Aleatorias	C.S.	Tiempo total (s)
2 gdl	21.52	171.08	10.82	2.76	81.02	7.74	29.43	331.27	91.06	6.10	54.45	3.29	46.35
7 gdl	17.62	139.99	8.03	2.18	42.32	3.68	488.22	275.06	81.07	3.75	51.67	3.64	499.99

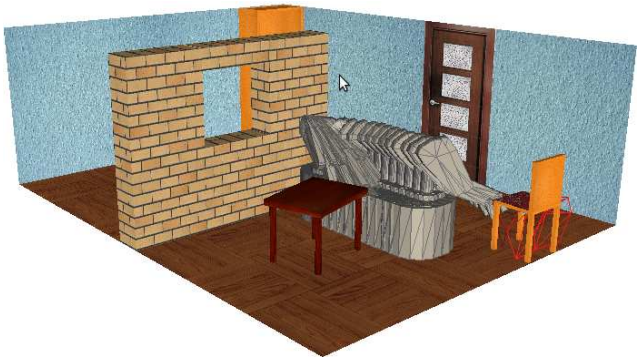
M. = Muestras *R.C. = Regiones convexas* *V.E.T. = Valor esperado del tiempo* *C.S. = Configuraciones de sensado*

Tabla 8.2: Resultados para 30 corridas en el ambiente 2, dormitorio



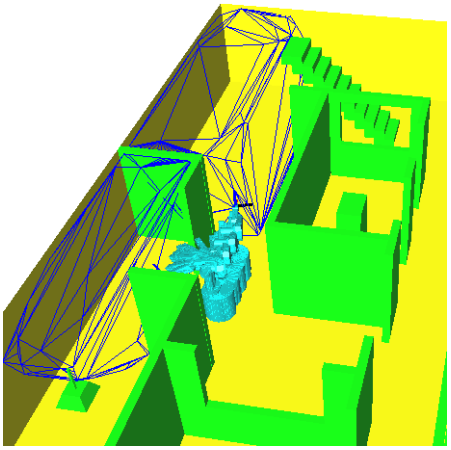
(a)

(b)

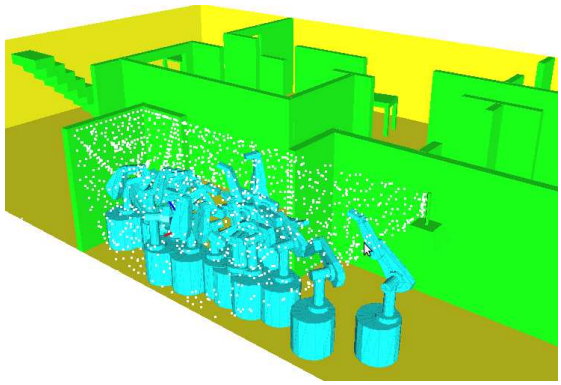


(c)

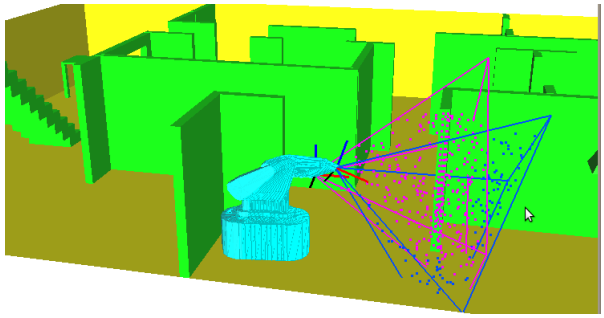
Figura 8.25: Sensado de regiones bajo la mesa y la silla



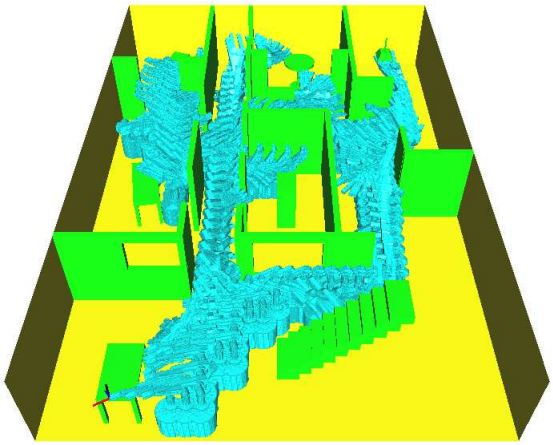
(a) Regiones C_1 y C_2



(b) Configuraciones de sensado en C_2



(c) Camino lineal entre configuraciones



(d) Camino global de búsqueda

Figura 8.26: Búsqueda de un objeto con un sensor limitado

Ambiente 3, oficina				
			Tiempo de ejecución (s)	
	No. de regiones	No. de configuraciones de sensado	Caminos con TREC	Caminos sin TREC
	20	44	5.97	9.30
	21	63	17.46	31.90
	21	46	7.37	18.10
	22	51	7.06	11.28
	23	61	13.19	36.15
	22	49	10.05	22.48
	25	49	7.30	11.89
	22	57	8.17	20.55
	21	48	7.66	17.52
	23	51	6.17	18.34
Promedio	22	51.9	9.04	19.75
Desv. est.	1.41	6.35	3.64	8.67

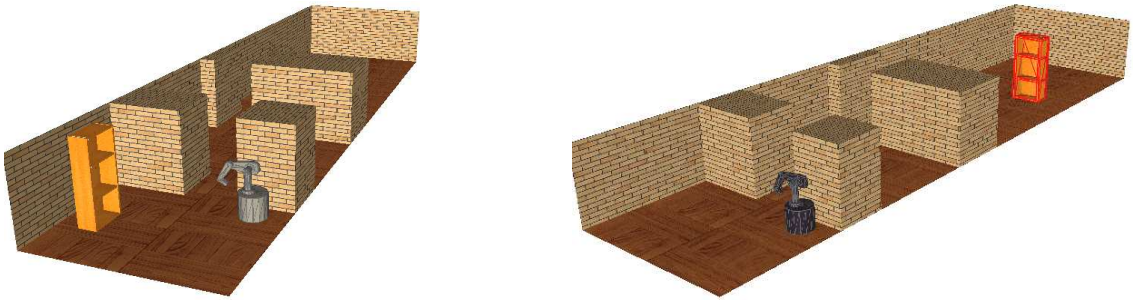
Tabla 8.3: Tiempos de ejecución con la técnica de retraso en la examinación de colisiones, (TREC)

8.6. Reparación de planes globales de búsqueda

Por último, presentamos los resultados en simulación para la estrategia propuesta para la reparación de un plan global de búsqueda, cuando se presenta una modificación local en el ambiente. Utilizamos el ambiente 4, que son tres cuartos consecutivos con dos pasillos que los conectan, ver figura 8.4. En el capítulo 7 utilizamos el ambiente 4 para explicar la estrategia para reparar un plan global, en esta sección presentaremos los resultados obtenidos. De acuerdo al plan global, tenemos un conjunto de regiones convexas que dividen el espacio libre del ambiente, así como un orden para visitar a cada región; en el plan también están incluidos los conjuntos de configuraciones de sensado que cubren cada región convexa y así como los ordenes para cada conjunto de configuraciones de sensado.

8.6.1. Ejemplo 1

Para el primer ejemplo, el librero que se muestra en la figura 8.27(a), se moverá del primer cuarto al último cuarto, a la posición que se muestra en la figura 8.27(b).



(a) Posición inicial (b) Posición final

Figura 8.27: Cambio de posición del librero.

Como primer punto, veamos el plan global original, empezando por las regiones convexas y su orden. Se generaron 10 regiones convexas en el espacio libre del ambiente, la figura 8.28 muestra las regiones convexas en malla de color cian. Las imágenes de las regiones se presentan en la misma secuencia con la que están en el orden establecido por nuestro algoritmo. El orden para visitar las regiones es el siguiente:

$$C_{ID7\_1} \rightarrow C_{ID11\_2} \rightarrow C_{ID10\_3} \rightarrow C_{ID2\_4} \rightarrow C_{ID3\_5} \rightarrow$$
$$C_{ID1\_6} \rightarrow C_{ID6\_7} \rightarrow C_{ID9\_8} \rightarrow C_{ID5\_9} \rightarrow C_{ID4\_10}$$

La figura 8.29(a) muestra el camino global para visitar todas las regiones y la figura 8.29(b) muestra un ejemplo del camino para ir de la región C_{ID10_3} a la región C_{ID2_4}. El número total de configuraciones de sensado para cubrir las 10 regiones convexas fue de 49, en la figura 8.30 podemos ver las configuraciones de sensado que cubren la región C_{ID7_1}, la cual es la primera en el orden. El tiempo total para generar el plan original de búsqueda, para cubrir el ambiente completo con el sensor limitado, fue de 2 minutos y 31 segundos y el valor esperado del tiempo en encontrar el objeto con este plan es de 776.3 unidades.

Como segundo punto, mostramos el espacio del ambiente relacionado con el cambio de lugar del librero. La imágen 8.31 muestra las regiones convexas del plan original que se relacionan con el cambio; la figura 8.31(a) muestra las regiones asociadas a la posición original del librero y la figura 8.31(b) muestra las regiones que serán modificadas en la nueva posición. Las regiones en 8.31(a) corresponde al conjunto $\{C_i\}_t = \{C_{ID11\_2},\ C_{ID10\_3},\ C_{ID6\_7},\ C_{ID9\_8},\ C_{ID5\_9},\ C_{ID4\_10}\}$ y la región en 8.31(b) corresponde al conjunto $\{C_i\}_{t+1} = \{C_{ID2\_4}\}$. Así también, las figuras 8.31(c) y 8.31(d) visualizan los conjuntos de muestras asociadas a las regiones afectadas, conjuntos S_t y S_{t+1} respectivamente. En la figura 8.31(e) vemos las muestras que están en colisión con el obstáculo en la posición original del librero, So_t, se consideran estas muestras ya que representan el espacio del ambiente que quedará vacio al mover el librero. A su vez, la imagen 8.31(f) visualiza las muestras que están en el espacio que ocupará el librero en la nueva posición, So_{t+1}, el espacio representado por este conjunto de muestras tiene que ser excluido, sino la nueva cobertura convexa colisionaría con el librero.

A continuación presentamos como se modificó el plan global original. Se creó el conjunto C_Δ que consta de 7 nuevas regiones convexas en el espacio relacionado al conjunto de muestras $S_\Delta = (S_t \cup So_t) \cup (S_{t+1} \backslash So_{t+1})$, ver figura 8.32(a).

$$C_\Delta = \{C_{ID12},\ C_{ID15},\ C_{ID16},\ C_{ID17},\ C_{ID19},\ C_{ID20},\ C_{ID21}\}$$

Las regiones que se mantuvieron sin modificación fueron 3, $\{C_i\}_{t_0} = \{C_{ID7},\ C_{ID3},\ C_{ID1}\}$, y se presentan en la figura 8.32(b). Entonces, la nueva cobertura convexa, $N = C_\Delta \bigcup \{C_i\}_{t_0}$, para el plan reparado consta de 10 regiones convexas. El nuevo orden de visita de las regiones es el siguiente:

$$C_{ID7\_1} \rightarrow C_{ID12\_2} \rightarrow C_{ID17\_3} \rightarrow C_{ID21\_4} \rightarrow C_{ID19\_5} \rightarrow$$
$$C_{ID20\_6} \rightarrow C_{ID16\_7} \rightarrow C_{ID15\_8} \rightarrow C_{ID3\_9} \rightarrow C_{ID1\_10}$$

La región C_{ID7} preserva el primer lugar en el nuevo orden, ya que no se ve afectada por el cambio en el ambiente. El conjunto de regiones $\{C_i\}_t$ se elimina y en el espacio ocupado por

estás regiones se ubica la región C_{ID12}, la cual es la segunda en el nuevo orden, de acuerdo con el algoritmo 7.1, la región C_{ID12} ocupa el orden de la región C_{ID11}, ya que es la primera región del orden original con la que C_{ID12} colisiona. Una vez eliminadas las regiones del plan original C_{ID11} y C_{ID10}, la siguiente región en el orden era C_{ID2}, la cual colisiona con las regiones en el conjunto $C_{\Delta} \setminus \{C_{ID12}\}$, es por ello que las regiones en $C_{\Delta} \setminus \{C_{ID12}\}$ toman los lugares del 3 al 8 en el nuevo orden. Por último, las regiones C_{ID3} y C_{ID1} tienen los lugares 9 y 10 del orden de visita de la nueva cobertura convexa. C_{ID3} y C_{ID1} conservan la secuencia del plan original, es decir, la región C_{ID3} se visita antes que la región C_{ID1}. La figura 8.33 muestra las regiones convexas para el plan reparado, las mallas en color cian corresponden a las nuevas regiones convexas y las mallas en color rojo corresponden a las regiones originales que se mantuvieron sin cambio, las imágenes de las regiones se presentan en orden de visita proporcionado por nuestro algoritmo. La imágen 8.34(a) muestra el camino global del plan reparado.

Se necesitaron 42 configuraciones de sensado para cubrir el conjunto de regiones convexas del plan reparado. Sin embargo, las configuraciones de sensado de la regiónes C_{ID7_1} y C_{ID1_10} se conservaron. El tiempo para reparar el plan fue de 1 minuto y 12 segundos. El valor esperado para encontrar el objeto fue 793.68 unidades. La figura 8.34 muestra el camino global para visitar las regiones y un ejemplo de las configuraciones de sensado para una región del plan reparado de búsqueda; la figura 8.34(b) muestra el camino para ir de la región C_{ID12_2} a la región C_{ID17_3}, 8.34(c) muestra las configuraciones de sensado para cubrir la región C_{ID17_3} y por último la figura 8.34(c) muestra el camino local para visitar las configuraciones de sensado para esa región.

8.6.2. Ejemplo 2

Para este ejemplo, movimos el librero en el mismo cuarto donde se ubicaba originalmente, ver figura 8.35. El plan original consta de 11 regiones coonvexas. La figura 8.37 muestra las regiones convexas en el orden del plan original, así como los caminos para alcanzarlas. El orden es el siguiente:

$$C_{ID10\_1} \rightarrow C_{ID8\_2} \rightarrow C_{ID7\_3} \rightarrow C_{ID5\_4} \rightarrow C_{ID1\_5} \rightarrow$$
$$C_{ID2\_6} \rightarrow C_{ID3\_7} \rightarrow C_{ID6\_8} \rightarrow C_{ID9\_9} \rightarrow C_{ID10\_10} \rightarrow C_{ID4\_11}$$

Se necesitaron 44 configuraciones de sensado para cubrir el ambiente con el sensor limitado. El tiempo total de ejecución del plan original fue de 1 minuto y 58 segundos. El valor esperado del tiempo para encontrar el objeto fue de 796 unidades.

El plan original se modificó de la siguiente manera:

- Regiones convexas afectadas por el cambio

$$\{C_i\}_t \bigcup \{C_i\}_{t+1} = \{C_{ID10\_1},\ C_{ID8\_2},\ C_{ID7\_3},\ C_{ID5\_4},\ C_{ID6\_8},\ C_{ID9\_9},\ C_{ID10\_10},\ C_{ID4\_11}\}$$

- Regiones que se mantienen sin modificación

$$\{C_i\}_{t_0} = \{C_{ID1\_5},\ C_{ID2\_6},\ C_{ID3\_7}\}$$

- Regiones convexas nuevas

$$C_\Delta = \{C_{ID12},\ C_{ID13},\ C_{ID14},\ C_{ID15},\ C_{ID16},\ C_{ID17},\ C_{ID18},\ C_{ID19}\}$$

- Nuevo orden de visita de las regiones convexas en $N = C_\Delta \bigcup \{C_i\}_{t_0}$, ver figura 8.38

$$C_{ID17\_1} \rightarrow C_{ID18\_2} \rightarrow C_{ID16\_3} \rightarrow C_{ID14\_4} \rightarrow C_{ID13\_5} \rightarrow$$
$$C_{ID19\_6} \rightarrow C_{ID15\_7} \rightarrow C_{ID12\_8} \rightarrow C_{ID1\_9} \rightarrow C_{ID2\_10} \rightarrow C_{ID3\_11}$$

Para el plan reparado, se eliminaron 8 regiones del plan original, figura 8.31(a) y se crearon 8 nuevas regiones, en el primer cuarto donde el librero cambió de lugar, y se conservaron 3 regiones del plan original, las regiones creadas en los dos cuartos donde no está el librero y la región que se crea en el pasillo que conecta los cuartos, figuras 8.28(f), 8.28(d) y 8.28(e), respectivamente. La figura 8.36(a) muestra el camino global para el plan original y 8.36(b) muestra el camino global del plan modificado. Se necesitaron 43 configuraciones de sensado para cubrir el ambiente modificado. El tiempo de ejecución para modificar el plan fue de 21.5 segundos. El valor esperado del tiempo para encontrar el objeto ligado al nuevo plan es de 771.20 unidades.

En la reparación del plan global para este ejemplo, básicamente se modificaron los sub-ordenes de visita de las regiones en el primer cuarto, así como las configuraciones de sensado para cubrir éste cuarto. El sub-orden y las configuraciones de sensado de los otros dos cuartos y el pasillo que los conecta se conservaron sin modificación.

El ambiente 4 por sus dimensiones y estructura produce un número pequeño de regiones convexas y además la mayoría de ellas se traslapan unas con otras, como se puede ver en las figuras del ejemplo 1. Entonces, cuando el obstáculo cambia de lugar provoca que una porción considerable del ambiente este involucrada en el cambio. Esto se puede constatar con los ejemplos anteriores al tener más de la mitad de las regiones relacionadas con la modificación

del ambiente. Por esta razón, el orden de visita de las regiones se altera en su mayoría. Sin embargo, también podemos notar de los ejemplos anteriores que la estrategia de reparación realiza, en lo posible, los cambios al plan original de manera local, al preservar la secuencia de visita de las regiones convexas originales que permanecen intactasy mantener las configuraciones de sensado y los caminos entre las configuraciones de las regiones originales.

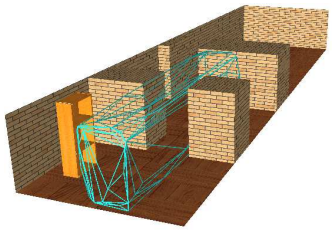
En resumen, en las tablas 8.4 y 8.5 presentamos los datos representativos del plan de búsqueda original y el plan reparado. Aunque en el ejemplo 2 el valor esperado del tiempo mejoro con el nuevo plan, en general nuestros experimentos reportan que el valor esperado del tiempo del plan de búsqueda reparado es ligeramente mayor que el plan original. Mientras el tiempo de ejecución para reparar el plan si se reduce considerablemente, con respecto al tiempo de ejecución del plan original. El número de regiones convexas y configuraciones de sensado permanecen muy similares debido a que es el mismo espacio libre del ambiente que se necesita cubrir en ambos planes.

	R. convexas	Configuraciones	Tiempo de ejecución	V. E. del tiempo
Ejemplo 1	10	49	2 min. 31 s.	776.3
Ejemplo 2	11	44	1 min. 58 s.	796

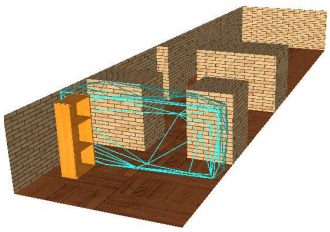
Tabla 8.4: Plan de búsqueda original

	R. convexas	Configuraciones	Tiempo de ejecución	V. E. del tiempo
Ejemplo 1	10	42	1 min. 12 s.	793.68
Ejemplo 2	11	43	21 s.	771.68

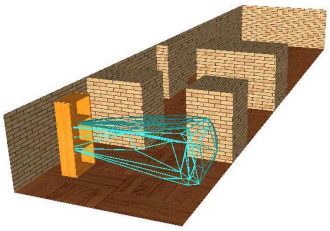
Tabla 8.5: Plan de búsqueda reparado



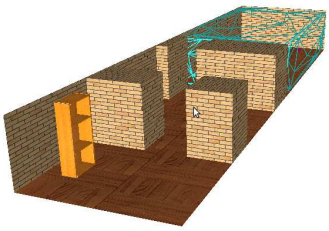
(a) C_{ID7_1}



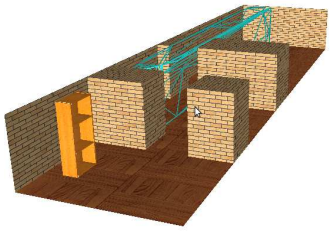
(b) C_{ID11_2}



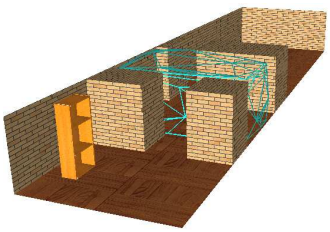
(c) C_{ID10_3}



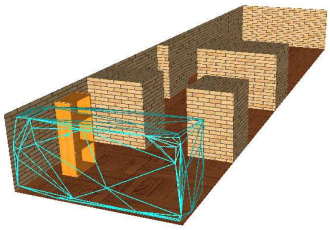
(d) C_{ID2_4}



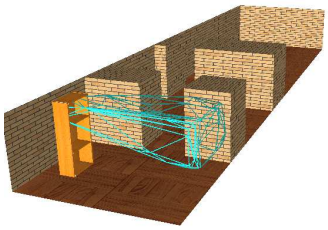
(e) C_{ID3_5}



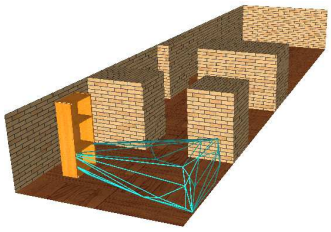
(f) C_{ID1_6}



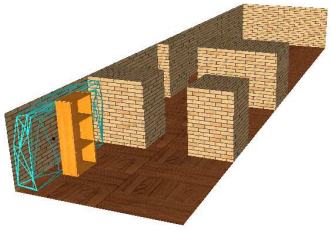
(g) C_{ID6_7}



(h) C_{ID9_8}



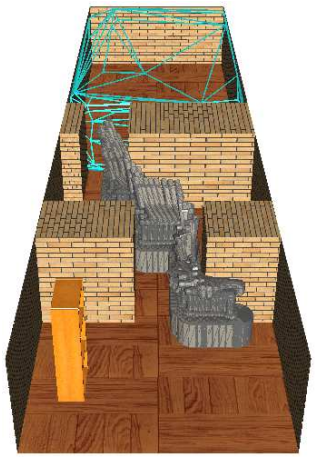
(i) C_{ID5_9}



(j) C_{ID4_10}

Figura 8.28: Regiones Convexas en el ambiente 4

(a) Camino global del plan original



(b) Camino de C_{ID10_3} a C_{ID2_4}

Figura 8.29: Caminos para la búsqueda

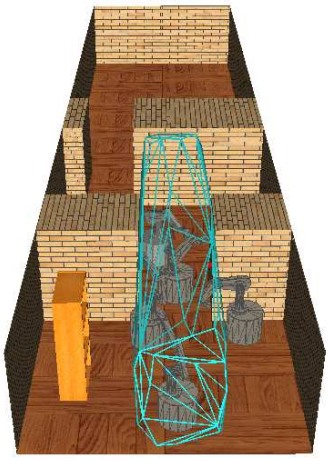
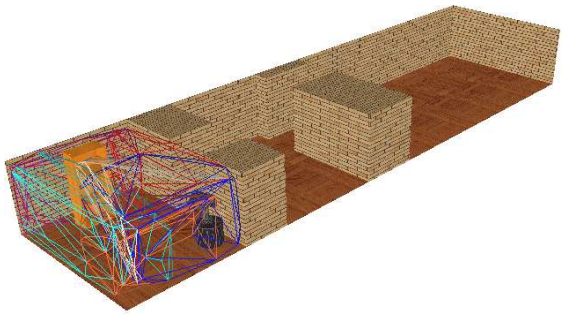
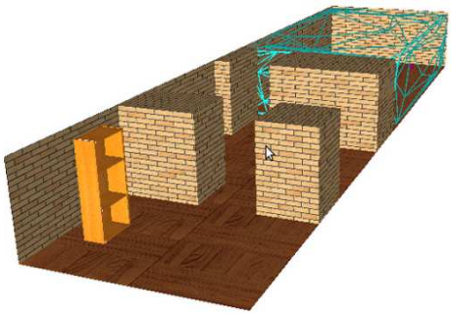


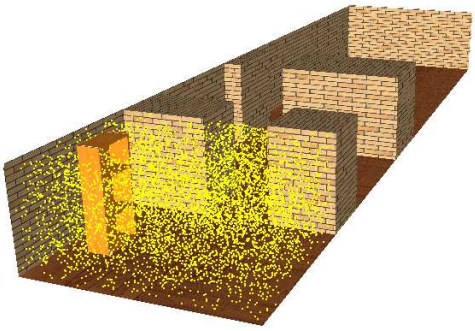
Figura 8.30: Configuraciones de sensado para cubrir la region convexa C_{ID7_1}



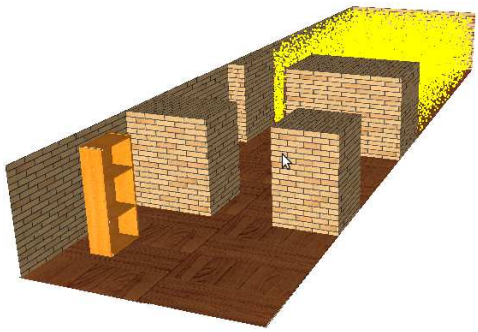
(a) Regiones asociadas a la posición original, $\{C_i\}_t$



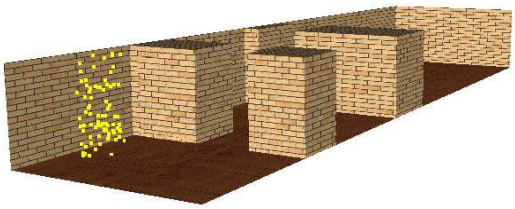
(b) Regiones asociadas a la nueva posición, $\{C_i\}_{t+1}$



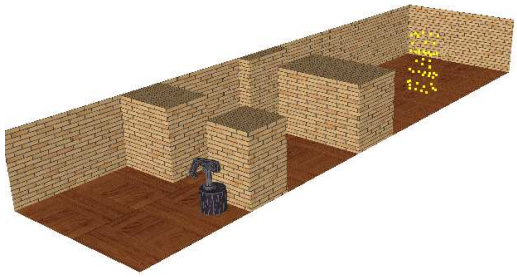
(c) Muestras relacionadas al conjunto $\{C_i\}_t$, S_t



(d) Muestras asociadas al conjunto $\{C_i\}_{t+1}$, S_{t+1}

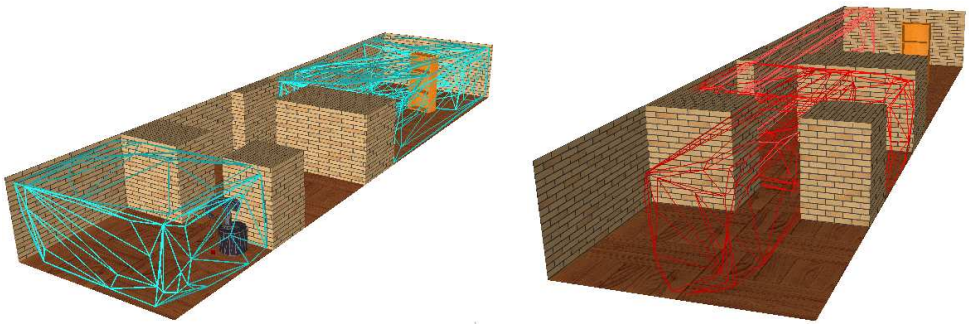


(e) Muestras en colisión en la posición inicial, So_t



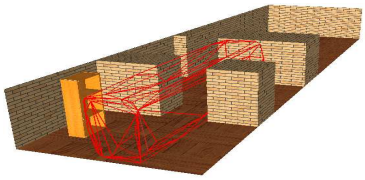
(f) Muestras en colisión en la posición final, So_{t+1}

Figura 8.31: Espacio del ambiente asociado al cambio del obstáculo

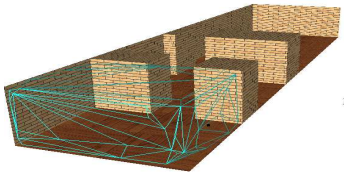


(a) Regiones nuevas, C_Δ (b) Regiones sin modificación, $\{C_i\}_{t_0}$

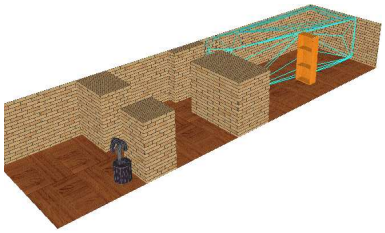
Figura 8.32: Nueva cobertura convexa, N



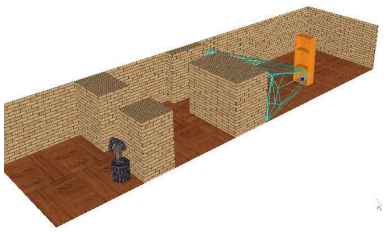
(a) C_{ID7_1}



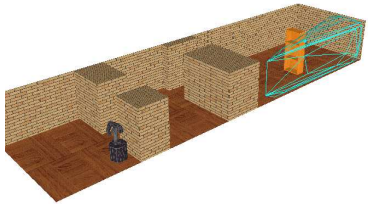
(b) C_{ID12_2}



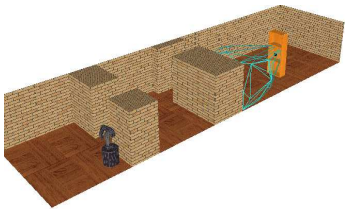
(c) C_{ID17_3}



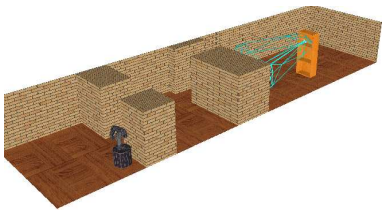
(d) C_{ID21_4}



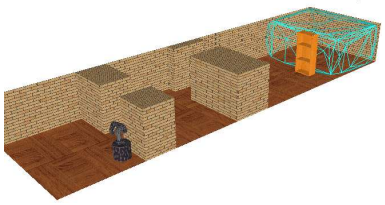
(e) C_{ID19_5}



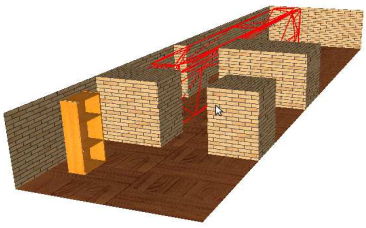
(f) C_{ID20_6}



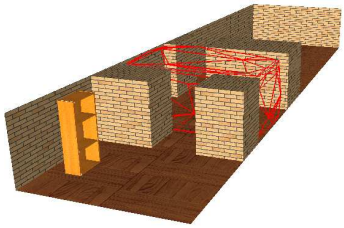
(g) C_{ID16_7}



(h) C_{ID15_8}



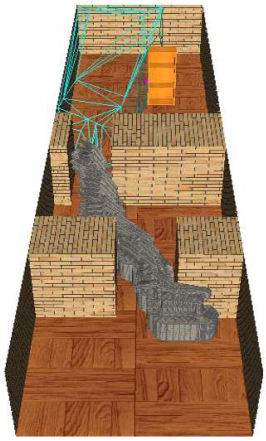
(i) C_{ID3_9}



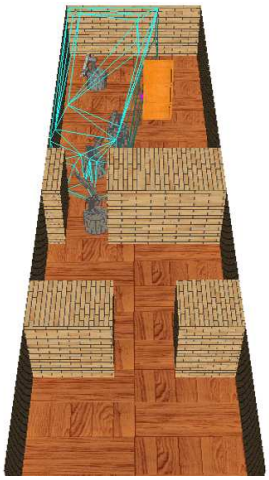
(j) C_{ID1_10}

Figura 8.33: Nueva cobertura convexa y orden de visita de las regiones

(a) Camino global del nuevo plan



(b) Camino entre dos regiones consecutivas en el nuevo orden



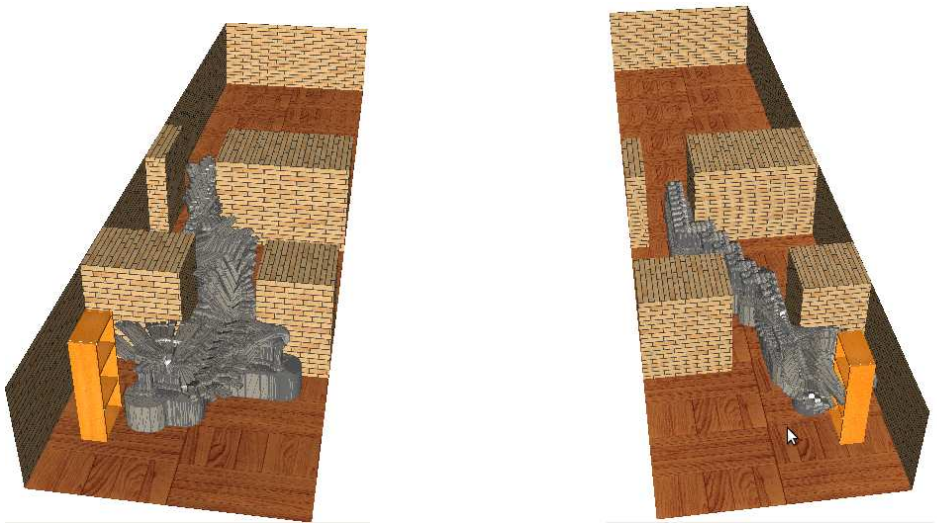
(c) Configuraciones de sensado de la región C_{ID17_3}

(d) Camino para visitar las configuraciones de sensado

Figura 8.34: Nuevo plan para la búsqueda

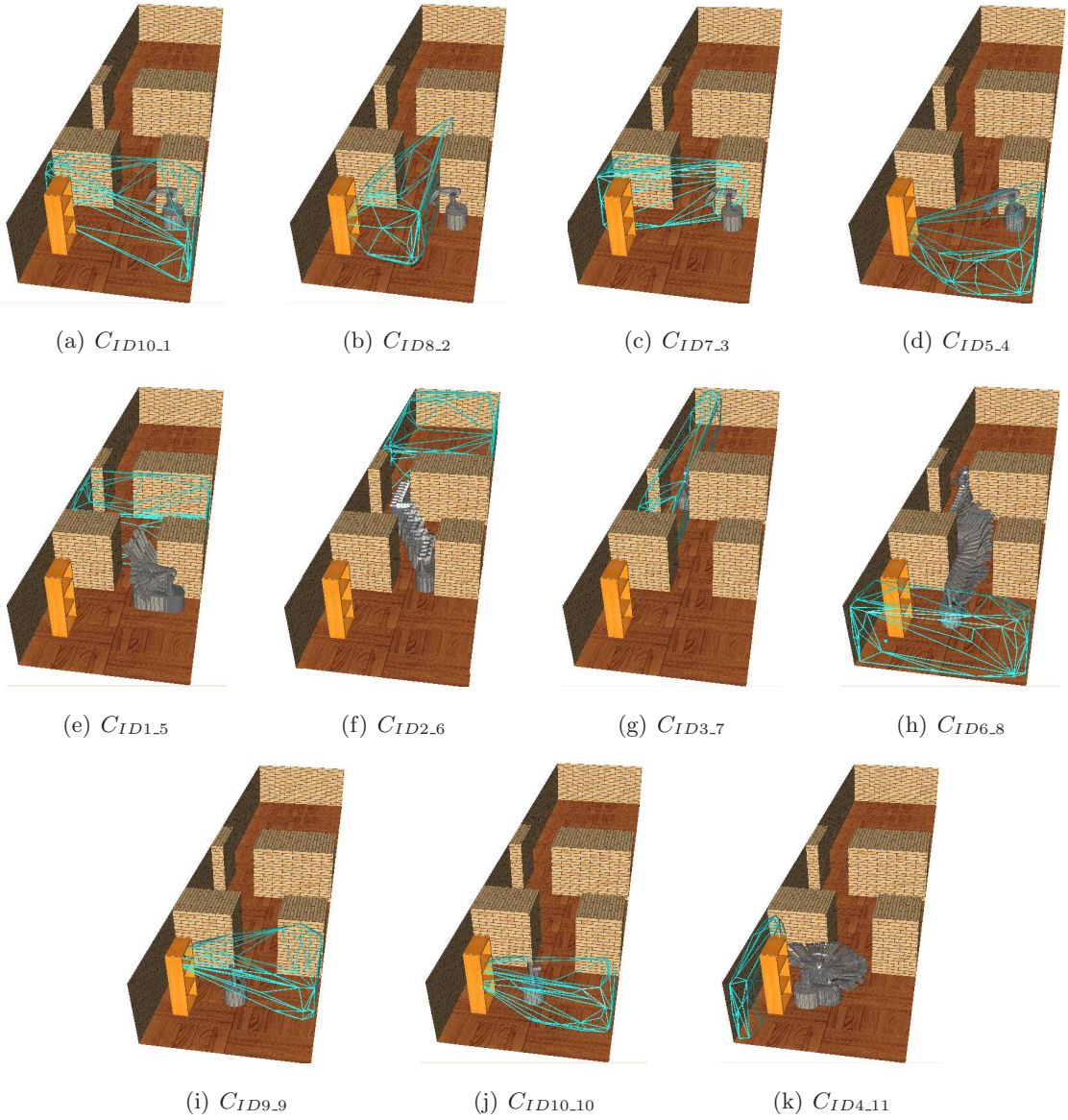
(a) Posición inicial (b) Posición final

Figura 8.35: Cambio de posición del librero. Ejemplo 2



(a) Camino global del plan original (b) Camino global del nuevo plan

Figura 8.36: Caminos Globales

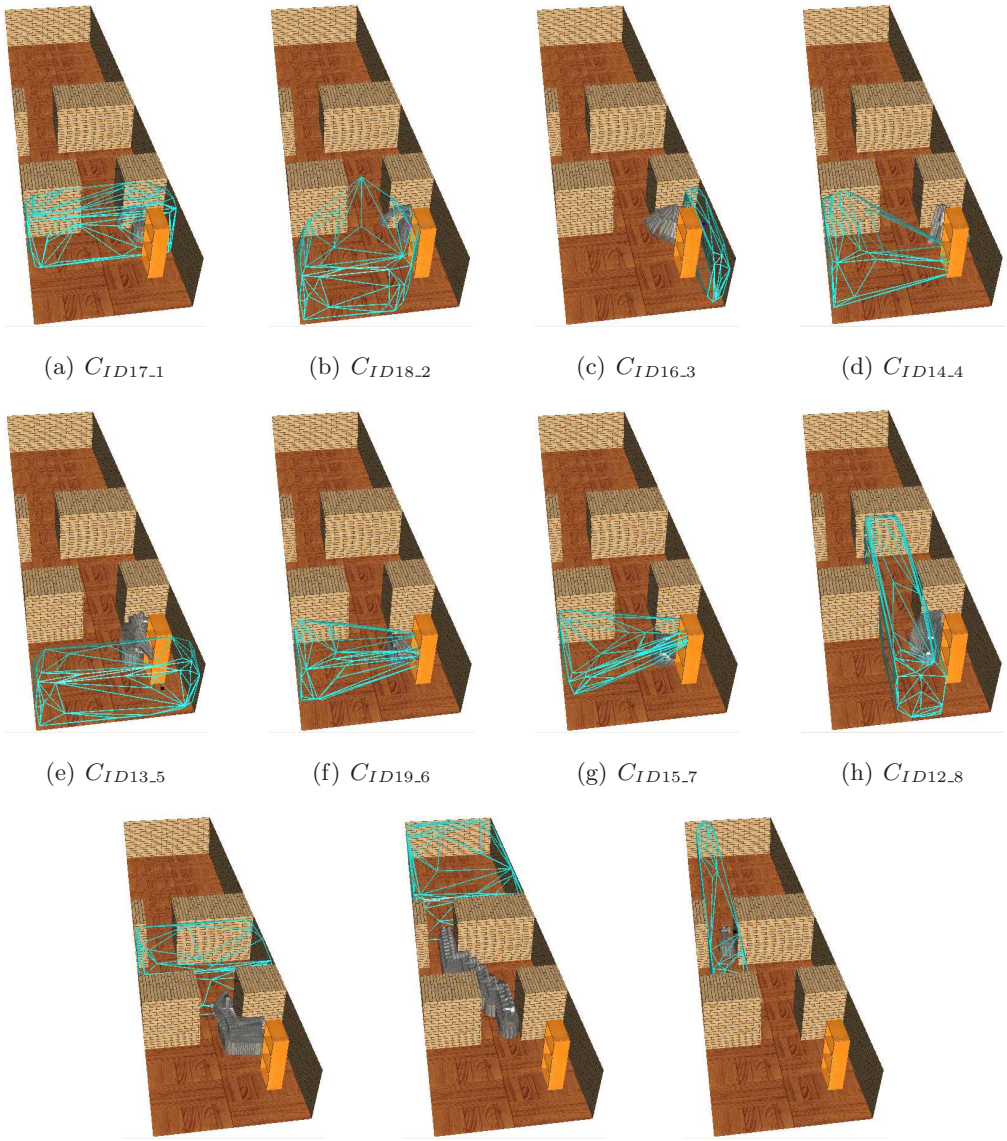


(a) C_{ID10_1} (b) C_{ID8_2} (c) C_{ID7_3} (d) C_{ID5_4}

(e) C_{ID1_5} (f) C_{ID2_6} (g) C_{ID3_7} (h) C_{ID6_8}

(i) C_{ID9_9} (j) C_{ID10_10} (k) C_{ID4_11}

Figura 8.37: Regiones Convexas y caminos globales. Ambiente 4, ejemplo 2



(a) C_{ID17_1} (b) C_{ID18_2} (c) C_{ID16_3} (d) C_{ID14_4}

(e) C_{ID13_5} (f) C_{ID19_6} (g) C_{ID15_7} (h) C_{ID12_8}

(i) C_{ID1_9} (j) C_{ID2_10} (k) C_{ID3_11}

Figura 8.38: Nuevo orden de las regiones convexas y caminos globales. Ambiente 4, ejemplo 2

Capítulo 9

Conclusiones

En el presente trabajo de tesis, nos enfocamos al problema de buscar un objeto con un robot manipulador móvil en un ambiente tridimensional conocido. Debido a la elevada complejidad computacional relacionada a las búsquedas y consultas en ambientes en tres dimensiones, decidimos utilizar un método basado en muestreo aleatorio para determinar la forma y tamaño de las regiones de visibilidad. Estudiamos la dependencia del plan de búsqueda en términos de las capacidades del sensor.

En nuestros experimentos, primero consideramos, un sensor omnidireccional. La descomposición en regiones convexas nos otorga la flexibilidad para determinar donde colocar el sensor y su convexidad nos da garantía de ver la región entera desde cada punto en su interior si usamos un sensor omnidireccional. Hemos presentado un procedimiento para generar trayectorias dependientes de la métrica para medir el movimiento del robot (es decir, trayectorias que mantienen un compromiso entre mover la base y el brazo del robot). Después, extendemos nuestro método al caso de un sensor limitado en campo de vista y rango. Bajo estas limitaciones del sensor observamos que el robot tiene que moverse más y visitar más configuraciones de sensado.

Proponemos la selección de un sub-conjunto de grados de libertad a ser optimizados, lo cual significativamente reduce el tiempo de computo de la generación del plan de búsqueda para encontrar el objeto.

Proponemos un método para reparar planes de búsqueda generados previamente. Mostramos que si el ambiente cambia localmente, también nuestro plan puede modificarse localmente. Para reparar un plan de búsqueda generamos un nuevo subconjunto de configuraciones de sensado y un nuevo orden para visitarlas y consideramos únicamente las regiones relacionadas al cambio

del ambiente 3D. La ventaja principal de reparar el plan localmente en lugar de generar un plan completo nuevamente, es que el tiempo de ejecución necesario para reparar el plan es en general más pequeño que el tiempo requerido para generar un plan completo. Si bien el valor esperado del tiempo se ve afectado, en general, se mantiene cercano al valor del plan original.

Finalmente, en este trabajo, suponemos que la función de distribución de probabilidad que caracteriza la posición del objeto es uniforme (es decir, la probabilidad de encontrar el objeto en una región dada es directamente proporcional al tamaño de la región). Creemos que en general es un buen a priori, dado que para considerar otros tipos de distribuciones de probabiliad, de antemano necesitariamos conocer el tipo de objeto que se busca. Sin embargo en un escenario donde se busca varias veces uno o más objetos, se podría iniciar la búsqueda suponiendo una distribución uniforme y modificar la distribución de acuerdo a los lugares donde el objeto fue encontrado más frecuentemente. Dejamos este problema como trabajo futuro.

Las publicaciones obtenidas en esta tesis hasta el momento son las siguientes [13], [15], [14] y [68].

Apéndice A

Conceptos básicos

En esta sección se presentan algunos conceptos básicos y especificaciones que integran el tema desarrollado.

A.1. Configuración y espacio de configuraciones

En el trabajo desarrollado por Lozano-Pérez [46] se introdujo el concepto de *espacio de configuraciones*, con el cual cualquier robot sin importar su forma geométrica es representado como un punto. Bajo este concepto, el problema de planificación de movimientos se transforma. Ahora el problema es encontrar un camino unidimensional en el espacio de configuraciones libres.

- Configuración. Una configuración de un robot es una especificación de la posición y orientación de todos los puntos del robot relativos a un sistema de coordenadas fijo.
- Espacio de configuraciones. El espacio de configuraciones o C-space (del inglés) es el espacio de todas las posibles configuraciones del robot.
- Dimensión del espacio de configuraciones. Es igual al número de variables independientes en la representación de la configuración, tambien conocidos como los **grados de libertad** del robot, (DOF por sus siglas en inglés).

Un problema crítico es que para la mayoría de los robots manipuladores el espacio de configuraciones es de alta dimensión y su cálculo explícito es computacionalmente costoso.

A.2. Convención Denavit-Hartenberg

El problema de cinemática directa trata con la relación entre las articulaciones individuales del robot manipulador y la posición y orientación del efector final. Más formalmente, es determinar la posición y orientación del extremo de un robot, con respecto a un sistema de coordenadas de referencia, una vez conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.

Un brazo robótico es un conjunto de eslabones conectados por articulaciones, ver figura A.1. Donde

a) Cada eslabón es un cuerpo rígido.

b) Cada eslabón está conectado a lo más a dos articulaciones.

c) Cada articulación tiene un grado de libertad.

d) Cada articulación conecta a dos eslabones.

e) No hay ciclos.

Las articulaciones son numeradas de manera creciente desde la base del robot, así el primer eslabón es 0 y la primera articulación es 1.

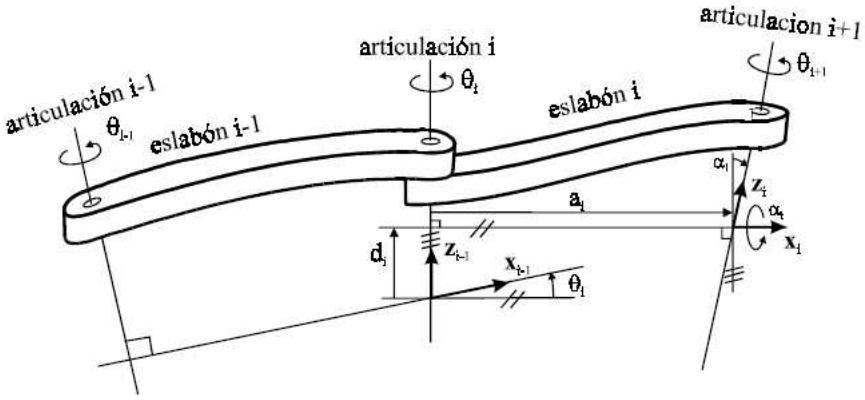


Figura A.1: Asignación de los marcos de referencia en la convención Denavit-Hartenberg

La representación Denavit-Hartenberg[1] se ha convertido en la forma estándar para representar y modelar los movimientos de un robot. El método inicia con un enfoque sistemático

[1]Información detallada en la referencia [47]

al asignar y etiquetar un sistema de coordenadas ortonormal (x, y, z) a cada articulación del robot. Entonces es posible relacionar una articulación con la siguiente, será posible pasar de una a la siguiente mediante cuatro transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Estas transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permitan relacionar el sistema de referencia del elemento i con el sistema del elemento $i-1$. Las transformaciones en cuestión son las siguientes:

- Rotación alrededor del eje Z_{i-1} un ángulo θ.
- 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, un ángulo α_i .

Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$^{i-1}A_i = R_{z,\theta_i} Tras_{z,d_i} Tras_{x,a_i} R_{x,\alpha_i}$$

Realizando el producto de matrices obtenemos:

$$\begin{aligned}
^{i-1}A_i &= R_{z,\theta_i} Tras_{z,d_i} Tras_{x,a_i} R_{x,\alpha_i} \\
&= \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ -S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & -S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Los cuatro parámetros a_i, α_i, d_i y θ_i son conocidos como, longitud del eslabón, torcedura del eslabón, desplazamiento del eslabón y ángulo de la articulación, respectivamente.

Las transformaciones con esos cuatro parámetros deben cumplir con las siguientes dos restricciones:

DH-1: X_i es perpendicular a Z_{i-1}. DH-2: X_i intercepta a Z_{i-1}.

A.2.1. Determinación de los parámetros y asignación de marcos

d_i Desplazamiento del eslabón, distancia desde O_{i-1} a la intersección de Z_{i-1} con X_i sobre el eje Z_i (asociado a la articulación i).

θ_i Es el ángulo entre X_{i-1} y X_i sobre el eje Z_{i-1} (usando la regla de la mano derecha.)

α_i Torcedura del eslabón, ángulo desde Z_{i-1} a Z_i con respecto al eje X_i.

a_i Longitud del eslabón, es la más corta distancia entre Z_{i-1} y Z_i o distancia desde la intersección de Z_i con X_i al origen O_i a lo largo de Z_i.

Para cada articulación i asignar Z_{i-1} como el eje de actuación para la articulación i, lo siguiente es asignar O_i y X_i, (dado que Y_i puede obtenerse por la regla de la mano derecha). Para asignar X_i y determinar a_i vemos las relaciones entre los eslabones, la articulación i conecta el eslabón $i-1$ al eslabón i, dado que el eslabón es rígido, la relación espacial entre la articulación i e $i+1$ nunca cambia.

Para la asignación de O_i y X_i se tienen dos casos:

- Coplanares
  - Z_{i-1} y Z_i se intersectan.
  - $a_i = 0$
  - O_i es el punto de intersección.
  - $X_i = \pm \frac{Z_{i-1} \times Z_i}{\|Z_{i-1} \times Z_i\|}$
- No coplanares
  - Z_{i-1} no es coplanar con Z_i
  - α_i ángulo desde Z_{i-1} a Z_i sobre X_i
  - a_i más corta distancia entre ellos
  - O_i intersección de la más corta distancia entre ellos
  - X_i dirigido a lo largo de las más cortas líneas

A.3. Especificaciones del robot

Como ya se mencionó se trabaja con un robot manipulador móvil, figura A.2. El robot se compone de una base móvil y un brazo que se crea de una secuencia de eslabones y articulaciones. Los eslabones son los miembros rígidos que conectan las articulaciones, o ejes. Los ejes son componentes móviles del robot que causan movimiento relativo entre los eslabones contiguos. Entre estos eslabones el último se le conoce como muñeca y es donde comúnmente se implanta el sensor, también se le denomina efector final.

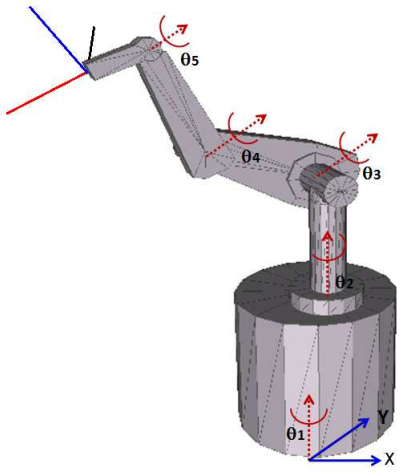


Figura A.2: Robot manipulador móvil

Hay una gran variedad de sensores, incluyendo sensores táctiles, sensores de visión, láser, etc. En nuestro trabajo vamos a suponer un sistema robot-sensor denominado "eye in hand", es decir, contamos con un sensor en el efector final. El sensor es de tiempo discreto, es decir sensa el ambiente solo en determinados instantes de tiempo.

El robot cuenta con siete grados de libertad. Dos grados de libertad correspondientes a la posición de la base (x, y), un grado de libertad de rotación de la base y cuatro grados de libertad de rotación de las articulaciones del brazo.

El ambiente en 3D está conformado por poliedros y se considera estático, donde los únicos agentes móviles son el robot y, ocasionalmente, un óbstaculo.

A.4. Otros conceptos útiles

- Algoritmo completo. En la solución de un problema, el algoritmo propuesto se llamará completo, si es capaz de encontrar una solución en un tiempo acotado si esta existe, sino existiera ninguna solución al problema, el algoritmo es capaz de reportar que dicha solución no existe.

- Algoritmo probabilísticamente completo. Un algoritmo es considerado probabilísticamente completo cuando la probabilidad de encontrar una solución tiende a uno cuando el número de las muestras generadas tiende a infinito. En general su ventaja radica en ser práctico y relativamente fácil de implementar.

- Método local. Dadas dos configuraciones q y q' del sistema, un método local se refiere a un algoritmo que calcula un camino factible $L(q, q')$ que conecte ambas configuraciones.

- Roadmap. Un roadmap es un grafo cuyos nodos son configuraciones libres de colisión. Dos nodos q y q' son adyacentes si el camino $L(q, q')$ calculado por el método local se encuentra en el espacio de configuraciones libres (del inglés CS_{free}). Las aristas corresponden a estos caminos que conectan dos configuraciones libres de colisión.

- Dominios o regiones de visibilidad. Para un método local L, el dominio de visibilidad de una configuración q se define como:

$$Vis_L(q) = \left\{ q' \in CS_{free} | L(q, q') \subset CS_{free} \right\}$$

  A la configuración q se le denomina el guardia de $Vis_L(q)$. Dos configuraciones mantienen visibilidad si ambos caen en el mismo dominio de visibilidad.

- Cobertura del espacio de configuraciones libres. Un conjunto de guardias constituye una cobertura de CS_{free} si la unión de sus dominios de visibilidad cubren el espacio libre. La existencia de una cobertura finita depende de la forma del CS_{free} y del método local. Tal cobertura finita puede no existir.

- Partición. Siguiendo la definición matemática, la familia de subconjuntos $A_i \ : \ i = 1, , 2, 3, ..., n$ de un conjunto A es una partición (sobre A) si se cumple que:

  1. $A_i \neq \emptyset, \quad i = 1, \ldots, n$
  2. $\bigcup_{i=1,\ldots,n} A_i = A$
  3. $A_i \bigcap A_j \neq \emptyset \Rightarrow A_i = A_j, \quad i, j = 1, \ldots, n$

Por lo tanto, se trata de un recubrimiento en el que los subconjuntos pertenecientes a la familia, dos a dos, son disjuntos (es decir, su intersección es vacía). Si el concepto lo asimilamos con respecto a una descomposición del ambiente en regiones, una partición será un conjunto de regiones que 1) contienen una porción del ambiente, 2) la unión de todas las regiones en el conjunto serán el ambiente y 3) la intersección dos a dos de las regiones es vacía. La diferencia fundamental entre una cobertura y una partición del espacio radica en que las regiones de la cobertura pueden compartir áreas en común, es decir su intersección no es el vacío.

- Distancia Manhattan o distancia L_1 es una métrica en la cual la distancia entre dos puntos es la suma de las diferencias (absolutas) de sus coordenadas. Formalmente, la distancia L_1 entre dos vectores p y q en un espacio vectorial real n-dimensional con un sistema de coordenadas cartesianas fijo es la suma de las longitudes de las proyecciones del segmento de línea entre los puntos sobre el sistema de ejes de coordenadas. [32]

Bibliografía

[1] E. U. Acar and H. Choset. "Exploiting critical points to reduce position error for sensor-based navigation". *IEEE International Conference on Robotics and Automation*, 1:3831–3837, 2002.

[2] T. Basar and G. J. Olsder. "Dynamic noncooperative game theory". *Classics In Applied Mathematics*, 1982.

[3] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson. "Optimal paths for landmarkbased navigation by differential drive vehicles with field-of- view constraints". *IEEE Transactions on Robotics*, 23:47 – 59, Feb 2007.

[4] D. Bienstock and P. D. Seymour. "Monotonicity in graph searching". *Journal of Algorithms*, 12:239–245, 1991.

[5] J. Bittner and P. Wonka. "Visibility in computer graphics". *Journal of Environment and Planning B: Planning and Design*, 5(30):729–756, 2003.

[6] F. Bourgault, T. Furukawa, and H. Durrant-whyte. "Coordinated decentralized search for a lost target in a bayesian world". *International Conference on Intelligent Robots and Systems*, 1:48–53, 2003.

[7] H. Brönnimann and M.T. Goodrich. "Almost optimal set covers in finite vc-dimension". *Discrete and Computational Geometry*, 14:463–479, 1995.

[8] J. F. Canny. *"The Complexity of Robot Motion Planning"*. The MIT Press, 1988. ACM doctoral dissertation awards, 1987.

[9] W.-P. Chin and S. C. Ntafos. "Shortest watchman routes in simple polygons". *Discrete Computational Geometry*, 6:9–31, 1991.

[10] H. Choset, K. M. Lynch, S. Hutchinson, W. Burgard G. Kantor, L. E. Kavraki, and S. Thrun. *"Principles of Robot Motion: Theory, Algorithms, and Implementations"*. The MIT Press, Boston, USA, 2005.

[11] F. Durand. "3D visibility: Analytical study and applications". *Ph.D. thesis, Universite Joseph Fourier*, Jul 1999.

[12] F. Durand, G. Drettakis, and C. Puech. "The visibility skeleton: A powerful and effcient multi-purpose global visibility tool". *Computer Graphics, (Proceedings of SIGGRAPH '97)*, 1:77–88, 1997.

[13] J. Espinoza-León and R. Murrieta-Cid. "Repairing plans for object finding in 3-D environments". *IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS 2011*, pages 4528 – 4535, 2011.

[14] J. Espinoza-León and R. Murrieta-Cid. "A motion planer for finding and object in 3-D environments with a mobile manipulator robot equipped with a limited sensor". *IBERAMIA10, LNCS*, 6433:532–541, Springer, 2010.

[15] J. Espinoza-León, A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. "A motion planning strategy for finding and object with a mobile manipulator in three-dimensional environments". *Advanced Robotics*, 25(13–14):1627–1650, August, 2011.

[16] M. R. Garey and D. S. Johnson. *"Computers and Intractability"*. W. H. Freeman and Company, 1979.

[17] H. González-Banos and J. Latombe. "A randomized art-gallery algorithm for sensor placement". In *Symposium on Computational Geometry*, 2001.

[18] J. E. Goodman and J. O'Rourke. *"Handbook of Discrete and Computatational Geometric"*. CRC Press, 1997.

[19] G. R. Guerequeta and M. A. Vallecillo. "Técnicas de diseño de algoritmos". *Servicio de Publicaciones de la Universidad de Málaga*, 1998.

[20] L. J. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. "Visibility-based pursuit-evasion in a polygonal environment". *5th Workshop on Algorithms and Data Structures*, 1272:17–30, 1997.

[21] L. J. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. "Visibility-based pursuit-evasion problem". *International Journal of Computational Geometry and Applications*, 9:471–494, 1999.

[22] P. J. Heffernan. "An optimal algorithm for the two guards problem". *International Journal of Computational Geometry and Applications.*, 6:15–44, 1996.

[23] J. P. Hespanha, M. Prandini, and S. Sastry. "Probabilistic pursuit-evasion games: A one-step nash approach". In *Conference on Decision and Control*, 2000.

[24] J. Hopcroft, J. Schwartz, and M. Sharir. *"Planning, geometry, and complexity of robot motion "*. Ablex Publishing, Norwood, NJ, USA, 1987.

[25] S. A. Hutchinson and A. C. Kak. "Planning sensing strategies in a robot work cell with multi-sensor capabilities". *IEEE Transactions on Robotics and Automation*, 5(6):765–783, Dic 1989.

[26] C. Icking and R. Klein. "The two guards problem". *Symposium on Computational Geometry*, 1:166–175, 1991.

[27] R. Isaacs. *"Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization"*. John Wiley and Sons, Inc, 1965.

[28] V. Isler, S. Kannan, and S. Khanna. "Locating and capturing an evader in a polygonal environment". *Algorithmic Foundations of Robotics VI*, 17/2005:251–266, 2005.

[29] V. Isler, S. Kannan, and S. Khanna. "Randomized pursuit-evasion in a polygonal environment". *IEEE Transactions on Robotics*, 21(5):875–884, Oct 2005.

[30] B. Jung and S. Sukhatme. "Tracking targets using multiple robots: the effect of environment occlusion". *Autonomous Robots*, 13(3):191–205, 2002.

[31] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". *IEEE Transations on Robotics and Automation*, 12(4):566–580, Ago 1996.

[32] E. F. Krause. *"Taxicab geometry: An Adventure in non-euclidean geometry"*. New York, 1986.

[33] S. Lacroix, P. Grandjean, and M. Ghallab. "Perception planning for a multi-sensory interpretation machine". *IEEE International Conference on Robotics and Automation*, 2:1818–1824, May 1992.

[34] A. H. Land and A. G. Doig. "An automatic method of solving discrete programming problems". *Econometria*, 28(3):497–520, 1960.

[35] A. S. Lapaugh. "Recontamination does not help to search a graph". *Journal of The ACM*, 40:224–245, 1993.

[36] P.-S. Laplace. *"Théorie Analityque des Probabilitiés".* Courceir, Paris, 1812.

[37] J. C. Latombe. *"Robot motion planning".* Kluwer, Dordrecht, The Netherlands, 1991.

[38] H. Lau, S. Huang, and G. Dissanayake. "Optimal search for multiple targets in a built environment". *International Conference on Intelligent Robots and Systems*, 1:3740–3745, 2005.

[39] S. M. LaValle. *"Planning Algorithms".* Cambridge University Press, Cambridge, N.Y., USA, 2006. Free of charge: http://msl.cs.uiuc.edu/planning/.

[40] S. M. LaValle, H. González, C. Becker, and J. C. Latombe. "Motion strategies for maintaining visibility of a moving target". *IEEE International Conference on Robotics and Automation*, 1:731–736, Abr 1997.

[41] S. M. LaValle and J. Hinrichsen. "Visibility-based pursuit-evasion: the case of curved environments". *IEEE International Conference on Robotics and Automation*, 17(2):196–202, Abr 2001.

[42] S. M. LaValle and J. J. Kuffner. "Randomized kinodynamic planning". *The International Journal of Robotics Research*, 20(5):378–400, May 2001.

[43] S. M. LaValle, D. Lin, L. C. Guibas, J. C. Latombe, and R. Motwani. "Finding an unpredictable target in a workspace with obstacles". *IEEE International Conference on Robotics and Automation*, 1:737–742, Ago 1997.

[44] A. Lazanas and J. C. Latombe. "Landmark-based robot navigation". *Algorithmica*, 13(5):472–501, 1995.

[45] D. T. Lee and A. K. Lin. "Computational complexity of art gallery problems". *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

[46] T. Lozano-Perez. "Spatial planning: A configuration space approach". *IEEE Transations on Computers*, 32(2):108–120, 1983.

[47] S. Hutchinson M. W. Spong and M. Vidyasagar. *"Robot Modeling and Control"*. John Wiley and Sons, 2006.

[48] F. Makedon and I. H. Sudborough. "Minimizing width in linear layouts". *Automata, Languages and Programming*, 154/1983:478–490, 1983.

[49] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. "The complexity of searching a graph". *Journal of The ACM*, 35:18–44, 1988.

[50] B. Monien and H. Sudborough. "Min cut is NP-complete for edge weighted trees". *Theoretical Computer Science*, 58(1-3):209–229, Jun 1988.

[51] R. Murrieta, A. Sarmiento, and S. A. Hutchinson. "On the existence of a strategy to maintain a moving target within the sensing range of an observer reacting with delay. *IEEE/RSJ International Conference on Intelligent Robots and Systems"*., 1:1184–1191, 2003.

[52] R. Murrieta-Cid, H. H. González-Banos, and B. Tovar. "A reactive motion planner to maintain visibility of unpredictable targets". *IEEE International Conference on Robotics and Automation*, 1:4242–4248, 2002.

[53] R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S Hutchinson. "Surveillances strategies for a pursuer with finite sensor range". *Int. Journal of Robotics Research*, 26(3):233–253, 2007.

[54] R. Murrieta-Cid, C. Parra, and M. Devy. "Visual navigation in natural environments: From range and color data to a landmark-based model". *Journal Autonomous Robots*, 13(2):143–168, Sep 2002.

[55] R. Murrieta-Cid, B. Tovar, and S. Hutchinson. "A sampling-based motion planning approach to maintain visibility of unpredictable targets". *Journal Autonomous Robots*, 19(3):285–300, 2005.

[56] J. O'Rourke. "The complexity of computing minimum convex covers for polygons". *In 20th Annual Allerton Conference on Communication, Control and Computing*, 1982.

[57] J. O'Rourke. *"Art gallery theorems and algorithms"*. Oxford University Press, New York, NY., 1987.

[58] J. O'Rourke. *"Computational geometry in C"*. Cambridge University Press, 1994.

[59] T. D. Parsons. "Pursuit-evasion in a graph". *Theory and Application of Graphs*, 642/1978:426–441, 1978.

[60] H. Plantinga and C. Dyer. "Visibility, occlusion, and the aspect graph". *International Journal of Computer Vision*, 5(2):137–160, 1990.

[61] K. Tanie Q. Huang and S. Sugano. "Stability compensation of a mobile manipulator by manipulator motion: feasibility and planning". *Advance Robotics*, 13(6-8):25–40, 1999.

[62] S. Sachs, S. M. Lavalle, and S. Rajko. "Visibility-based pursuit-evasion in an unknown planar environment". *International Journal of Robotic Research*, 23:3–26, 2004.

[63] M. Saha, J. C. Latombe, Y. C. Chang, and F. Prinz. "Finding narrow passages with probabilistic roadmaps: The small-step retraction method". *Autonomous Robots*, 19(3):301–319, 2005.

[64] M. Saha, T. Roughgarden, J.C. Latombe, and G. Sánchez. "Planning tours of robotic arms among partitioned goals". *Int. Journal on Robotics Research*, 25:207–223, 2006.

[65] M. Saha, G. Sánchez, and J.C. Latombe. "Planning multi-goal tours for robot arms". *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA 2003)*, 2003.

[66] G. Sánchez and J. C. Latombe. "A single-query bi-directional probabilistic roadmap planner with lazy collision checking". *Proc. Int. Symp. on Robotics Research (ISRR01), Lorne*, pages 403–417, 2003.

[67] G. Sánchez and J.C. Latombe. "On the delaying collision cheching in PRM planning". *Int. Journal on Robotics Research*, 21(1):5–26, 2002.

[68] A. Sarmiento, J. Espinoza-León, R. Murrieta-Cid, and S. Hutchinson. "A motion planning strategy for rapidly finding and object with a mobile manipulator in 3-D environments". *MICAI 2008, LNAI*, 4827:562–572, 2008.

[69] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. "An efficient strategy for rapidly finding an object in a polygonal world". *IEEE/RSJ International Conference on Intelligent Systems (IROS)*, 1:1153–1158, 2003.

[70] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. "A multi-robot strategy for rapidly searching a polygonal environment". *Advances in Artificial Intelligence IBERAMIA 2004*, 3315/2004:484–493, 2004.

[71] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. “Planning expected-time optimal paths for searching known environments”. *IEEE/RSJ International Conference on Intelligent robots and Systems, (IROS)*, 1:872–878, 2004.

[72] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. “A sample-based convex cover for rapidly finding an object in a 3-D environment”. *IEEE International Conference on Robotics and Automation*, 1:3486–3491, 2005.

[73] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. “An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environment”. *Advanced Robotics*, 23(12-13):1533–1569, 2009.

[74] T. C. Shermer. “Recent results in art galleries”. *Proceedings of The IEEE*, 80(9):1384–1399, Sep 1992.

[75] K. Shubina and J. K. Tsotsos. “Visual search for an object in a 3D environment using a mobile robot”. *Tecnical Report CSE-2008-02, Department of Computer Science and Engineering, York University*, 2008.

[76] I. Suzuki and M. Yamashita. “Searching for a mobile intruder in a polygonal region”. *SIAM Journal on Computing*, 21(5):863–888, Oct 1992.

[77] J. P. Laumond T. Simeon and C. Nissoux. “Visibility based probabilistic roadmaps”. *Advanced Robotics*, 14(6):477–493, 2000.

[78] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. “A survey of sensor planning in computer vision”. *IEEE Transactions on Robotics and Automation*, 11:86–104, Feb 1995.

[79] B. Tovar, L. Munoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, Raúl Monroy, and Seth Hutchinson. ”Planning exploration strategies for simultaneous localization and mapping”. *Journal Robotics and Autonomous Systems*, 54(4):314–331, Abr 2006.

[80] L. Tseng, P. Hefferman, and D. Lee. “Two guards walkability of simple polygons”. *International Journal of Computational Geometry and Applications*, 8(1):85–116, 1998.

[81] J. K. Tsotsos and K. Shubina. “Attention and visual search: Active robotic vision systems that search”. In *International Conference on Computer Vision Systems*, 2007.

[82] J. Urrutia. “Art gallery and illumination problems”. *Handbook of Computational Geometry, Elsevier Science*, 1998.

[83] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry. "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation". *IEEE Transactions on Robotics and Automation*, 18(5):662–669, Oct 2002.

[84] K. Yamazaki, M. Tomono, and T. Tsubouchi. "Pose planning for a mobile manipulator based on joint motions for posture adjustment to end-effector error". *Advance Robotics 22*, 22:411–431, 2008.

[85] Y. Ye and J. K. Tsotsos. "Sensor planning for 3D object search". *Computer Vision and Image Understanding*, 73(2):145–168, 1999.