Benemérita

Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

" Exploración basada en sensores para ambientes 2D y 3D "

Tesis Profesional

Que para obtener el grado de:

Maestro en Ciencias de la Computación

Presenta

Juan Manuel Morales Barragán

Asesor

Dr. Abraham Sánchez López

Puebla, Pue. Febrero 2013

Índice general

Introducción

En un mundo desconocido, habitado por obstáculos, un robot parte de un estado inicial, y tiene que explorar su mundo y construir un modelo que represente con similitud, los obstáculos habitados en dicho ambiente [6].

En aplicaciones de servicio, se requieren de sistemas robóticos para llevar a cabo determinadas tareas en entornos que son parcial o completamente desconocidos. La detección, planificación y ejecución de movimiento debe estar debidamente entrelazados con el fin de descubrir y navegar por las porciones del medio ambiente que son relevantes para la tarea asignada. Se tiene experiencia en la exploración de robots móviles, con el método de exploración SRT (Sensor-based Random Tree), además de que hemos presentado en distintos foros, contribuciones importantes en esta línea.

Un punto interesante en el trabajo realizado, es que no se trabajó solo con robots móviles (ambientes 2D), sino que la propuesta se llevó más allá de esta restricción, es decir la consideración de ambientes 3D (que es el caso de los manipuladores y los manipuladores móviles). Es el caso de los espacios 3D, se llevo a cabo la extensión de dicho método, gracias a que dicho sistema de sensado de los robots permitió construir la región segura local, o un espacio circunvecino.

En el sistema se desarrolló un método para la exploración basada en sensores de entornos desconocidos para un sistema robótico equipado con medidores de distancia (es decir, puede ser un robot móvil, un robot manipulador o un robot manipulador móvil). El método se basa en la generación incremental de una estructura de datos llamado árbol aleatorio de exploración (AAE). La expansión del AAE se efectúa gracias a la información del ambiente, donde el proceso de percepción tiene lugar. En particular, los limites de la región explorada se utilizan para guiar la búsqueda de las configuraciones candidatas desde el punto de vista informativo [6].

La planificación de movimientos y navegación son componentes muy importantes en la operación de robots móviles autónomos. El problema de navegación de movimientos trata con trayectorias libres de colisiones para un robot desde una posición origen a una posición

destino en un ambiente poblado de obstáculos. Los algoritmos de navegación conocidos a menudo se refieren a aproximaciones de planificación de movimientos basados en sensores.

Varios métodos se han desarrollado en planificación de movimientos (MP) y planificación de movimientos basados en sensores (SBMP). No obstante, muchas investigaciones han dirigido estos problemas separadamente. Los métodos de planificación de movimientos están basados en el completo conocimiento del robot y del ambiente [5].

Estos métodos construyen algunas trayectorias (conjunto de sub-metas) libres de obstáculos. Sus principales ventajas son las de proveer la existencia de una solución que permita al robot la búsqueda de su destino y genere la construcción de un mapa libre de colisiones. Sin embargo, tienen algunos inconvenientes, por ejemplo, se necesita un modelo exacto del ambiente. Por lo cual es difícil manejar correctamente modificaciones al ambiente cuando se tienen nuevos objetos u objetos dinámicos [5].

La exploración de ambientes desconocidos puede considerarse como un problema fundamental para robots móviles, dado que involucra todas las capacidades fundamentales de tales sistemas, por ejemplo, la percepción, la planificación, la localización y la navegación. Desde un punto de vista práctico, la exploración es una tarea central en aplicaciones tales como misiones planetarias, intervenciones en áreas hostiles y construcción automática de mapas [5].

Capítulo I

1. Estado del Arte

Hay dos formulaciones básicas del problema de la planificación de caminos y la navegación basadas en la disponibilidad del modelo del ambiente. En un ambiente conocido, el modelo se da como entrada, así el problema de planificación de movimientos se convierte en uno de programación geométrica. En un ambiente desconocido, la ausencia del modelo requiere que el robot obtenga información local del ambiente empleando un sistema de sensado. Una de las principales diferencias entre ambas formulaciones, radica en que, el camino para llegar de una ubicación origen a una ubicación destino puede ser pre-planeado antes de ejecutar cualquier movimiento. En el último caso, el camino debe ser calculado incrementalmente por medio de la exploración de nuevas zonas del ambiente.

En un ambiente desconocido tenemos dos aspectos críticos: a) el cómputo se sustenta en información local (o parcial) y b) el sensado es parte integral de la navegación. A causa del primer aspecto, los algoritmos para ambientes desconocidos con frecuencia se les llama algoritmos en línea [22]. Como lo vislumbra el segundo aspecto, en un ambiente desconocido, el algoritmo es necesario para catalogar las operaciones del sensor. En general, en ambientes desconocidos distintos y con robots navegadores equipados con diferentes tipos de sensores se requieren diversos algoritmos.

En el presente capítulo, describiremos brevemente algunos métodos de exploración y planificación de movimientos en ambientes desconocidos que han sido parte fundamental en el desarrollo de nuestro proyecto de investigación. Algunos de estos métodos fueron creados para sistemas de robots manipuladores y otros para robots móviles, pero todos ellos contienen puntos claves para moldear nuevos métodos de exploración. Este capítulo también puede considerarse como una introducción, ya que posteriormente reportaremos otros métodos de exploración en ambientes desconocidos como resultado de nuestro trabajo.

1.1. SBIC-PRM

La investigación hecha por Yu y Gupta [15], [20] para la planificación de movimientos usando sensores, presenta un sistema real "Eye-in-hand" para un brazo manipulador con muchos grados de libertad. Ellos exploran la adaptabilidad del método PRM para este sistema, construyendo incrementalmente el roadmap con la información reportada por los sensores del espacio libre en el ambiente físico, este roadmap representa la conectividad del espacio libre conocido, dentro del cual, el robot puede moverse para detectar más espacio físico. A este método le llamaron SBIC-PRM (del inglés, sensor-based incremental construction of probabilistic roadmap)[19].

1.1.1. Algoritmo SBIC-PRM

Dado un espacio parcialmente conocido, el espacio de configuraciones del robot implícitamente se divide en tres regiones. La región blanca (el espacio libre, C_{libre}), la región negra (los obstáculos, C_{obs}) y la región gris que corresponde al espacio desconocido. Un conjunto de configuraciones se genera aleatoriamente, de tal forma que se crean tres tipos de configuraciones (denominadas marcas) de acuerdo a la región, así tenemos marcas blancas, negras y grises, ver figura 1.1.

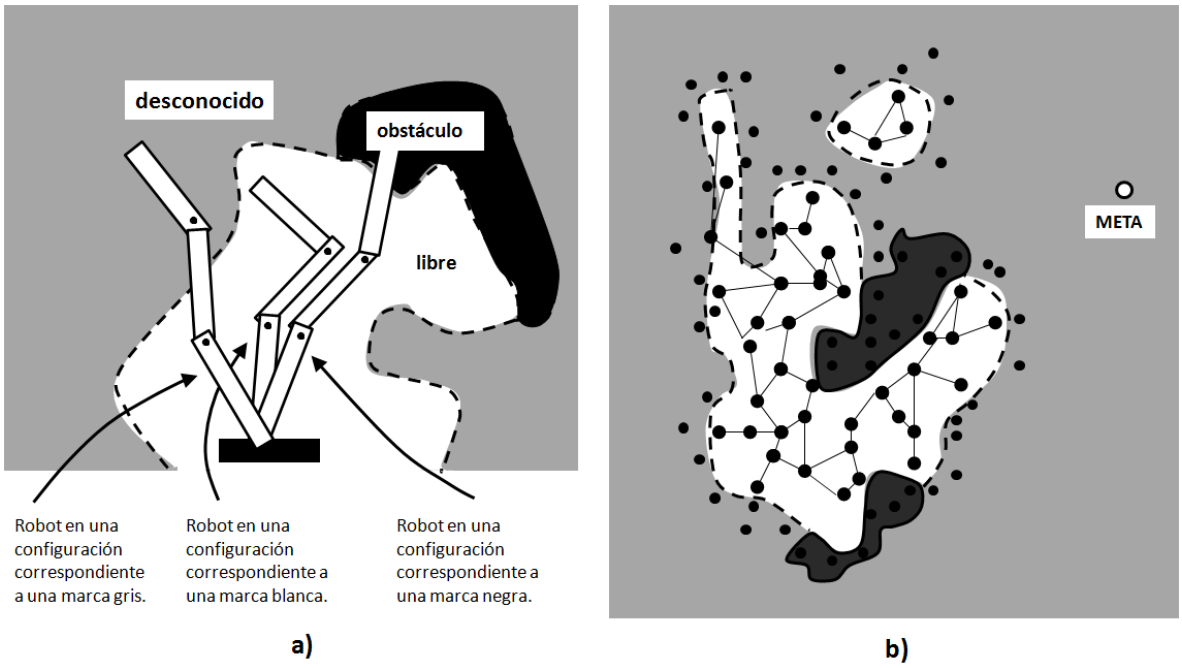


Figura 1: a) Ejemplo de marcas gris, blanca y negra en el espacio físico. b) Los tres tipos de marcas expuestas sobre el espacio de configuraciones. La región blanca es el C_{libre} conocido, las marcas sobre esta región son blancas. El roadmap es el grafo que se muestra dentro de esta región. La región obscura indica el espacio conocido de los obstáculos, C_{obs} , las marcas aquí son negras. La región gris es el C-espacio desconocido en ella las marcas son grises.

En principio, el robot se encuentra en la configuración inicial dentro de un espacio libre inicial. Existe un planificador que genera el correspondiente roadmap. De ahí, el paso clave del algoritmo es expandir incrementalmente el roadmap R que representa la conectividad del actual espacio libre. Para cualquier iteración dada, la i-ésima, el planificador, en primer lugar, intenta alcanzar la configuración meta ejecutando el planificador local desde todas las marcas en la región blanca. Si existe alguna que regrese éxito, el planificador habrá encontrado un camino libre de colisión. Si no sucede, el planificador toma un nuevo escaneo desde una marca blanca. El planificador mueve al robot a este nodo de vista, para actualizar el espacio libre conocido, en sí no se calcula explícitamente el espacio físico libre, en vez de ello, se agregan más marcas de manera que el roadmap se expanda. El proceso completo puede pensarse como un frente de onda del espacio libre, representado por las marcas grises, dirigido hacia la configuración meta. Una vez

actualizado el roadmap, el planificador nuevamente intenta alcanzar la meta desde todas las marcas blancas y el proceso entero se repite, hasta que la meta es alcanzada o el planificador realiza un número máximo de iteraciones.

1.2. Plan-N-Scan

El sistema presentado por Renton et al. [24] llamado Plan-N-Scan, es muy similar en hardware al utilizado posteriormente por Yu y Gupta, trabaja con un robot PUMA el cual tiene montado en la "muñeca" un sensor de alcance.

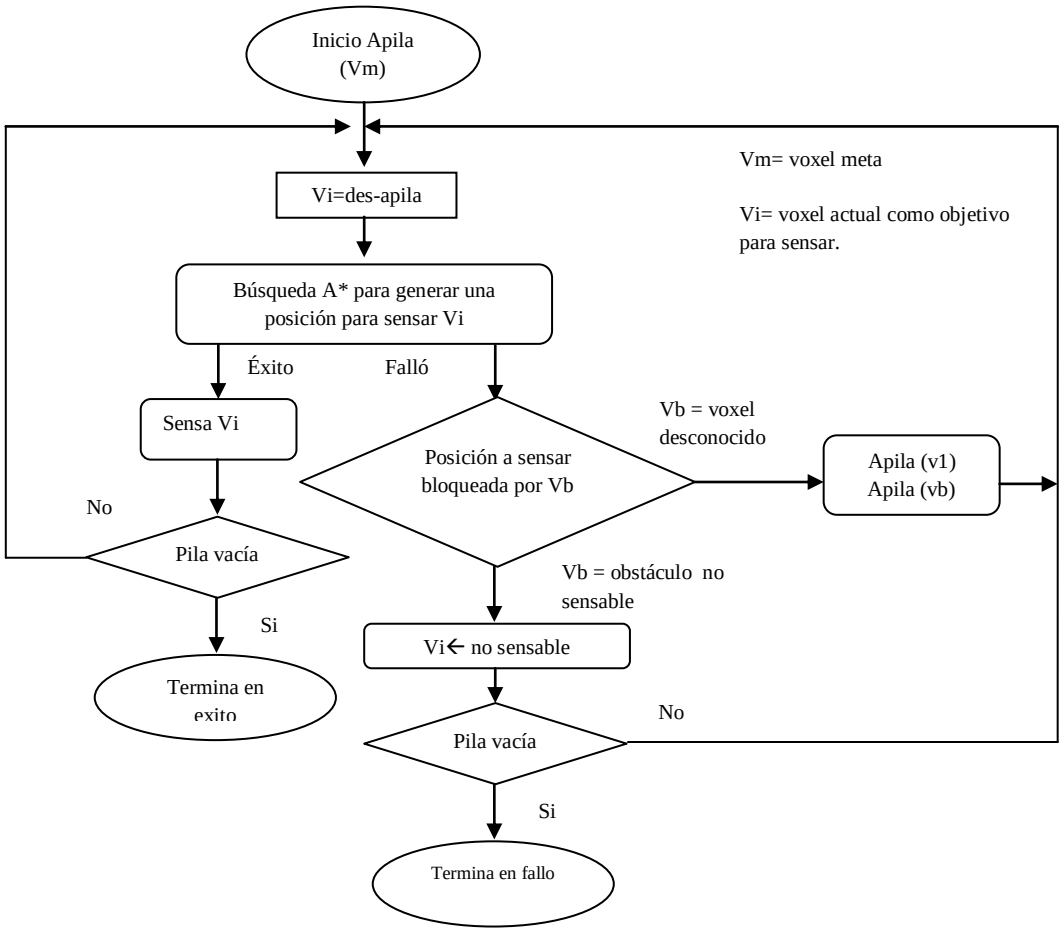


Figura 2: Diagrama de flujo del algoritmo Plan-N-Scan

El sistema incrementalmente sensa una secuencia de objetivos seleccionados de acuerdo a sensados previos. El resultado es un "voxel map", una malla tridimensional, conveniente para el chequeo de colisiones. La planificación de movimientos se apoya en un algoritmo A*.

Plan-N-Scan mantiene una estructura de datos tipo pila donde almacena el objetivo final y objetivos intermedios (voxels que serán sensados). La pila se inicializa con el voxel meta y en cada iteración el algoritmo extrae un voxel sub-meta. El planificador primero selecciona una posición de vista para el voxel sub-meta y planifica un camino libre de colisión o determina si se requieren objetivos adicionales donde sensar y así alcanzar la sub-meta [16].

Las posiciones de vista tienen que satisfacer dos criterios: (1) la línea de visión del sensor al objetivo no puede estar obstruido y (2) el manipulador debe estar libre de colisión cuando se encuentre en la posición de vista, si ninguna posición cumple con los criterios entonces se puede utilizar la búsqueda A* o declarar a la sub-meta no sensable. Una vez que se tienen la posición de vista, se invoca una búsqueda A* para encontrar un camino libre de colisión a la posición objetivo, si esto sucede entonces el voxel sub-meta es extraído de la pila, si la posición objetivo no puede alcanzarse debido a colisiones potenciales con el espacio desconocido, entonces se insertan en la pila objetivos intermedios. El algoritmo termina con éxito cuando la pila está vacía y el sistema alcanzó el objetivo final. Podría suceder que la pila este vacía debido a que el voxel meta se haya declarado no sensable, por lo que el algoritmo termina en fallo. La figura 2 muestra el diagrama de flujo del planificador [16].

1.3. Algoritmo de exploración utilizando funciones de valoración

Kruse, Gutschet y Wahl describen un sistema similar al método anterior (sólo en simulación) [10]. Su algoritmo de planificación de caminos también construye un roadmap utilizando campos de potencial. Generalmente, no es necesaria la ejecución

del algoritmo de planificación, ya que la configuración a la que se mueve el robot a partir de la configuración actual puede alcanzarse directamente a través de un camino lineal en el espacio de configuraciones. Después de cada sensado, el grafo sólo se extiende, en vez de recalcularse completamente. Heurísticamente examinan los alrededores de la configuración donde se efectuó el sensado, debido a que sólo utilizan cinemática di-recta es difícil localizar nuevas áreas libres en el C-espacio. Tras varias extensiones, el grafo tiende a alargarse y degenerar. Por lo tanto, si la planificación de caminos falla, se requiere recalcular completamente el grafo. Esto ´ultimo no pasa con el método de SBIC-PRM.

Dichos autores dan mayor atención al algoritmo de planificación de vista, es decir, planificar la siguiente configuración donde se realizará el próximo sensado. Para la planificación de vista utilizan restricciones sobre los movimientos del robot y funciones de valoración en el C-espacio. Las restricciones son: a) La siguiente configuración de vista está libre de colisión; b) La siguiente configuración de vista puede alcanzarse por un camino libre de colisión; c) El sensado en la siguiente configuración de vista debería obtener mucha información nueva del área desconocida. Esta restricción se manipula introduciendo una función de valoración $R_{inf}(q)$, la cual es la fracción del volumen desconocido en el volumen total sensado. d) La siguiente configuración de vista no debería estar lejos de la actual. Esta restricción se observa a través de la función $R_{dist} = 1 - d(q, q_{act})/d_{max}$, donde $\mathrm{q_{act}}$ es la configuración actual del robot, $\mathrm{d(q, q_{act})}$ es una función que calcula la distancia entre las configuraciones y d_{max} es la distancia máxima en el C-espacio. La próxima configuración de vista se elige de modo que la función $R(q) = R_{inf}(q)(\omega R_{dist}(q) + 1 - \omega)$ se maximice sobre el C-espacio discreto, sujeta a las dos primeras restricciones, donde w $\in$[0, 1] es un parámetro de peso. La restricción sobre $\mathrm{R_{dist}}$ es critica sólo cuando el tiempo para moverse a cada nueva configuración de sensado toma una gran porción en la iteración escaneo-planificación-movimiento. La tercera restricción, R_{inf} es razonable pero no es suficiente.

1.4. Roadmaps probabilísticos usando sensores para robots tipo carro

Sánchez y Zapata proponen un método para el problema de planificación de movimientos usando sensores para robots tipo carro [15]. Adaptan los métodos PRM y lazy-PRM a la exploración de ambientes desconocidos para aprovechar la información reportada por los sensores y así calcular un camino factible libre de colisión.

Para el proceso de planificación tanto la configuración inicial como la final son nodos en el roadmap. El robot comienza en la configuración inicial donde se genera un lazy-roadmap inicial intentando hacer un muestreo del espacio de trabajo. Para cualquier iteración dada i, el planificador intenta alcanzar el nodo, más cercano a la configuración meta a través de un camino construido por un planificador local (caminos Reeds and Shepp). El paso clave es guiar la búsqueda con la información del sensor, usando una estrategia A*, y actualizar el grafo eliminando aristas que llevarían a una configuración en colisión. Un proceso asegura que tanto el nodo como la trayectoria están en un área libre de colisión para que el robot se mueva sin riesgo a chocar con un obstáculo.

Si hay éxito en calcular un camino factible el robot se mueve a la configuración y el proceso se repite consecutivamente hasta alcanzar la configuración meta o el algoritmo termina con fallo al exceder un tiempo máximo determinado. Una característica clave del algoritmo es la destreza dada al sistema para regresar a la configuración alcanzada si ya no es posible el avance hacia la configuración meta desde la configuración actual.

Las dos características más poderosas de este enfoque son las capacidades de escapar de mínimos locales, por medio del proceso de retroceso y salir con “fallo” sino se puede encontrar un camino.

1.5. GGV para la navegación de robots tipo carro usando sensores

Nagatani, Iwai y Tanaka desarrollaron un método para la navegación de robots tipo carro usando sensores [3] basándose en el grafo generalizado de Voronoi (GGV).

Desde el punto de vista de la completitud y seguridad el GGV tiene la ventaja de describir caminos para la navegación de robots móviles, sin embargo es imposible aplicarlos directamente debido a la limitación existente en el ángulo de conducción de tales robots. Para resolver este problema, proponen un planificador local de caminos suaves para deformar localmente el GGV usando una función de evaluación y así poder seguir los caminos trazados por el GGV lo más fiel posible [14].

Para realizar la navegación del robot móvil utilizando el GGV siguen el siguiente procedimiento: (1) Por medio de los sensores se adquiere la información local del ambiente; (2) Empleando un algoritmo convencional calculan el GGV local; (3) Se construye el espacio de configuraciones (C-espacio); (4) Se calculan candidatos de caminos suaves; (5) Aplicando una función de evaluación a cada camino candidato en el C-espacio se elige el mejor; y (6) Ejecutar el camino planeado en el área local. Los pasos 1 al 6 se repiten hasta que se construya completamente el GGV.

Usualmente, los movimientos de exploración del robot siguen el GGV. Cuando encuentra un punto de reunión (punto donde se encuentran tres o más aristas del GGV), el robot guarda la ubicación de este punto, y elige una arista inexplorada. Cuando encuentra un punto límite (punto donde dos o más obstáculos convexos son conectados), el robot regresa; si ya no hay aristas por explorar, la tarea de exploración termina.

A pesar de que el algoritmo encuentra caminos factibles para los robots tipo carro, la solución no es simple y el costo computacional es mayor que el enfoque convencional del GGV [14].

En el capítulo siguiente se aborda el método STR, diseñado para la exploración de ambientes desconocidos usando sensores para robots móviles omnidireccionales. Se presenta una variante del método junto a los resultados obtenidos en simulación.

1.6. Método SRT

La exploración de ambientes desconocidos puede considerarse como un problema fundamental para los robots móviles, dado que involucra todas las capacidades fundamentales de estos sistemas, por ejemplo, la percepción, la planificación, la localización y la navegación. Desde un punto de vista práctico, la exploración es una tarea central en aplicaciones tales como misiones planetarias, intervenciones en áreas hostiles, construcción automática de mapas, entre otras.

Una definición ampliamente aceptada sobre la exploración es la siguiente: "el acto de moverse a través de un ambiente desconocido mientras se construye un mapa que pueda utilizarse para subsecuentes navegaciones". El rendimiento de las estrategias de exploración debe ser valorado en base a la calidad del mapa obtenido y del tiempo necesario para construirlo. Muchas de las técnicas existentes caen dentro de la clase de exploración basada en fronteras. La lógica de este enfoque es que el robot debe moverse hacia los límites (la frontera) de las áreas seguras exploradas y del territorio desconocido para maximizar la información obtenida a través de nuevas percepciones [11].

Es interesante adoptar una perspectiva más general dentro de la Inteligencia Artificial, de acuerdo con la cual, la exploración es "el proceso de seleccionar acciones en aprendizaje activo". En el paradigma de aprendizaje activo, los datos de entrenamiento son obtenidos como resultado de las acciones del aprendiz. En particular, cuando el robot recopila información sobre su ambiente mediante movimiento y sensado, se considera como un caso de aprendizaje activo de orden sensitivo, porque el flujo de datos es resultado de todas las acciones pasadas del robot. El problema central de la exploración es cómo seleccionar la siguiente acción. La exploración basada en fronteras se logra cuando el criterio es la maximización de la utilidad de las acciones.

Sin embargo existe otra opción, es decir, usar un mecanismo de selección aleatoria (también llamado camino aleatorio). Las ventajas de esta elección son (1) simplicidad y (2) el hecho de que cualquier secuencia de acciones se ejecutará eventualmente. La ´ultima propiedad mencionada lleva a la completitud: se encontrará una solución cuando esta exista. Por otra parte, la selección de una acción puramente aleatoria puede ser muy ineficiente [11].

El enfoque del problema emana de las técnicas de planificación de movimientos aleatorios (PMA), las cuales construyen roadmaps en el espacio de configuraciones libre usando muestreo aleatorio y verificando las colisiones. En PMA, el problema planteado es de aprendizaje activo de orden-libre: lo que se observa del ambiente depende sólo de la última acción (búsqueda aleatoria), ya que el mapa del ambiente está disponible progresivamente y el robot se traslada para recopilar más información. Por lo cual, los planificadores aleatorios son considerados como estrategias de exploración orientadas a objetivos utilizando la selección aleatoria de acciones. Como ya se había mencionado, la completitud (probabilística) de estos planificadores es inherente a su naturaleza, además, se puede lograr una mayor eficiencia agregando algunas heurísticas al esquema aleatorio básico. El método RRT es un ejemplo típico [8].

El método de exploración implementado se basa en la generación aleatoria de configuraciones en un área segura local detectada por los sensores. Se crea una estructura de datos llamada árbol aleatorio de exploración usando sensores (SRT), el cual representa el roadmap del área explorada asociado a una región segura (RS). Cada nodo del SRT consiste de una configuración libre y su región segura local (RSL) asociada; la RS es simplemente la unión de todas las RSLs pertenecientes al árbol. La RSL es una estimación del espacio libre circunvecino a una configuración dada del robot; en general, su forma dependerá de las características del sensor pero también puede reflejar diferentes posturas de percepción.

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

1.7. Aportación

Este trabajo fue realizado con el objetivo de manejar una serie de métodos robustos para efectuar con éxito en un primer tiempo, la exploración de ambientes desconocidos basada en sensores para ambientes 2D y 3D.

En el capítulo 2 presentamos los algoritmos de exploración para ambientes de 2 dimensiones y profundizamos en el uso del algoritmo SRRT y cierta técnica de percepción. En el capítulo 3 movemos el algoritmo de los ambientes 2D a los ambientes 3D con el uso de manipuladores. Finalmente, la conclusión y discusión de trabajos a futuro se presentan con más detalle en el capítulo 4.

Capitulo 2

2. Algoritmos de exploración 2D

Oriolo, Vendittelli, Freda y Troso presentan en [12] un método de exploración de ambientes desconocidos usando sensores para un robot móvil. El método se basa en la generación de una estructura de datos incremental aleatoria llamada árbol aleatorio de exploración usando sensores (SRT, del inglés Sensor-Based Random Tree), la cual representa un roadmap del área explorada asociada a una región segura. Este método está inspirado en los árboles aleatorios de exploración rápida (RRTs), presentados en detalle en el capítulo dos. Pueden obtenerse diversas estrategias de exploración adaptando al método general, diferentes técnicas de percepción. En [12] se exponen y comparan dos técnicas; la primera, SRT-Ball, los autores la denominan una técnica conservadora y conveniente para usar sensores con ruido. La segunda técnica de percepción llamada SRT-Star es menos conservadora, es decir, confía más en la información reportada por los sensores.

2.1. Exploración de ambientes desconocidos con el método SRT

El método SRT se introdujo con ciertas consideraciones sobre el robot y el ambiente de trabajo. Más adelante se describe el método de exploración desde un punto de vista general, es decir, independiente de una estrategia de percepción particular. Finalmente, se presenta la variante adoptada y los resultados obtenidos por medio de la herramienta de simulación desarrollada.

2.1.1. Hipótesis de trabajo

El robot debe explorar un espacio de trabajo, es decir, un ambiente con obstáculos. Siguiendo las siguientes suposiciones:

1. El espacio de trabajo es plano, es decir $\mathbb{R}^2$ o un sub-conjunto de $\mathbb{R}^2$.

2. El robot es libre de trasladarse en cualquier dirección (un robot holonómico o robot de vuelo libre). De esta forma, el espacio de configuraciones es una copia del espacio de trabajo con los obstáculos crecidos tanto como lo requiera el tamaño del robot.
3. El robot siempre conoce su configuración q, (localización perfecta).
4. El robot está equipado con un sistema de sensores el cual provee en cada configuración q la estimación del espacio libre circunvecino. Esta estimación, llamada "Región Segura Local" en q, se denota por S.
5. Una pequeña región del espacio físico circundante al robot y al sistema de sensado en la configuración inicial es libre, este requerimiento es importante porque de lo contrario ningún movimiento puede ejecutarse después del primer sensado.

2.1.2. Algoritmo SRT

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

(y el camino que lleva a ella) este contenida en la región segura local. El algoritmo que implementa el método SRT se describe en la figura 3.

En cada iteración *k* del algoritmo, se efectúa un proceso de percepción (es decir, sensado del ambiente y recopilación de datos), para obtener la región S que estima el espacio libre circundante al robot en la configuración actual, q_{act}. Un nuevo nodo, que contiene la configuración q_{act} y su *RSL* asociada, se agrega al árbol *T*. La forma de representar S en la estructura SRT depende de la estrategia de percepción: en general, podría usarse una descripción algebraica de sus límites.

En el punto de la configuración actual, la función DIR_ALEATORIA genera una dirección aleatoria de exploración θ_{rand} y la función RADIO calcula el radio r de *S* en la dirección θ_{rand}, ver figura 4. Una nueva configuración candidata q_{act} se determina tomando un paso de longitud $\alpha \cdot r$ en dirección a θ_{rand}. La constante $\alpha < 1$ garantiza que q_{cand} se encuentra en el área segura *S* y puede alcanzarse a través de un camino contenido en *S*; valores próximos a 1 incrementan la capacidad de exploración del algoritmo, mientras que valores más pequeños aumentan el margen de seguridad [30].

CONSTRUIR_SRT(q_{ini}, K_{max}, I_{max}, α, d_{min}**)**

1. $q_{act} = q_{ini}$;
2. **para** k=1 **a** K_{max}
   1. S ← PERCEPCION(q_{act});
   2. AGREGA(T,(q_{act}, S));
   3. i←0;
   4. **repetir**
      1. θ_{rand} ← DIR_ALEATORIA;
      2. r← RADIO(S, θ_{rand});
      3. q_{cand} ← DESPLAZAR(q_{act}, θ_{rand},α·r);
      4. i←i+1;
   5. **hasta que** (VALIDA(q_{cand}, d_{min}, T) o i=I_{max})
   6. si VALIDA(q_{cand}, d_{min}, T) **entonces**
      1. MOVER_A (q_{cand});
      2. q_{act} ←q_{cand};
   7. **si no**
      1. MOVER_A ($q_{act}.padre$);
      2. q_{act} ← $q_{act}.padre$;
3. **Regresa T;**

Figura 3: Algoritmo SRT

Una vez generada q_{cand} de forma aleatoria en la región segura *S*, pasa por un proceso de validación efectuado por la función VALIDA. Como se muestra en la figura 4, q_{cand} (i) debe estar alejada de q_{act} a una distancia mayor a una distancia mínima prefijada d_{min} y (ii) no debe situarse en la región segura local de otra configuración previa en *T*. Si la validación tiene éxito, el robot se mueve a q_{cand} y el ciclo se repite. De lo contrario, el algoritmo genera otras configuraciones aleatorias desde q_{act} hasta encontrar una configuración valida o exceder un número máximo de intentos, I_{max}. En el último caso, el robot regresa al nodo padre de q_{act}, donde se ejecuta nuevamente el ciclo [30].

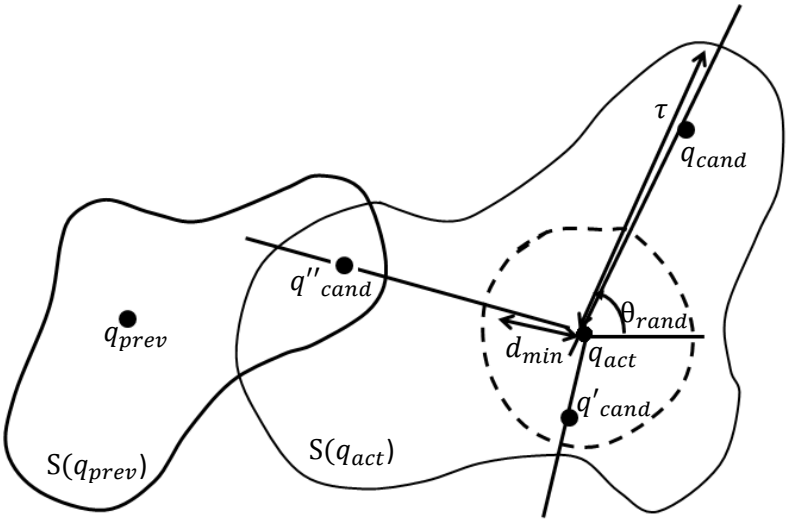


Figura 4: Generación de configuraciones candidatas con el método SRT. En este caso, q_{cand} es valida, mientas q' y q'' no lo son, la primera se encuentra a una distancia menor a d_{min} de q_{act} y q'' se ubica en la región segura local de otro nodo.

Típicamente, cuando el espacio libre ha sido explorado completamente, el algoritmo fallaría en encontrar una nueva dirección de exploración y el robot hará un proceso automático de retorno a la configuración inicial.

Una comparación del método SRT con el planificador RRT origina los siguientes comentarios:

- En comparación con el RRT, la estructura SRT es un árbol con aristas de longitud variable, dependiendo del radio de la *RSL* en dirección a θ_{rand}. Por lo tanto, durante la exploración, el robot recorrerá longitudes más

largas en regiones con pocos obstáculos y más pequeñas en virtud de objetos a su paso. También, no es necesario un chequeo de colisiones ya que las configuraciones candidatas son generadas dentro del área segura.

- Desde el punto de vista de la exploración, el método SRT es substancialmente en profundidad. Dado que el árbol se expande a partir de q_{act}, la posición actual del robot; en contraste con la expansión en anchura típica de los RRT puros, los cuales, no se emplean para exploración usando sensores. La introducción del mecanismo de retroceso es una consecuencia de la naturaleza del recorrido en profundidad del algoritmo SRT.
- El método STR mantiene algunas de las características más importantes del RRT, como ser conveniente para espacios de configuraciones de altas dimensiones y fácilmente modificable tanto para restricciones holonómicas como no holonómicas.

2.2. Exploración con SRT-Radial

Como se mencionó, la forma de la región segura local *S* refleja las características del sensor, así como la técnica de percepción adoptada. A su vez, la estrategia de exploración estará fuertemente afectada por la forma de *S*. En [12] se presenta una variante del método llamada SRT-Star, la cual involucra una estrategia de percepción que toma completamente la información reportada por el sistema de sensado, además de explotar la información proporcionada por los sensores en todas direcciones. En SRT-Star, *S* es una región con forma similar a una estrella debido a la unión de varios 'conos' con diferentes radios cada uno, ver figura 5. El radio del *i-ésimo* cono η_i es la distancia mínima entre la distancia del robot al obstáculo más cercano o el rango máximo medible con los sensores. Por lo tanto, para poder calcular *r*, la función RADIO primero debe identificar a que cono corresponde θ_{rand}.

Por el contrario, bajo la variante implementada en este proyecto la forma de *S*, idealmente, en ausencia de obstáculos, es circular por lo que es innecesaria la identificación del cono. A esta variante le denominamos SRT-Radial, en la cual una vez generada la dirección de exploración θ_{rand}, la función RADIO traza un rayo desde la ubicación actual hacia el borde de S, la porción comprendida dentro de *S* representa el radio en la dirección θ_{rand}, ver figura 6. Por lo tanto, en presencia de obstáculos la forma de *S* se deforma y para diferentes direcciones de exploración las longitudes de los radios varían [30].

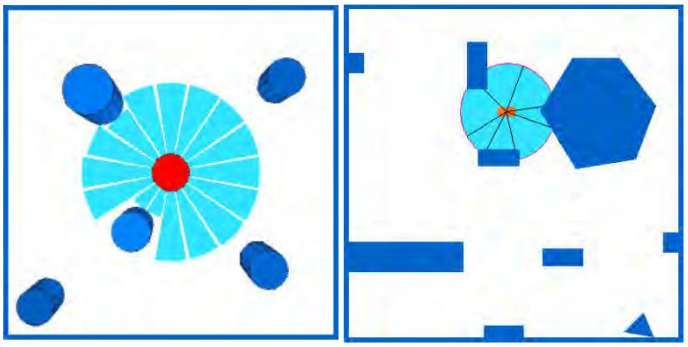


Figura 5 y 6 :(5) Región segura local S obtenida con la estrategia de percepción SRT-Star. Note que la extensión de S en algunos conos es reducida por el rango de alcance del sensor y (6) muestra diferentes radios obtenidos en la región segura local S con la estrategia de percepción SRT-Radial y la aplicación de la librería GPC.

2.3. Experimentos y resultados

Para ilustrar el comportamiento de exploración de la estrategia SRT-Radial se presentan los resultados del algoritmo desarrollado en C# (Microsoft Visual Studio 2010), aprovechando la estructura de la librería MSL ((Motion Strategy Libray) es una librería para el desarrollo y prueba de algoritmos de planificación de movimientos, incluye varios planificadores que usan RRT, PRM y FDP; así como un ambiente gráfico en OpenGL para la descripción de la escena.), así como su interfaz grafica para visualizar el ambiente de trabajo, el resultado de la exploración y el árbol generado pro el mismo. Para simular el modulo de percepción del sistema de sensado, se utilizo la librería GPC ((General Polygon Clipping) es una librería de operaciones para el manejo de recorte de polígonos.).

2.3.1. Detalles de la implementación

Las modificaciones hechas al método SRT radican principalmente en la condición de término del algoritmo y el tipo de robot móvil considerado. Para efectuar las simulaciones suponemos la disponibilidad de un sensor telemétrico rotacional a 360° situado sobre el robot, en general, se puede fácilmente extender el sistema para soportar cualquier número y tipo de sensores.

También es importante recordar que una consideración fundamental para el funcionamiento del método es la hipótesis de una perfecta localización.

2.3.2. Resultados obtenidos con SRT-radial, en ambientes 2D

2.3.2.1. Configuraciones (ambientes)

a) Sencillo

b) Medio

c) Complejo

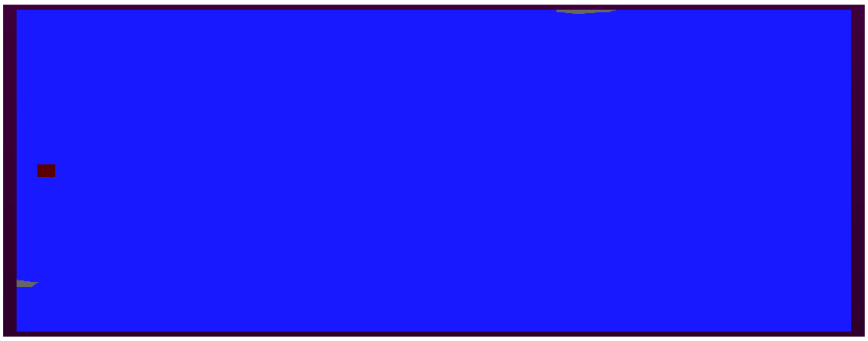
Figura 7: (a,b,c): Ambientes y posición inicial del robot para las pruebas del método SRT-Radial en 2D

2.3.2.2. Tiempos

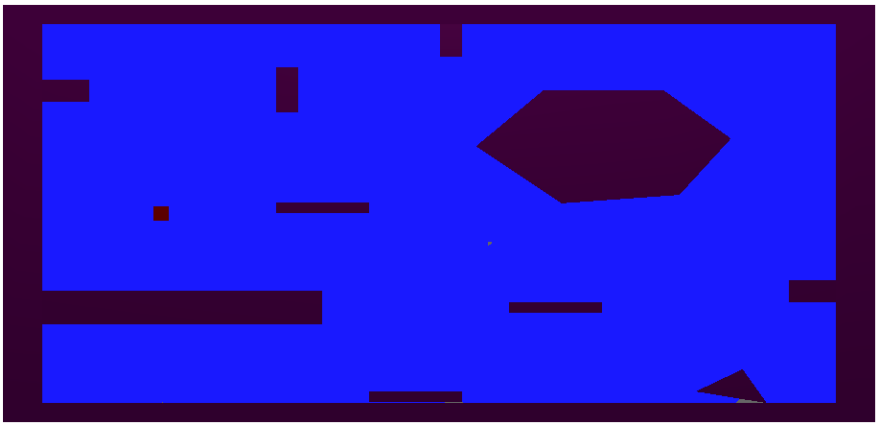
Tiempos (seg)	a)	b)	c)
1	4.10666666666667	3.9	1.37666666667
2	3.92666666666667	3.7966666666667	2.67833333333
3	3.9	3.4333333333333	2.67833333333
4	3.95166666666667	4.0316666666667	1.63666666667
5	3.64	3.3283333333333	2.41833333333
6	4.3666666666667	3.77	1.87166666667
7	3.875	3.875	1.56
8	3.745	4.0033333333333	1..6111111117
9	3.4316666666667	3.9783333333333	2.26166666667
10	3.355	3.7166666666667	2.93833333333
Media	3.82983333	3.78333333	1.942

2.3.2.3. Resultados (mapa)

a) Sencillo



b) Medio



c) Complejo

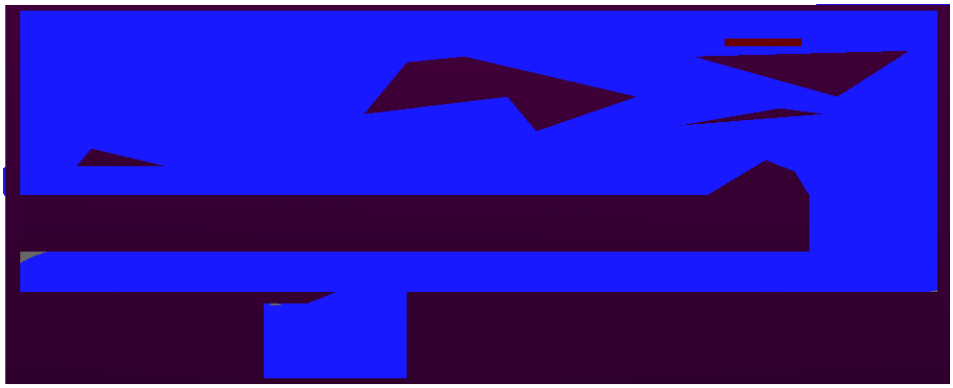
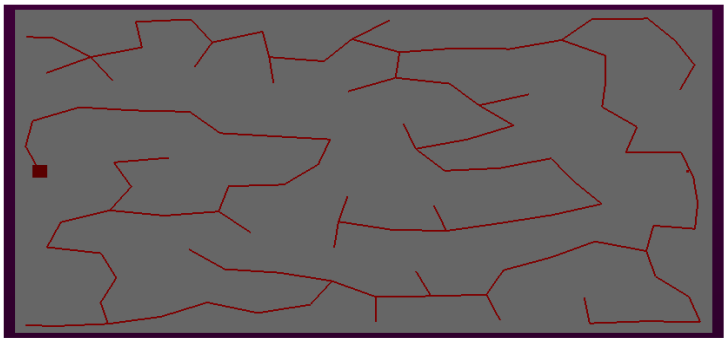


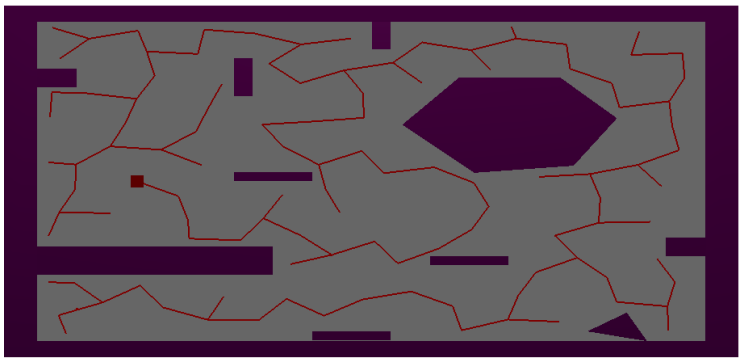
Figura 8: (a,b,c): Región segura (región azul) obtenida por el algoritmo SRT-Radial, en cada uno de los ambientes que exploro el robot.

2.3.2.4. Resultados (árbol)

a) Sencillo



b) Medio



c) Complejo

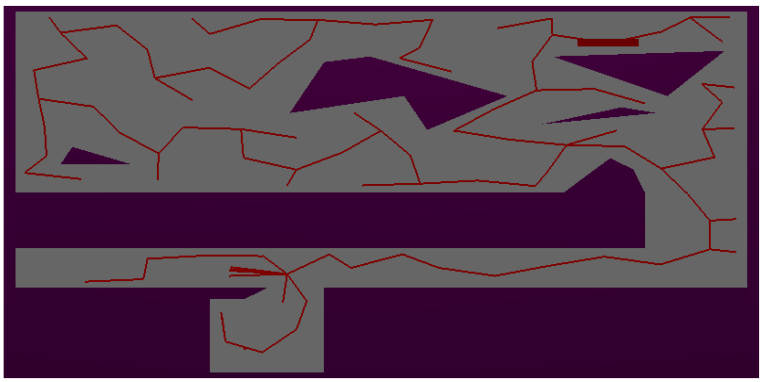


Figura 9: (a,b,c): Camino encontrado por el robot tipo carro en cada uno de los ambientes de prueba.

2.4. Conclusión

De acuerdo a los resultados experimentados en la plataforma, podemos deducir, la gran variedad de resultados que nos provoca el hecho de trabajar con un muestreo aleatorio, para la elección de un estado siguiente.

Muchas veces los tiempos mostrados, pueden disfrazar nuestro resultado, obteniendo algo inesperado, es decir, el hecho de encontrar tiempos muy cortos, no significa que el resultado sea el optimo y viceversa con los tiempos más largos.

Pero ahora planteamos una cuestión al aire; ¿Y si el problema a resolver no se encuentra en un mundo bidimensional, si no que en un mundo tridimensional?

La soluciones encontrada hasta el momento, nos limita a esta situación.

El ampliar la solución mostrada en 2D, para cualquier ambiente 3D, engrandece el campo de trabajo, y la capacidad de dar solución a un mayor número de problemas.

Capitulo 3

3. Algoritmos de exploración 3D

Se presenta un método para la exploración basada en sensores de ambientes desconocidos por un sistema robótico equipado con medidores de distancia. El método se basa en la generación incremental de una estructura de espacio de configuración de datos llamado árbol basado en el sensor de exploración (SET). La expansión de la SET es impulsado por la información del ambiente, donde el proceso de percepción tiene lugar. En particular, las fronteras de la región explorada se utilizan para guiar la búsqueda de configuraciones desde el punto de vista informativo [29].

Diversas estrategias de exploración se pueden obtener por una instancia del método general SET con diferentes técnicas de muestreo.

3.1. Representación de los robots manipuladores

La cinemática es la ciencia del movimiento que trata el tema sin considerar las fuerzas que lo ocasionan. Dentro de esta ciencia se estudian la posición, la velocidad, a aceleración y todas las demás derivadas de alto orden de las variables de posición (con respecto al tiempo o a cualquier otra variable). En consecuencia, el estudio de la cinemática de manipuladores se refiere a todas las propiedades geométricas y basadas en el tiempo del movimiento. Las relaciones entre estos movimientos y las fuerzas y momentos de torsión que los ocasionan constituyen el problema de la dinámica [24].

El estudio de la cinemática de manipuladores se relaciona, entre otras cosas, con la manera en que cambian las ubicaciones de los componentes de un mecanismo robótico, a medida que se articula dicho mecanismo. Se tocara un método en especial para

calcular la posición y la orientación del efector final del manipulador relativo a la base del mismo, como una función de las variables de las articulaciones[24].

3.1.1. Descripción de vínculos

Un manipulador puede considerarse como un conjunto de cuerpos conectados en una cadena mediante articulaciones. Estos cuerpos se llaman vínculos o segmentos. Las articulaciones forman una conexión entre un par adyacente de vínculos. El termino **par menor** se utiliza para describir la conexión entre un par de cuerpos, cuando el movimiento relativo se caracteriza por dos superficies que se deslizan una sobre otra [9].

Las consideraciones de diseño mecánico recomiendan que los manipuladores se construyan generalmente de articulaciones que exhiban solo un grado de libertad. La mayoría de los manipuladores tienen **articulaciones angulares** o articulaciones deslizantes llamadas **articulaciones prismáticas**. En el extraño caso de que un mecanismo este construido con una articulación que tenga *n* grados de libertad, puede modelarse como *n* articulaciones de un grado de libertad, conectadas con n-1 vínculos de longitud cero. Por lo tanto y sin perder la generalidad, consideramos solo manipuladores que tengan articulaciones con un solo grado de libertad.

Un **vinculo** se considera solamente como un cuerpo rígido que define la relación entre dos ejes de articulaciones adyacentes de un manipulador. Los vínculos se enumeran empezando desde la base inmóvil del brazo, a la cual podríamos llamar vinculo 0. El primer cuerpo móvil es el vinculo 1 y así, sucesivamente, hasta llegar al extremo libre del brazo, el cual es el vinculo *n*. Para poder posicionar en efector final en espacio 3D de forma general se requiere un mínimo de seis articulaciones. Los manipuladores comunes tienen cinco o seis articulaciones. Algunos robots no son realmente tan simples como una cadena cinemática abierta; tienen una configuración de paralelogramo u otras estructuras cinemáticas cerradas[9].

Los ejes de articulación se definen mediante líneas en el espacio. El eje de articulación *i* se define mediante una línea en el espacio, o la dirección de un vector, sobre el cual gira el vincula *i* con respecto al vinculo *i-1*. Resulta que para todos los fines cinemáticas, un vinculo puede especificarse con dos números (**longitud** y **torsión** del vinculo), los cuales definen la ubicación relativa de los ejes en el espacio.

Los vínculos adyacentes tienen un eje de articulaciones común entre ellos. Uno de los parámetros de interconexión tienen que ver con la distancia a lo largo de este eje común, desde un vinculo hasta el siguiente. Este parámetro se llama **desplazamiento de vinculo**. El desplazamiento en el eje de la articulación *i* se llama d_i. El segundo parámetro describe la cantidad de rotación sobre este eje común entre vínculos y su vinculo adyacente. Este parámetro se llama **ángulo de rotación**, θ_i.

Cualquier robot puede describirse en forma cinemática proporcionando los valores de cuatro cantidades para cada vinculo. Dos describen el vinculo en si, y los otros dos describen la conexión del vinculo con un vinculo adyacente. En el caso de una articulación angular, θ_i, se llama **variable de articulación** y las otras tres cantidades son **parámetros de vinculo** fijos. Para las articulaciones prismáticas, d_i es la variable de articulación y las otras tres cantidades son parámetros de vinculo fijos. La definición de mecanismos por medio de estas cantidades es una convención que generalmente se le conoce como notación **Denavit-Hartenberg**.

3.1.2. Representación DH

En 1955 Denavit y Hartenberg propusieron un método matricial que permite establecer de manera sistemática un sistema de coordenadas [25].

La representación de Denavit-Hartenberg (D-H) establece que seleccionándose adecuadamente los sistemas de coordenadas asociados a cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón [25].

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

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

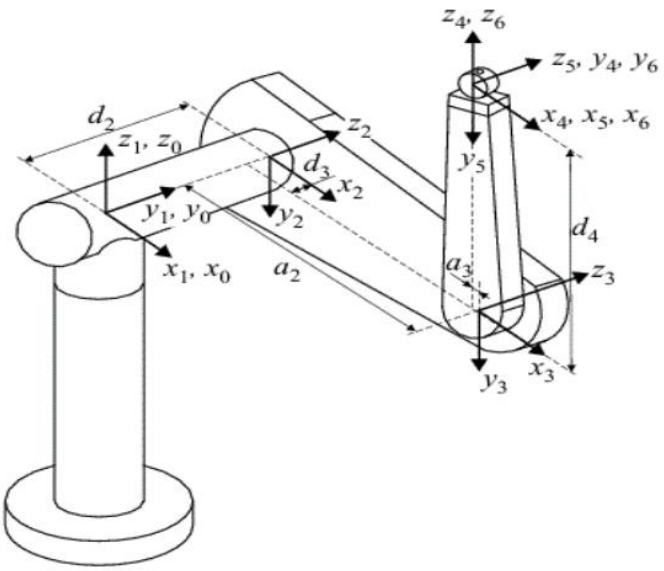


Figura 10: Sistemas de coordenadas de un brazo manipulador.

3.1.2.1. Algoritmo de Denavit-Hartenberg

Para que la matriz A_{i-1}^{i} relacione los sistemas coordenados es necesario que los sistemas coordenados se determinen mediante los siguientes pasos:

1. Numerar y etiquetar el eslabón fijo (base) como 0.

2. Numerar y etiquetar los eslabones móviles desde 1 hasta el n eslabón móvil.

3. Localizar y numerar el eje de cada articulación y etiquetarla comenzando desde z_0 hasta z_{n-1}. Si la articulación es rotativa, el eje será su propio eje de giro. Si la articulación es prismática, el eje será a lo largo del cual se produce el desplazamiento.

Establecimiento del sistema coordenado de la base.

4. Establecer el sistema coordenado de la base estableciendo el origen O_0 como cualquier punto del eje z_0. Arbitrariamente establecer los ejes X_0 y Y_0 respetando la regla de la mano derecha.

Establecimiento de los sistemas coordenados de las demás articulaciones.

5. Localizar el origen O_i:
    a. En la intersección del eje z_i con la línea normal común a la intersección de z_i y z_{i-1}.

    b. En la intersección de z_i y z_{i-1}, si es que z_i y z_{i-1} se interceptan.

    c. En la articulación i, si z_i y z_{i-1} son paralelos.

6. Establecer X_i
    a. A lo largo de la línea normal común entre los ejes z_i y z_{i-1} que pasa por O_i.

    b. En la dirección normal al plano formado por z_i y z_{i-1}, si es que estos dos ejes se intersectan.

7. Establecer Y_i de acuerdo a la regla de la mano derecha.

8. Establecimiento del sistema coordenado de la herramienta (extremo)
    a. Localizar el sistema coordenado *n-ésimo* en el extremo del robot. Si es una articulación rotacional, establecer z_n a lo largo de la dirección z_{n-1} y establecer el origen O_n de la manera que más convenga a lo largo de z_n preferentemente en el centro de la pinza o la punta de cualquier herramienta que el robot tenga montada.
    b. Establecer x_n y y_n de acuerdo a la regla de la mano derecha. Si la herramienta es una pinza, es común establecer el eje y_n entre los "dedos" de la pinza y x_n será orto-normal a z_n y y_n.

9. Crear una tabla con los parámetros D-H de los eslabones:

Eslabón i	θ_i	d_i	a_i	α_i

Donde:

$\boldsymbol{\theta_i}$ es el ángulo formado por los ejes X_{i-1} y X_i medido en un plano perpendicular a Z_{i-1} utilizando la regla de la mano derecha. Este es un parámetro variable en articulaciones rotatorias.

$\boldsymbol{d_i}$ es la distancia a lo largo del eje Z_{i-1} desde el origen O_{i-1} hasta la intersección del X_i con el eje Z_{i-1}. Este es un parámetro variable en articulaciones prismáticas.

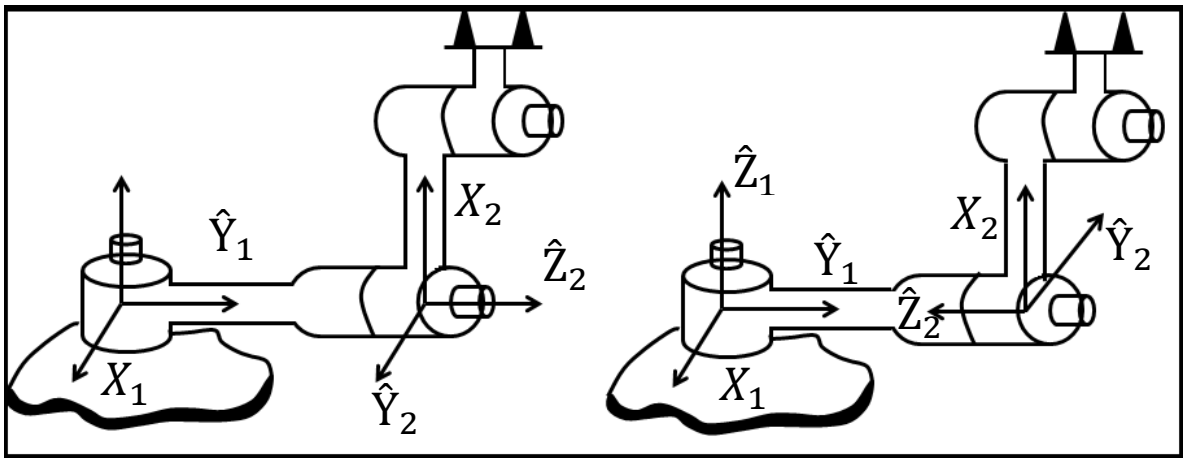
$\boldsymbol{a_i}$ para articulaciones rotatorias, es la distancia a lo largo del eje X_i desde el origen O_i hasta la intersección del eje X_i con el eje Z_{i-1}.

Para articulaciones prismáticas, es la distancia mas corta entre los ejes.

$\boldsymbol{\alpha_i}$ es el ángulo formado por los ejes Z_i y Z_{i-1} medido en un plano perpendicular al eje X_i utilizando la regla de la mano derecha.

Cada una de estas cuatro operaciones se puede expresar mediante una matriz rotación-traslación homogénea básica y el producto de estas cuatro matrices de transformación homogéneas básicas da una matriz de transformación homogénea compuesta.

10. Realizar la matriz D-H de transformación homogénea A_{i-1}^{i} para cada eslabón de acuerdo a los datos de la tabla anterior [26].



$a_1 = 0$	$a_2 = L_2$		$a_1 = 0$	$a_2 = L_2$	
$\alpha_1 = -90°$	$\alpha_2 = 0$	θ_2=-90°	$\alpha_1 = 90°$	$\alpha_2 = 0$	θ_2=90°
$d_1 = 0$	$d_2 = L_1$		$d_1 = 0$	$d_2 = -L_1$	

Figura 11: Ejemplos de valores D-H para manipuladores

Representación matricial:

$$A_i^{i-1} = T(0,0,d_i)T(z,\theta_i)T(x,\alpha_i)T(a_i,0,0)$$

Desarrollando la expresión:

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&d_i\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos\theta_i&-\sin\theta_i&0&0\\\sin\theta_i&\cos\theta_i&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&a_i\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&\cos\alpha_i&-\sin\alpha_i&0\\0&\sin\alpha_i&\cos\alpha_i&0\\0&0&0&1\end{bmatrix}$$

Y así obtenemos la expresión general de D-H, para θ_i, d_i, a_i y α_i.

$$A_i^{i-1} = \begin{bmatrix}\cos\theta_i&-\cos\alpha_i\sin\theta_i&\sin\alpha_i\sin\theta_i&a_i\cos\theta_i\\\sin\theta_i&\cos\alpha_i\cos\theta_i&-\sin\alpha_i\cos\theta_i&a_i\sin\theta_i\\0&\sin\alpha_i&\cos\alpha_i&d_i\\0&0&0&1\end{bmatrix}$$

Una vez obtenida la expresión general D-H, podemos encontrar la inversa de esta transformación tal como:

$$\left[A_i^{i-1}\right]^{-1} = A_{i-1}^{i} = \begin{bmatrix}\cos\theta_i&\sin\theta_i&0&-a_i\\-\cos\alpha_i\sin\theta_i&\cos\alpha_i\cos\theta_i&\sin\alpha_i&-d_i\sin\alpha_i\\\sin\alpha_i\sin\theta_i&-\sin\alpha_i\cos\theta_i&\cos\alpha_i&-d_i\cos\alpha_i\\0&0&0&1\end{bmatrix}$$

Donde d_i, a_i y α_i son constantes, mientras que θ_i es la variable de la articulación para un manipulador de tipo revolución. Para una articulación prismática, la variable articulación es d_i, mientras que θ_i, a_i y α_i son constantes [26].

Para una articulación prismática, la variable es d_i, mientras que θ_i, a_i y α_i son constantes. En este caso:

$$A_i^{i-1} = T(z, \theta_i)T(0,0, d_i)T(x, \alpha_i)$$

$$= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & 0 \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y su inversa es:

$$\left[A_i^{i-1}\right]^{-1} = A_{i-1}^i = \begin{bmatrix} \cos\theta_i & \sin\theta_i & 0 & 0 \\ -\cos\alpha_i \sin\theta_i & \cos\alpha_i \cos\theta_i & \sin\alpha_i & -d_i \sin\alpha_i \\ \sin\alpha_i \sin\theta_i & -\sin\alpha_i \cos\theta_i & \cos\alpha_i & -d_i \cos\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Utilizando la matriz $\left[A_i^{i-1}\right]$ se puede relacionar un punto p_i en reposo en el elemento *i* y expresado en coordenadas homogéneas con respecto al sistema de coordenadas *i* en el sistema de coordenadas *i-1* establecido en el elemento *i-1* por:

$$p_i = A_i^{i-1} p_i$$

donde $p_{i-1} = [X_{i-1}, Y_{i-1}, Z_{i-1}, 1]^T$ y $p_i = [X_i, Y_i, Z_i, 1]^T$.

3.2. Estructuras de datos geométricas

El problema de la exploración de un mundo desconocido con un sistema robótico multi-cuerpo, como un manipulador fijo o móvil, es más difícil. Esto se debe fundamentalmente al hecho de que el espacio de detección (el mundo) y la planificación del espacio (el espacio de configuración) son de naturaleza muy diferente. En particular, el primero es un espacio euclidiano de dimensión 2 o 3, mientras que el segundo es una variedad en la presencia de las coordenadas angulares y tiene una dimensión igual al

número de grados de libertad de que el robot, por lo general de 6 o más. Aunque las fronteras del ambiente claramente conservan su valor informativo, utilizando esta información para planear acciones en el espacio de configuración no es sencilla. En la literatura existen pocos trabajos que abordan este problema principalmente para los manipuladores de base fija, por ejemplo, ver [26].

De hecho, para trabajar sobre un ambiente 3D, nos implica un mayor número de grados de libertad para el robot así como el uso de estructuras geométricas, distintas a las usadas en un ambiente 2D, tales como:

- Octrees
- Voxeles

3.2.1. Voxel y octree

El **vóxel** (del inglés volumetric pixel) es la unidad cúbica que compone un objeto tridimensional. Constituye la unidad mínima procesable de una matriz tridimensional y es, por tanto, el equivalente del píxel en un objeto 3D [27].

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

Las imágenes con vóxeles se usan generalmente en el campo de la medicina y se aplican, por ejemplo, en la tomografía axial computarizada o para las resonancias magnéticas. De este modo, los profesionales pueden obtener un modelo preciso en tres dimensiones del cuerpo humano.

Actualmente, su uso ya se ha extendido en multitud de campos como la medicina, ingeniería, cine, videojuegos...

Al igual que los píxels, los vóxels no contienen su posición (x,y,z) en el espacio 3D, sino que esta se deduce por la posición del vóxel dentro del archivo de datos.

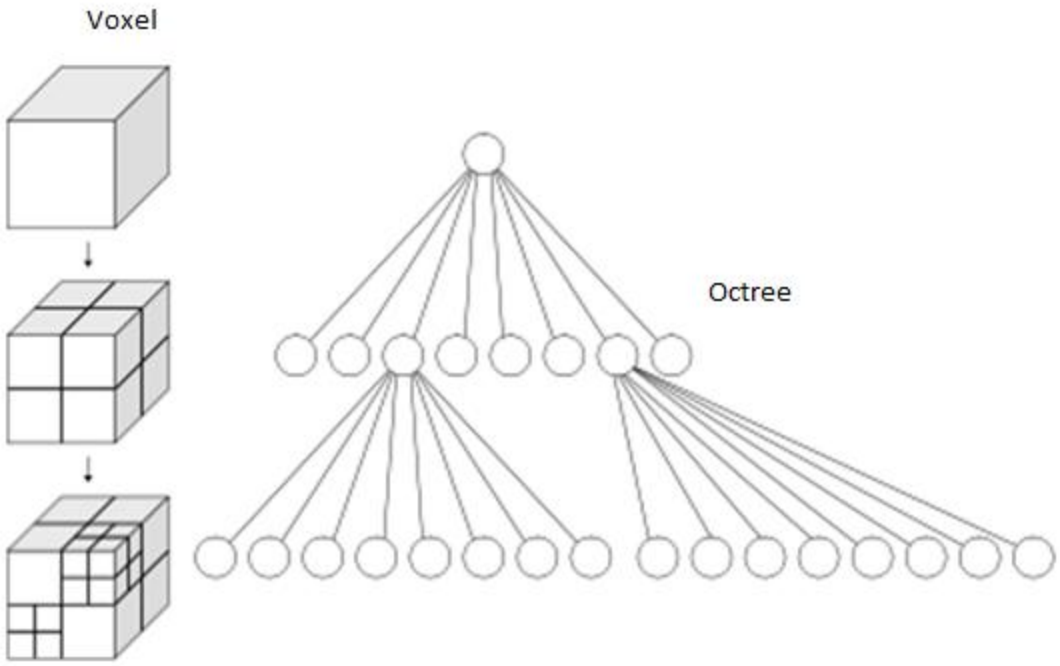


Figura 12: Voxel y octree.

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

En una estructura octree, cada nodo subdivide el espacio que representa en ocho octantes. En una región punto (PR) octree, el nodo almacena un punto tridimensional explicito, el cual es el "centro" de la subdivisión para ese nodo; el punto que define una de las esquinas para cada uno de los ocho hijos. En una octree MX, el punto de subdivisión es implícitamente el centro del espacio que el nodo representa. El nodo raíz de una PR octree puede representar un espace infinito; el nodo raíz de una octree MX debe representar un espacio con limite finito para que

los centros implícitos estén bien definidos. Las estructuras octree nunca se consideran árbol kd, ya que los árboles kd dividen en una dimensión mientras que las estructuras octree dividen alrededor de un punto. Los árboles kd además son siempre binarios, lo cual no se cumple para las estructuras octree.

3.3. Geometría solida reconstructiva

Geometría sólida constructiva, se define como la representación de un objeto mediante un árbol binario, donde los nodos no terminales representan operaciones booleanas (unión, diferencia, intersección y complemento) y los nodos hojas representan sólidos primitivos.

Cuando se representa un objeto mediante árboles CSG se utilizan una serie de primitivas que al combinarse por operadores booleanos generan el objeto deseado [23].

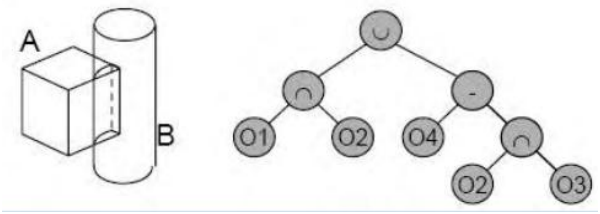


Figura 13

Es una técnica de modelado que consiste en la combinación de objetos y operaciones producen objetos más complejos.

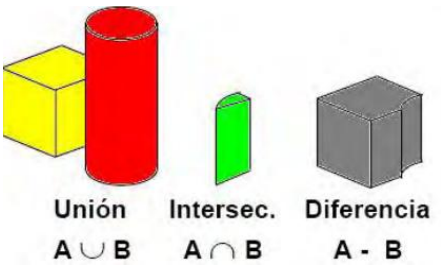


Figura 14

Las operaciones básicas son: Unión, diferencia, intersección.

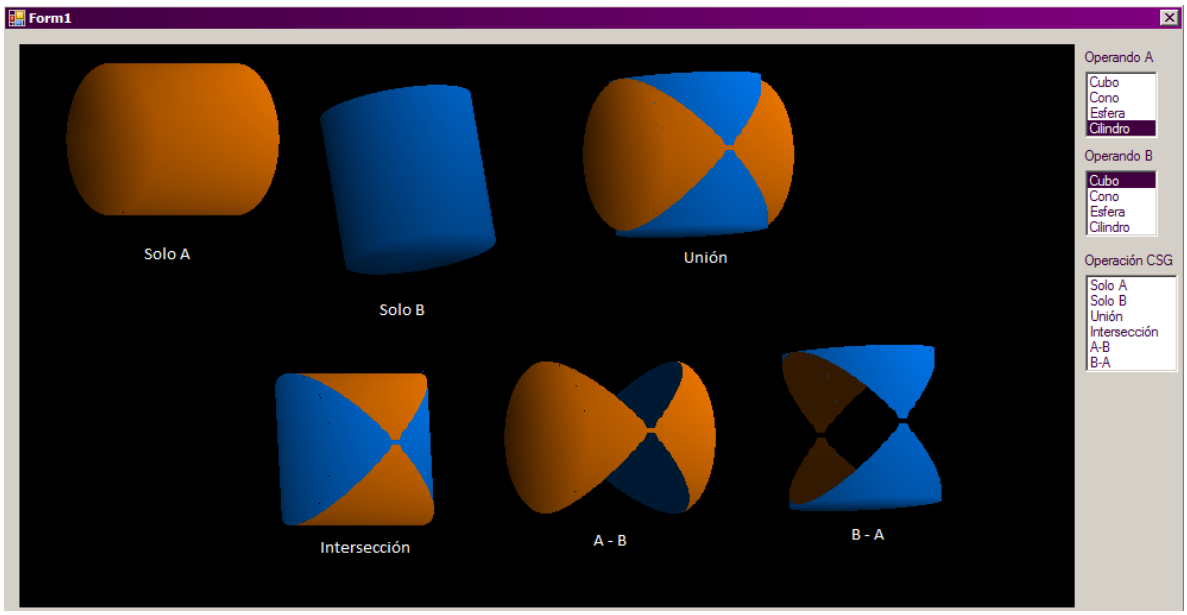


Figura 15: Ejemplo de cada una de las operaciones realizadas con geometría solida reconstructiva entes el cilindro A (naranja) y cilindro B (Azul)

3.4. Voxel Coloring

Voxel Coloring es una técnica utilizada para crear una reconstrucción de la escena 3D a partir de un conjunto bastante disperso de imágenes. Tal vez el aspecto más útil de esta técnica es el hecho de que puede crear una reconstrucción de la escena fotográfica sin correspondencia de imagen computacional. En este artículo se describen mis esfuerzos para implementar y probar un algoritmo 3D voxel coloring [22].

La técnica de Seitz y Dyer para la reconstrucción de la escena usando voxel coloring [22] trabaja por medio de la descomposición de la escena para ser reconstruida en pequeños elementos de volumen, llamados voxeles.

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

Si la diferencia entre los píxeles es baja, el voxel es considerado el color invariable, y en consecuencia se colorea. De lo contrario, la voxel permanece sin color. Después de un barrido del volumen de toda la escena, todos los voxeles correspondientes a los puntos de la escena deben ser de color, mientras que los que no corresponden a los puntos de la escena quedan sin color [27].

Toda la escena se puede reconstruir en un solo paso, haciendo uso de la restricción de visibilidad ordinaria. En concreto, por la descomposición de la escena en capas voxel e iniciar el barrido con la capa más cercana al volumen de la cámara, todas las oclusiones (oclusión: se usa para describir la manera como un objeto cercano al viewport mask se cierra) pueden ser explicadas por la retención para cada imagen en la imagen de conjunto de una máscara binaria, en un principio los ceros, con motivo de que los píxeles ya ha utilizado para colorear un voxel. Por ejemplo, cuando se realiza voxel coloring para una botella de salsa Tabasco en posición vertical, el algoritmo comienza en la tapa y se procede a la base. Píxeles de la imagen utilizada para la tapa de color voxels están marcados y no se consideran en la coloración de la base, incluso si un voxel en la base pasa a proyectar en algunos de ellos [21].

Cada voxel visitado es proyectado sobre el plano de cada imagen en el conjunto imagen usando parámetros de la cámara calculados durante la calibración. En general, un voxel, que es un cubo, proyecta una especie de hexágono en el plano de la imagen. Para simplificar el algoritmo, elegí en lugar de proyectar las esquinas del voxel en el plano de la imagen y luego usar el mínimo y máximo “x” y “y” y re-proyectar las coordenadas para crear una huella rectangular [22].

Una vez que la huella de un voxel es determinada, se encuentran los píxeles sin marcar en ella, y sus valores medios de rojo, verde y azul se calculan. Si el valor RGB calculado está dentro de un cierto umbral suministrado por el usuario del color de fondo, el voxel que se examina es considerado como un voxel de fondo y no de color. De lo contrario, la suma de los cuadrados de las diferencias se calcula entre los valores de los píxeles sin marcar y el valor medio para cada canal y se añade a un total acumulado que se mantiene para cada voxel sobre el conjunto de imágenes de entrada. Si, después de que su presencia se ha calculado para cada una de las imágenes de entrada, un voxel está decidido a no ser un voxel de fondo, estos totales se utilizan para calcular la desviación estándar de cada canal, y las desviaciones estándar para los tres canales se promedian y se compara con un umbral del usuario. Si la desviación estándar media es menor que el valor umbral, el voxel es considerado consistente y se colorea con el color medio de sus huellas, y todos los píxeles dentro de las huellas del voxel se marcan [22].

3.4.1. Algoritmo Voxel Coloring

S={ } // Inicializamos el conjunto del voxel coloring en ”vacío”

Para i=0 hasta r hacer // Atraviesa cada capa r

Para cada V en la i-esima capa de voxeles hacer

Si existe la suficiente relación de colores de los pixeles

Entonces agregar V a S

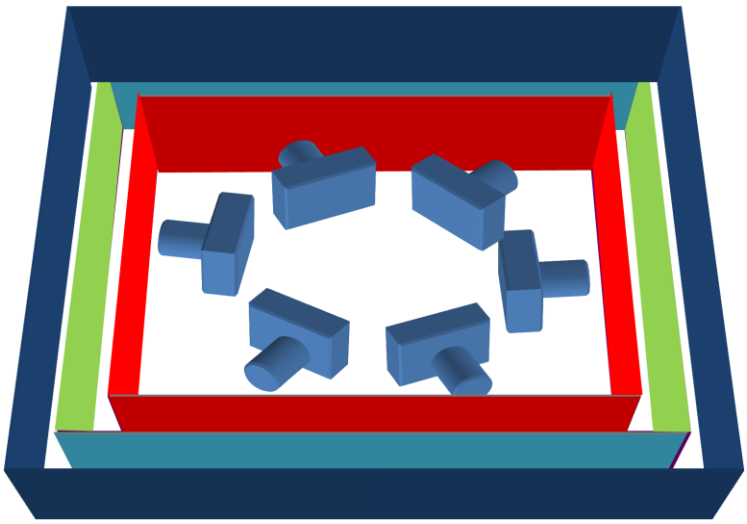


Figura 16: Representación del manejo de las vistas y las capas dentro el algoritmo Voxel coloring

3.4.2. Ejemplo del Voxel Coloring:

Aquí presentamos el conjunto de imágenes a utilizar para reconstruir la escena 3D:

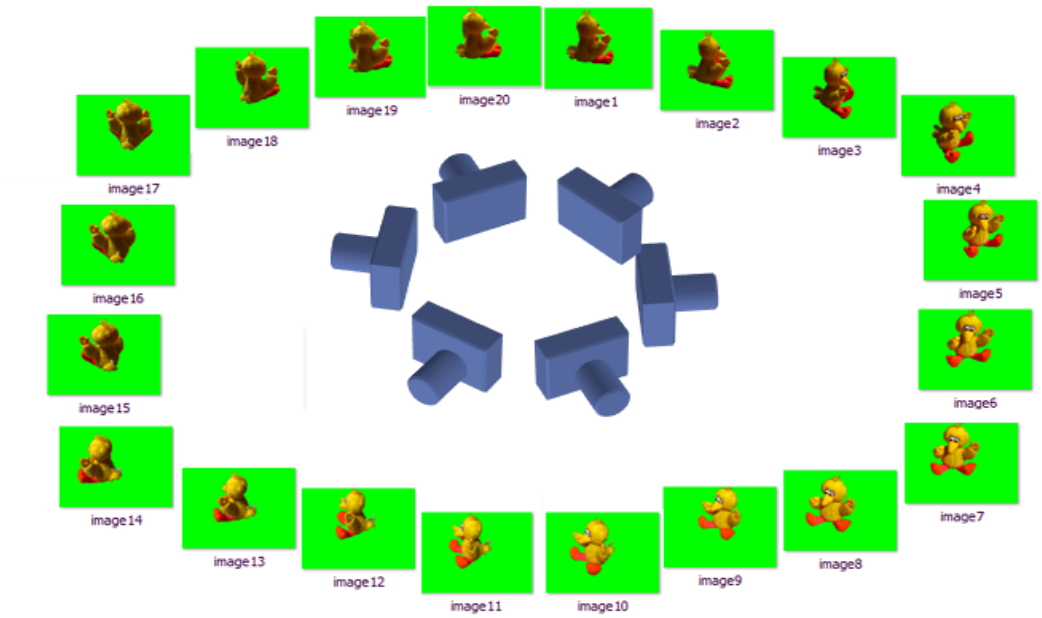


Figura 17: Conjunto de imágenes ocupadas para la construcción de la escena a base de voxeles.

Y a continuación presentamos resultados arrojados por las pruebas realizadas, dependientes del tamaño del voxel y del tamaño de la imagen:

Tamaño del voxel	Tamaño imagen	Tiempo
1(A)	**165X165X175**	**18266.493334 sec.**
3(B)	**165X165X174**	**789.716880 sec.**
5(C)	**165X165X175**	**204.631825 sec.**
10(D)	**170X170X180**	**41.186707 sec.**

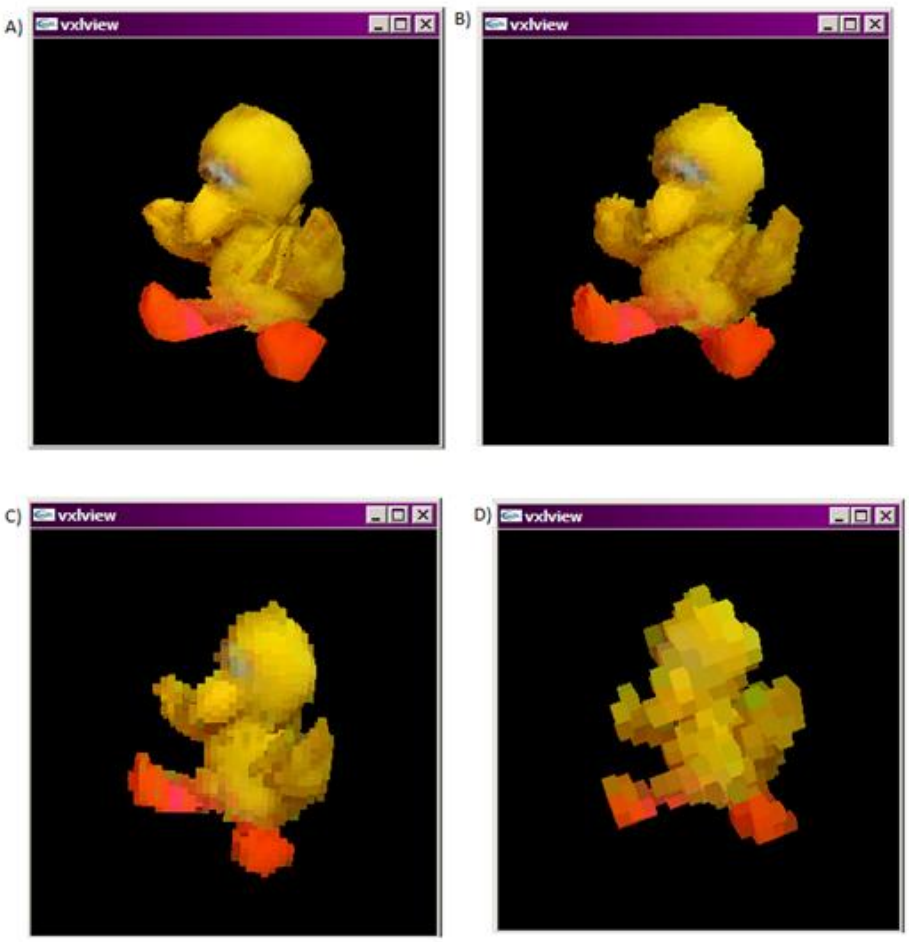


Figura 18: Imágenes reconstruidas a partir de una colección de imágenes, la calidad de la reconstrucción siempre dependerá del tamaño de la imagen y del tamaño del voxel.

3.5. Algoritmo SRT adaptado a ambientes 3D (SET)

En aplicaciones de robótica de servicio, se requieren de sistemas robóticos para llevar a cabo determinadas tareas en entornos que son parcial o completamente desconocidos. La detección, planificación y ejecución de movimiento debe estar debidamente entrelazado con el fin de descubrir y navegar por las porciones del medio ambiente que son relevantes para la tarea asignada. Por ejemplo, el sensor de la planificación basada en el movimiento aborda el problema de encontrar un camino libre de colisiones a partir de una configuración inicial hacia una configuración meta en un entorno desconocido [1], [2].

3.5.1. Introducción

Un escenario similar pero diferente, basado en sensores de exploración, en donde el problema es "cubrir" el medio ambiente tanto como sea posible con las percepciones sensoriales, es un problema clásico. A menudo, esto está destinado a la construcción de un mapa del entorno, que puede utilizarse para planificar y ejecutar nuevas acciones; sin embargo, esto puede no ser necesario, por ejemplo, si el objetivo es simplemente buscar un objeto perdido [4].

Existen una cantidad constante de propuestas en la literatura acerca de la exploración basada en sensores utilizando un solo robo móvil. Por lo general, se asume que el robot tiene forma de disco y está equipado con un telémetro láser omnidireccional. Para este problema, existen muchos algoritmos de exploración que caen en la clase de estrategias basadas en frontera [3] - [7]. Estas se basan en la idea de que el robot debe acercarse a la frontera entre las áreas exploradas e inexploradas de los ambientes con el fin de maximizar la utilidad esperada de los movimientos del robot.

En este trabajo se presenta el método SET (Sensor-based Exploration Tree), una estrategia para la exploración basada en fronteras para los sistemas robóticos

generales que puede verse como una extensión del método SRT para robots móviles que se describe en [7], [12]. La idea básica es guiar al robot para que el sistema sensorial pueda realizar una exploración a profundidad del mundo, sensando (escaneando) progresivamente regiones que son contiguas desde el punto de vista de la localización del sensor. La información recopilada acerca del espacio libre se mapea al roadmap en el espacio de configuraciones, el cual se construye a través de un procedimiento de muestreo. Este punto de vista se utiliza para seleccionar una configuración nueva, que se añade al SET. En el proceso de exploración, los robots alternan movimientos hacia delante y hacia tras en el SET, lo que podría verse como como un hilo de Ariadna (recordemos que es un método clásico de planificación de movimientos propuesto en los años 90, por J.M. Ahuatzin)[1].

Las dos estrategias principales de exploración pueden obtenerse instanciando el método general propuesto, con diferentes técnicas de muestreo. En la primera, el roadmap se expande utilizando un enfoque de método de muestreo global. En el segundo, el roadmap se construye en la forma de un bosque de árboles conectados, cada uno enraizado en una configuración con un punto de vista diferente y estos crecen a través de un procedimiento "local" de muestreo. Las dos estrategias propuestas se comparan mediante simulaciones con varios robots en diferentes escenarios [17].

3.5.2. Definición general del problema

El robot “se encuentra” en un mundo desconocido que contiene obstáculos. El cual se debe explorar para construir un modelo del mismo. La exploración se realiza intentando "cubrir" el mundo tanto como sea posible con los sistemas de percepción. Comenzamos con el planteamiento del problema para los sistemas robóticos generales. Este enfoque se puede particularizar a un caso específico al describir la estrategia de exploración propuesta.

3.5.2.1. Robot y modelos del mundo

El robot es llamado *R*. Se compone de una cadena cinemática de *r* cuerpos rígidos ($r \geq 1$) interconectado por uniones elementales. Esta descripción incluye: manipuladores de base fija, robots móviles y robots móviles multi-cuerpos, como vehículos con remolques, robots con forma de serpiente, humanoides y manipuladores móviles [18].

El *mundo W* es un subconjunto compacto de R^N, con $N = 2$ o 3. Este representa el espacio físico en el que el robot se mueve y percibe. *W* contiene obstáculos estáticos O_j, $j = 1, 2, \ldots, p$, cada uno de ellos subconjunto compacto conectado de *W*. El limite ∂W se supone que actúa como una "valla" y por lo tanto se considera como un obstáculo. La *región del obstáculo O* es la unión de ∂W y todos los obstáculos O_j, $j = 1, 2, \ldots, p$. El *mundo libre* es $W_{free} = W \setminus O$.

El *espacio de configuración* del robot se denota por *C* y una configuración de robot por *q*. Sea *R* (*q*) la región compacta de *W* ocupado por el robot en *q*. Se define la *región C-obstáculo* como *CO*, como el conjunto de configuraciones *q* tal que *R* (*q*) *F(q)* ∩ *O* ≠ El *espacio libre configuración es* $C_{free} = C \setminus CO$.

3.5.2.2. Modelo del sensor

El robot está equipado con un sistema de m sensores, cuya operación se formaliza de la siguiente manera. Se asume que el robot está en *q*, denotado $F_i(q) \subset R^N$ la región compacta ocupada por el *i-ésimo* campo de visión del sensor , que se supone que es en forma de estrella con respecto al sensor central $s_i(q) \in W$. Por ejemplo, en R^2, $F_i(q)$ puede ser un sector circular con el ápice $s_i(q)$, ángulo de apertura α_i y radio R_i, en donde este último es el rango de percepción (ver figura 19, izquierda).

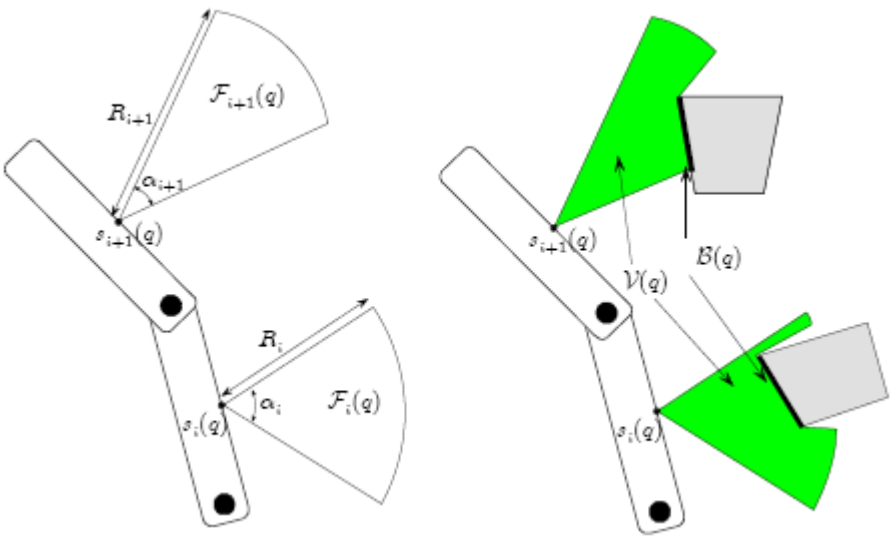


Figura 19: Izquierda: centros del sensor $s_i(q)$ y $s_{i+1}(q)$ de los campos de vista asociados $F_i(q)$ y $F_{i+1}(q)$ cuando el robot esta en la configuración q. Derecha: la vista $V(q)$ y los limites de los obstáculos visible $B(q)$.

El (total) *campo del sensor* en *q* es *F(q)*, definido como la unión de todos los $F_i(q)$ para *i* = 1, 2, ..., *m*.

Dada una configuración *q del robot*, un punto *p*ϵ*W* se dice que es *visible desde el sensor i-ésimo* si $p \epsilon F_i(q)$ y el segmento de línea abierta que une *p* y $s_i(q)$ no se interceptan *∂O∪∂R(q)*.

En cada configuración *q*, el sistema sensorial del robot regresa (ver figura 19, derecha):

- La *región visible libre* (o vista) *V(q)*, que considera todos los puntos de W_{free} que son visibles para por lo menos un sensor.
- El *límite del obstáculo visible B (q) = ∂O∪∂V(q)*, que considera todos los puntos de *∂O* que son visibles para por lo menos un sensor.

El sensor antes mencionado es una idealización de un telemetro "continuo". En la práctica, por ejemplo un sensor puede ser realizado por un telémetro láser de rotación, que devuelve la distancia al punto más cercano de un obstáculo en todas las direcciones (rayos) contenidos en su campo de visión

(con una cierta resolución). Otro de los sistemas sensoriales que satisface la descripción anterior es una cámara estereoscópica.

3.5.2.3. Tarea de exploración

El robot explora el mundo a través de una secuencia de acciones vista-plan-movimiento. Cada configuración donde se adquiere una visión es llamada *vista de configuración*. Sea q^0 la configuración inicial del robot y $q^1, q^2, \dots, q^k$ la secuencia de las configuraciones de vista asumidas por el robot hasta la *k-ésima* etapa de exploración. Cuando la exploración se inicia, todo el conocimiento endógeno inicial del robot se puede expresar como:

$$\varepsilon^0 = R(q^0) \cup V(q^0), \qquad \dots\dots\dots (1)$$

Donde $R(q^0)$ representa el volumen libre que ocupa el cuerpo del robot (calculado sobre la base de sensores propioceptivos) y $V(q^0)$ es la vista en q^0 (proporcionados por los sensores exteroceptivos. En el paso $k \geq 1$, la *región que es explorada* es:

$$\varepsilon^k = V(q^k) \cup \varepsilon^{k-1}.$$

Un requisito obvio es que $R(q^k) \subset \varepsilon^{k-1}$ para cada *k*, tenemos:

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

Un punto $p \epsilon W_{free}$ se define *explorado en el paso k* ,si está contenido en ε^k y se define inexplorado de otra manera. En cada paso *k*, $\varepsilon^k \epsilon W_{free}$ es la estimación actual del mundo libre.

Una configuración q es segura en el paso k si $R(q) \subset \varepsilon^k$. La *región segura* S^k recoge todas las configuraciones que son seguras en el paso k. Consideremos que $S^k \subseteq C_{free}$ representa una imagen de la configuración del espacio de ε^k y es la estimación actual de C_{free}. Un camino en C es seguro en el paso k si está completamente contenida en S^k. El objetivo de la exploración es ampliar ε^k tanto como sea posible a medida que aumenta k.

3.5.2.4. Estrategias de exploración

Supongamos que el robot es capaz de asociar una ganancia de información *I(q, k)* para cualquier q (seguro) en el paso k. Esto es una estimación de la información del mundo que puede ser descubierta mediante la adquisición de una vista de q en el paso actual.

Método SET

1. Actualiza SET y el modelo del ambiente.
2. Se extrae la frontera local libre LFB(q^k, k)
3. **SI** la frontera local libre es no vacía
   1. ($q^{k+1}, U(q^{k+1})$)←Busca configuración con utilidad máxima admisible en el interior de conjunto actual D(q^k, k)
4. **Si no**
   1. $U(q^{k+1})$←0
5. **Si** $U(q^{k+1}) \geq U_{min}$
   1. Planifica una camino seguro de q^k a q^{k+1}
   2. Se mueve a q^{k+1} y obtiene la vista del sensor
6. **Si no**
   1. Se mueve a la configuración padre.

Figura 20: Una descripción en pseudocódigo de la iteración *k-ésima* del método SET en el que el robot comienza en q^k.

Consideremos el paso de exploración *k-ésimo*, el cual inicia con el robot en q^k. Sea $Q^k \subset S^k$ la *región informativa segura*, es decir, el conjunto de configuraciones que tienen ganancia de información "No cero" y que se pueden alcanzarse desde q^k a través de un camino seguro en el paso k. En una estrategia de exploración general, la siguiente configuración de vista, q^{k+1} se elige en $Q^k \cap D\ (q^k, k)$, de acuerdo a

algún criterio de selección (por ejemplo, la maximización de la ganancia de información). El conjunto $D\ (q^k, k) \subseteq C$ denota un conjunto admisible alrededor de q^k en el paso *k*: su forma y tamaño determina la "localidad" de la búsqueda. Por ejemplo, si $D\ (q^k, k) = C$, una búsqueda global se lleva a cabo, mientras que si $D\ (q^k, k)$ está un pequeño "vecindario" de q^k la búsqueda es de alcance local.

Donde $Q^k \cap D\ (q^k, k)$ es no vacío, es decir, el conjunto admisible en el paso *k* contiene configuraciones informativas, un movimiento seguro se ejecuta hacia la nueva configuración de vista seleccionada (*avance hacia delante*). De lo contrario, el robot regresa hacia una configuración de la vista previamente visitada (*retroceder*).

La exploración se puede considerar *completa* en el paso *k*, si $Q^k = \emptyset$, es decir, no hay ninguna configuración $q \in S^k$ tal que (1) una nueva visión en q se puede extender a la zona actual explorada (2) el robot puede alcanzar q de q^k a través de un camino que es seguro en el paso *k*.

Con el fin de caracterizar completamente una estrategia de exploración, es necesario definir (i) el conjunto $D\ (q^k, k)$ (ii) la ganancia de información (iii) la estrategia de selección. En la práctica, debido a la complejidad del mapa de ε^k a S^k, también es crucial definir un procedimiento eficaz para calcular $Q^k \cap D\ (q^k, k)$.

3.5.2.5. Limite libre

Una herramienta útil, que proporciona información sobre Q^k sin necesidad de su cálculo explícito, es el *límite libre* (también llamado *frontera* en la literatura [3] - [7]) de la región explorada. En el paso *k*, el límite de la región explorada $\partial\varepsilon^k$ es la unión de dos conjuntos disjuntos:

- El *limite obstáculo* $\partial\varepsilon^k_{obs}$ el cual se realiza mediante la detección de las superficies de los obstáculos por la detección de una superficie obstáculo;

- El *limite libre* $\partial\varepsilon_{free}^{k}$, es decir, el complemento de $\partial\varepsilon_{obs}^{k}$, que conduce a las áreas potencialmente explorables.

El *límite* del obstáculo $\partial\varepsilon_{obs}^{k}$ puede ser calculada como la unión de todos los limites obstáculo visibles *B* (q^{i}), *i=1,2,…, k,* considerados por el robot hasta el *k-ésimo* paso. El límite libre $\partial\varepsilon_{free}^{k}$, entonces se calcula como de $\partial\varepsilon^{k} \setminus \partial\varepsilon_{obs}^{k}$.

El límite libre tiene algunas propiedades útiles. Dada una configuración segura *q*, se tienen las siguientes implicaciones

$$F(q) \cap \partial\varepsilon_{free}^{k} = \emptyset \Rightarrow I(q,k) = 0 \qquad \ldots\ldots\ldots (2)$$

Es decir, si ningún límite libre está presente en el campo total en q, entonces nada puede ser descubierto por la adquisición de vista de q. Por otra parte se tiene

$$\partial\varepsilon_{free}^{k} = \emptyset \Rightarrow Q^{k} = \emptyset \qquad \ldots\ldots\ldots (3)$$

De hecho, si el limite libre está vacío, no hay más puntos inexplorados en W_{free} y la exploración esta completa. De lo contrario de (2) y (3) en general no son ciertas.

A la luz de las implicaciones anteriores, y gracias al bajo costo computacional del límite, la idea central del método SET es guiar al robot hacia las configuraciones $q \in D$ (q^{k}, k) y que $F(q) \cap \partial\varepsilon_{free}^{k} \neq \emptyset$.

3.5.2.6. El método SET

En el método SET el robot construye incrementalmente la estructura de datos "árbol exploración basado en sensores" o SET por sus siglas en ingles Sensor-based Exploration Tree, cada nodo del SET representa una vista de

configuración, mientras que un arco entre 2 nodos representa un camino seguro que une las 2 configuraciones de vista.

En la figura 20 se muestra un ciclo completo de exploración del algoritmo SET. Primero daremos un comentario breve de estos pasos, y entonces discutiremos sus estructuras con algunos detalles.

El robot inicia en q^k. Primero, la estructura de datos SET y el modelo del ambiente son actualizados (línea 1). En particular, si una nueva vista de configuración ha sido alcanzada en una iteración previa: un nuevo nodo correspondiente es insertado en el SET, un nuevo arco es creado entre el previo y el nuevo nodo, la nueva vista adquirida es mezclada en el modelo del ambiente ($\varepsilon^k, \partial\varepsilon^k_{obs}$).

Enseguida, el limite libre segura LFB(q^k, k) por sus siglas en ingles *local free boundary* es extraída (línea 2). En términos generales, esto contiene alguna porción del límite libre la cual es visible para el sensor del algún $q \in D(q^k, k)$ *(se detalla más adelante).* Esto está definido de modo que puede ser fácil procesar y la siguiente implicación se mantenga:

$$\text{LFB}(q^k, k) = \emptyset \Rightarrow D(q^k, k) \cap Q^k = \emptyset \ldots\ldots\ldots (4)$$

Esto significa que $LFB(q^k, k) \neq \emptyset$ es una condición necesaria que debe ser verificada antes de la búsqueda de una vista de configuración en $D(q^k, k) \cap Q^k$ (línea 4); si $LFB(q^k, k) = \emptyset$, se ejecuta un paso de retroceso. La estrategia de selección adoptada *(se detalla más adelante)* consiste en la maximización una función de utilidad definida como U en $D(q^k, k)$. Dos estrategias de búsqueda, las cuales confían en las técnicas basadas en muestreo, son propuestas al final de esta parte. De acuerdo a (4), cuando el límite local libre es vacío no se lleva acabo búsqueda alguna y $U(q^{k+1})$ es puesto en 0 forzando a un paso de retroceso (línea 6).

En este punto la función de utilidad $U(q^{k+1})$ de la configuración de vista candidata q^{k+1} es comparada con U_{min}, un umbral mínimo fijo (línea 7).

Este tope es doble: 1) fuerza un paso de retroceso cuando el limite libre local es vacío, es decir, cuando $U(q^{k+1})$ es un conjunto 0. 2) filtros casi inútiles para las acciones del robot. Por lo tanto, si $U(q^{k+1}) > U_{min}$: q^{k+1} se convierte en la siguiente vista de configuración, el camino-planificado es invocado para procesar el camino de q^k a q^{k+1} (línea 8), q^{k+1} es alcanzado y una nueva vista es obtenida (línea 9). De otra manera, el paso de retroceso se lleva acabo y el robot se mueve a la configuración padre (línea 11).

Cuando el robot es incapaz de realizar "un avance hacia adelante" hacia los limites libres locales seguras, se fuerza un paso de retroceso a la raíz del SET (la configuración inicial), y así se realiza un mecanismo automático de retroceso (back-tracking).

3.5.2.6.1. Ganancia de información

Sea $V(q, k)$, la vista simulada la cual sería adquirida por el robot en q si el límite del obstáculo fue exactamente $\partial\varepsilon_{obs}^{k}$.

Ya que nuestro objetivo es extender ε^k tanto como sea posible, nosotros definimos la ganancia de información como $I(q, k)$, es la medida del conjunto de todos los puntos no explorados yacidos en $V(q, k)$. $I(q, k)$ puede ser procesado por el robot a través de un proceso de fundición de rayo.

Claramente, dependiendo de la representación del ambiente y la tarea del robot, otras definiciones de ganancia de información son posibles [13]. Por ejemplo, en la presencia de ruido del sensor una representación probabilística del ambiente puede ser adoptada y definiciones del límite libre basadas en entropía y la ganancia de información puede ser introducida fácilmente.

3.5.2.6.2. Conjunto admisible y límite local libre

Ya que la implicación (4) es requerida para sostener, las definiciones de conjunto admisible y limite local libre deben ser estrictamente relacionadas. En esta sub-sección, primero definiremos $D(q^k, k)$, entonces $LFB(q^k, k)$ es designada acorde a este estado. Por simplicidad, nos referimos al caso de un robot con un telémetro (m=1).

Sin pérdida de generalidad, asumimos el campo $F(q)$ en q es un cono esférico con vértice s(q), radio R (rango de percepción) y un ángulo de abertura α.

1) Conjunto admisible: En el método SET, $D(q^k, k)$ colecta todas las configuraciones q tal que (i) el centro del sensor s(q) esta dentro de una distancia máxima ρ de $s(q^k)$ (ii) s(q) y $s(q^k)$, son mutualmente visibles en el paso k, es decir el segmento de línea abierta (s(q), $s(q^k)$) no se intersecta con $\partial\varepsilon_{obs}^k$.

   El primer requerimiento, según la estructura algorítmica de re-envió impuesta, sugiere una "Estrategia de ambiente basada en primera profundidad". Esto es, el centro del sensor es guiado realizando un recorrido primero en profundidad del ambiente, regiones continuas exploradas incrementalmente. La segunda condición es un usual requerimiento en la exploración de ambientes desconocidos. Esto previene a la exploración de ambientes de continuos saltos entre cuartos continuos separados (por ejemplo: cuando $s(q^k)$ y $s(q^{k+1})$ son cercanas pero separadas pro una pared). Por otra parte, un a colección de locaciones detectadas visibles mutuamente en el mundo puede ser visto como un grafico visiblemente de locaciones reconocible conocidas con especial interés y propiedades topológicas.

2) Limite local libre: $LFB(q^k, k)$ colecta todos los puntos del limite libre $\partial\varepsilon_{free}^k$ los cuales (i) están contenidos en una bola $B(s(q^k), \rho + R)$ con

centro en $s(q^k)$ y un radio $\rho + R$ (ii) puede ser conectado a $s(q^k)$ a través de camino en el ambiente, completamente contenido en $\varepsilon^k \cap B(s(q^k), \rho + R)$. Cabe destacar que el parámetro ρ es inherente de la definición del conjunto admisible.

Es fácil de mostrar que la implicación (4) mantiene la definición de $LFB(q^k, k)$ y $D(q^k, k)$.

3.5.2.6.3. Estrategia de selección

El enfoque más común al evaluar la contribución de la acción de un robot hacia el cumplimiento de una tarea asignada al robot es la asociación de una función de utilidad. Una maximización es entonces usada para seleccionar la siguiente acción, posiblemente sobre un número finito de elecciones. Diferentes requerimientos en la tarea pueden ser trasladadas en términos parciales de utilidad/costo, los cuales son combinados en la función de utilidad total U. Una estrategia óptima debe maximizar la utilidad esperada sobre el camino total explorado, pero la complejidad del problema, junto con la falta de una información prioritaria, sugiere un enfoque más efectivo y codicioso, donde la evaluación de la utilidad de una acción esta basada en un simple pasó hacia adelante. Sin embargo, en principio, decisiones más inteligentes se vuelven posibles a mayor información obtenida acerca del ambiente.

Aquí, q^{k+1} es seleccionado en $D(q^k, k)$ así como maximizar la función de utilidad *U(q,k)= I(q,k)*. En principio el costo de navegación de q^k a q^{k+1} puede ser incluido en *U* en orden de reducir lo mayo posible comportamientos erróneos. Así que, como muestra en la siguientes secciones, la definición propuesta de $D(q^k, k)$ y una estrategia de búsqueda adecuada naturalmente limita comportamientos erróneos.

3.5.2.6.4. Estrategias de búsqueda

Durante la exploración, un modelo del espacio de configuración es incrementalmente actualizado para (i) la búsqueda de nuevas vistas de configuración y (ii) la realización de operaciones de planeación. Ya que los manipuladores típicamente tienen espacios de configuración de alta dimensión, esto es conveniente de usar simplemente basándonos en un enfoque de crecimiento de un roadmap el cual capture la conectividad de la región segura actual.

Búsqueda con crecimiento global

1. $\mathcal{G}^k \leftarrow$ se expande globalmente $\mathcal{G}^{k-1}$
2. $\tilde{D} \leftarrow$ se extrae de $\mathcal{G}^k$ el subconjunto de configuraciones caídas dentro del conjunto admisible $D(q^k, k)$
3. $(q^{k+1}, U(q^{k+1})) \leftarrow$ encuentra una configuración en $\tilde{D}$ con utilidad máxima

Figura 21: Una descripción del pseudocódigo de la estrategia de búsqueda con crecimiento global.

Búsqueda con crecimiento local

1. $T^k \leftarrow$ se expande el árbol enraizado en q^k dentro del C-space ball con centro en q^k y radio δ
2. $\tilde{D} \leftarrow$ se extrae de $\mathcal{G}^k$ el subconjunto de configuraciones caídas dentro del conjunto admisible $D(q^k, k)$
3. $(q^{k+1}, U(q^{k+1})) \leftarrow$ encuentra una configuración en $\tilde{D}$ con utilidad máxima
4. Si $U(q^{k+1}) \geq U_{min}$
5. $L^k \leftarrow$ se expande el árbol Lazy enraizado q^k dentro de $D(q^k, k)$
6. $\tilde{D} \leftarrow$ se extrae de L^k el subconjunto de configuraciones las cuales son seguras en el apso k y tienen una utilidad $U \geq U_{min}$
7. $(q^{k+1}, U(q^{k+1})) \leftarrow$ encuentra una configuración en $\tilde{D}$ alcanzable desde q^k

Figura 22: Una descripción del pseudocódigo de la estrategia de búsqueda con crecimiento local.

En particular, sea $\mathcal{G}^k$ el roadmap construido en el paso k en la región segura S^k. En $\mathcal{G}^k$, un nodo representa una configuración que es segura en el paso k, mientras un arco entre 2 nodos representa un camino local que es seguro en el paso k y que conecta las 2 configuraciones.

Una vez que ε^k es procesada fusionando $V(q^k)$ con ε^{k-1}, el roadmap $\mathcal{G}^k$ es obtenido expandiendo $\mathcal{G}^{k-1}$. Durante este proceso de expansión, configuraciones adicionales las cuales son seguras en el paso k son agregados a $\mathcal{G}^{k-1}$. En orden de encontrar estas configuraciones un chequeo de colisión es realizado en la reconstrucción del modelo del

ambiente en el paso k: de acuerdo a este modelo, ε^k es el ambiente libre disponible y $\partial\varepsilon^k$ es el límite del obstáculo. Note que, en este marco, la construcción del SET en el paso k representa el camino actual por el viajo el robot en el roadmap $\mathcal{G}^k$.

Dos instancias principales del método SET pueden ser obtenidas dependiendo de la estrategia usada para el crecimiento del roadmap. SET con "crecimiento global" (SET-GG), el cual realiza iterativamente una extensión global del roadmap. SET con "crecimiento local" (SET-LG), el cual construye el roadmap en forma de un bosque de arboles, cada crecimiento es a nivel local alrededor de vista de configuración almacenada.

1. SET-GG: en esta estrategia el roadmap $\mathcal{G}^k$ es expandido globalmente usando un enfoque basado en muestreo tal que (i) un algoritmo PRM multi-consulta o (ii) un algoritmo de árbol sencillo mono-consulta (RRT o EST). Una descripción en pseudocódigo de la estrategia es mostrado en la figura 21. En primera, el roadmap $\mathcal{G}^{k-1}$ es expandida globalmente para obtener $\mathcal{G}^k$ (línea 1). Aquí, una de las técnicas basadas en muestreo antes mencionadas es usada. Entonces, el subconjunto $\widehat{D}$ de configuraciones cae dentro del conjunto admisible común $D(q^k, k)$ es extraído de $\mathcal{G}^k$ (línea 2). Note que $\widehat{D}$ representa un estimado de $D(q^k, k)$. En este punto, q^{k+1} es encontrado como una configuración de $\widehat{D}$ con una utilidad máxima $U(q^{k+1})$ (línea 3). Cabe destacar que este enfoque inherentemente realiza un muestreo uniforme sobre el espacio de configuración libre.
2. SET-LG: Aquí, el roadmap $\mathcal{G}^k$ es construido en la forma de un bosque de arboles conectados. Cada uno de estos arboles en enraizado en una vista de configuración distinta. Por lo tanto, en este caso, un nodo del SET representa una configuración de vista junto con el árbol enraizado en esta configuración. Note que, en

cada paso, el SET-LG realiza preliminarmente un crecimiento local alrededor de q^k en el intento de maximizar localmente la función de utilidad, entonces, cuando las configuraciones no locales informadas son encontradas, se permite una búsqueda global.

Una descripción del pseudocódigo de la estrategia SET-LG es mostrada en la figura 22. Primero, un árbol T^k enraizado en la configuración actual a q^k es expandido (línea 1). Su expansión es presiona dentro de un espacio bola-C con centro en q^k y radio en δ. Aquí un algoritmo de árbol sencillo mono-consulta tal como RRT o EST pueden ser usados. Después, el subconjunto $\widehat{D}$ de configuraciones caídas dentro del conjunto admisible $D(q^k, k)$ es extraído de T^k (línea 2). En este punta una vista de configuración candidata q^{k+1} es encontrada como una configuración de $\widehat{D}$ con utilidad maximizada $U(q^{k+1})$ (línea 3).

Note que en esta etapa la expansión localmente limitada genera vistas de configuración candidatas las cuales son distantes de q^k a lo mas δ. Este mecanismo automáticamente limita el consto de navegación del siguiente movimiento del robot y evita la definición problemática de un función de utilidad mezclada (donde una penalización explicita debe ser puesta en la distancia recorrida in orden de evitar un comportamiento problemático).

Después, si $U(q^{k+1}) \geq U_{min}$, q^{k+1} se convierte en la nueva vista de configuración y la rutina de búsqueda retorna. De otra manera, la búsqueda global en el conjunto $D(q^k, k)$ es realizado. Eso es logrado en 3 pasos (línea 5-7). Primero (línea 5), un árbol $\mathcal{L}^k$ enraizado en q^k es expandido en el conjunto admisible $D(q^k, k)$: durante esa expansión se realiza un chequeo de no colisión (expansión Lazy). Aquí los RRT's son preferibles por su velocidad de expansión C-space. Después (línea 6), un subconjunto $\widehat{D}$ de configuraciones las cuales son seguras en el paso k y tienen una utilidad $U \geq U_{min}$ son extraídos de $\mathcal{L}^k$. Entonces (línea 7), un plan

mono-consulta es invocado para encontrar una configuración de $\widehat{D}$ la cual es alcanzable a través de un camino que es seguro en el paso k. Si ese plan falla en encontrar configuraciones alcanzables, entonces esto retorna $U(q^{k+1})$=0.

Cabe señalar que el SET-LG principalmente realiza un muestreo no uniforme sobre el espacio de configuraciones libre. En conclusión los distintos arboles enraizados en la vista de configuración pueden expandirse en sobre posición de las regiones de C-space. Este resultado no aceptado, puede ser casi evitado por la selección adecuada del radio δ del limitado C-space balls.

3.5.2.6.5. Planeación del camino

Una vez que una nueva vista de configuración q^{k+1} ha sido seleccionada, la planeación del camino debe procesarse un camino seguro conectando q^k a q^{k+1}. En el método SET, la planificación depende del uso de la estrategia de búsqueda.

En SET-GG, un camino seguro puede ser fácilmente encontrado en el roadmap $\mathcal{G}^k$. En SET-LG, dos casos son posibles: 1) q^{k+1} pertenece a el árbol T^k 2) q^{k+1} pertenece a el árbol Lazy $\mathcal{L}^k$. En el segundo caso la estrategia de búsqueda (figura 22, linea 7) retorna automáticamente un camino que es seguro en el paso k. Por simplicidad, este no ha sido hecho implícito en el pseudocódigo.

3.6. Experimentos y resultados

Para ilustrar el comportamiento de exploración de la estrategia SRT-Radial se presentan los resultados del algoritmo desarrollado en C# (Microsoft Visual Studio 2010), aprovechando la estructura de la librería MSL ((Motion Strategy Libray) es una librería para el desarrollo y prueba de algoritmos de planificación de movimientos,

incluye varios planificadores que usan RRT, PRM y FDP; así como un ambiente gráfico en OpenGL para la descripción de la escena.), al igual que una interfaz grafica para visualizar el ambiente de trabajo y el resultado de la exploración.

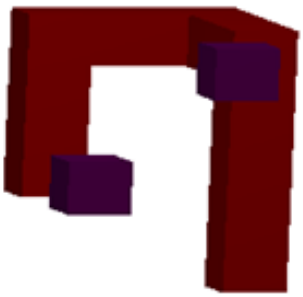
3.6.1. Detalles de la implementación

La simulación del algoritmo en condiciones de 3 dimensiones se lleva a cabo en un ambiente formado pro obstáculos y un manipulador fijo, encargado de explorar y descubrir dicho ambiente en el que se encuentra.

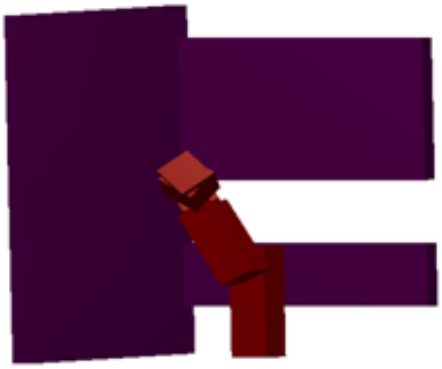
3.6.2. Resultados obtenidos con SRT-radial, en ambientes 3D

3.6.2.1. Configuraciones (ambientes)

a) Sencillo



b) Medio



c) Complejo

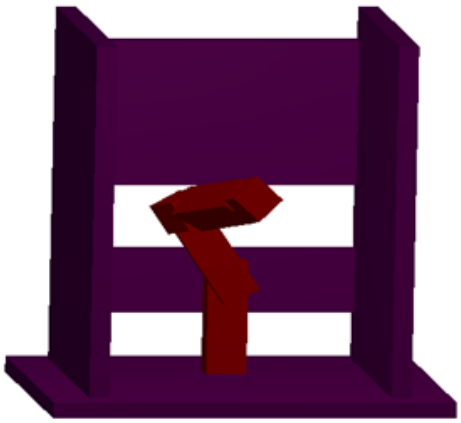


Figura 23: (a,b,c): Ambientes y posición inicial del robot para las pruebas del método SRT-Radial en 3D

3.6.2.2. Tiempos

Tiempos (seg)	a)	b)	c)
1	19.4281317	19.221465	16.6981317
2	19.2481317	19.1181317	17.9997983
3	19.221465	18.7547983	17.9997983
4	19.2731317	19.3531317	16.9581317
5	18.961465	18.6497983	17.7397983
6	19.6881317	19.091465	17.1931317
7	19.196465	19.196465	16.881465
8	19.066465	19.3247983	16.9325761
9	18.7531317	19.2997983	17.5831317
10	18.676465	19.0381317	18.2597983
Media	19.1512983	19.1047983	17.4245761

3.6.2.3. Resultados (Recorrido)

a) Sencillo

b) Medio

c) Complejo

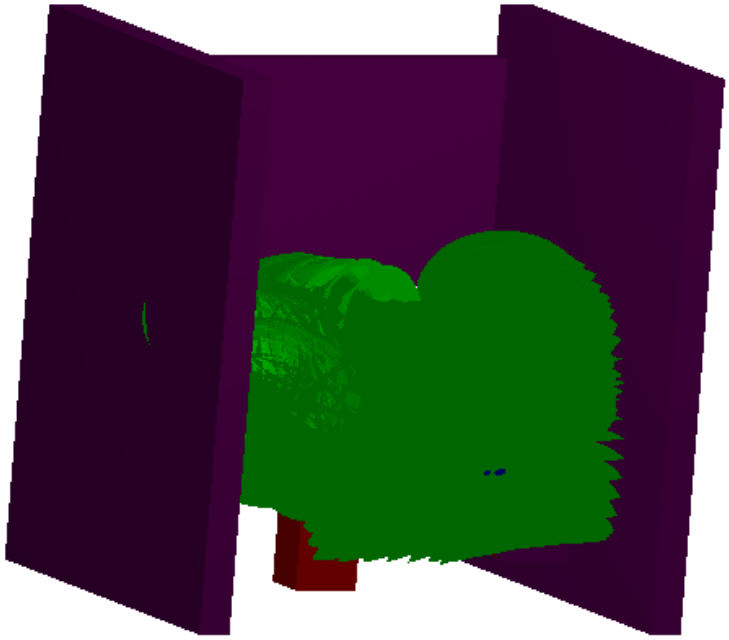
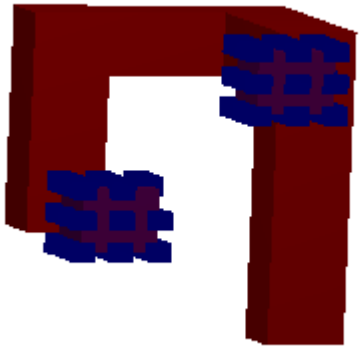


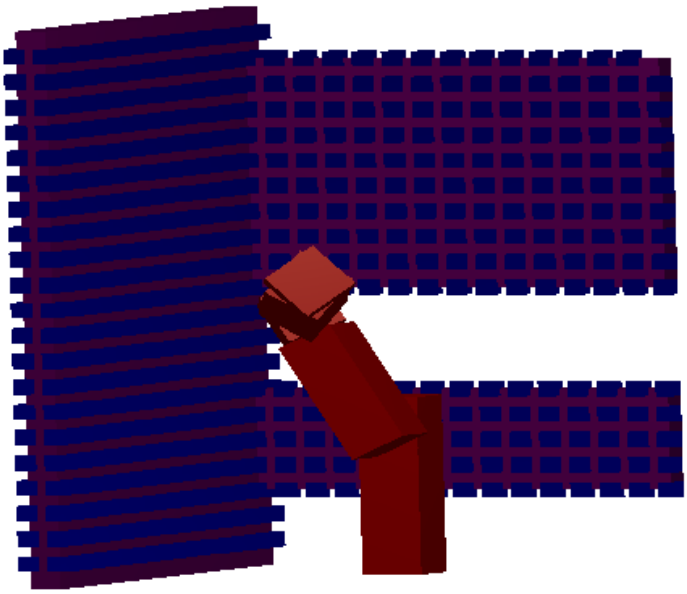
Figura 24: (a,b,c): Se muestra el uso de los conos como campo de visión de los manipuladores, durante su trabajo de exploración.

3.6.2.4. Resultados (mapa)

a) Sencillo



b) Medio



c) Complejo

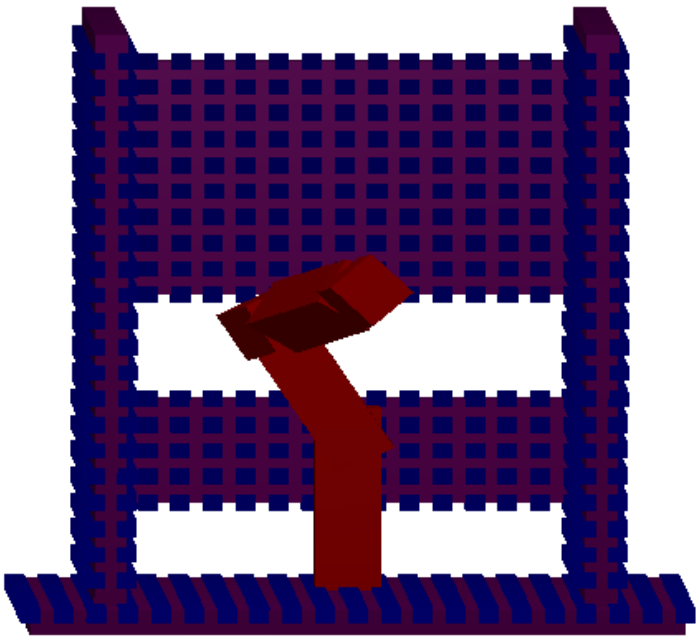


Figura 25: (a,b,c): Ambientes 3D voxelizados, devueltos por los manipuladores.

3.7. Conclusiones

Con las pruebas realizadas en los ambientes 3D, observamos la diferencia de tiempos, que arroja nuestro algoritmo, con respecto a los ambientes 2D.

Y el comportamiento hasta cierto punto es el esperado, puesto que desarrollar un algoritmo, en el cual se involucran 3 dimensiones, nos lleva a tomar en cuanta varios aspectos, que no eran necesarios en un ambiente de dos dimensiones.

Solo por mencionar, el movimiento del robot. Es un procedimiento el cual en 2 dimensiones, basta con manejar nuestras coordenadas (x,y). Pero si hablamos del correcto movimiento de un robot en un ambiente de 3 dimensiones, hablamos de procedimiento más complejos, que nos permitan obtener configuraciones validas para el robot y cada uno de sus cuerpos que a este lo forman.

Conclusiones y trabajos a futuro

Conclusiones:

Presentamos la implementación, en primer lugar, del método de exploración de ambientes desconocidos usando sensores para un robot móvil, el método SRT. Adaptamos al método, a la estrategia de percepción SRT-Radial, para aprovechar en gran parte la información proporcionada por los sensores. En seguida aplicamos y adaptamos el método para un robot móvil, usando el muestreo aleatorio, como estrategia de exploración. Esto se reflejo, al momento de decidir la dirección del siguiente estado candidato en que el robot móvil, podría colocarse. Todo lo anterior, lo englobamos en la parte del trabajo correspondiente a los ambientes bidimensionales (2D).

En segundo lugar, presentamos el método SET, usado para el manejo de manipuladores su tarea de exploración de ambientes desconocidos. Adaptamos el algoritmo SRT-Radial usando en 2 dimensiones, con la diferencia de que, ahora es un cono, el que proporciona el campo visual a nuestro manipulador en cada uno de sus estados. Y es la unión del total de los conos, la herramienta principal, para la tarea de exploración de nuestro robot manipulador. Esta segunda parte del trabajo en la correspondiente a los ambientes tridimensionales (3D).

Los resultados obtenidos a través de simulaciones muestran que las estrategias resuelven el principal problema de la exploración de ambientes desconocidos usando sensores, descubrir un ambiente donde existen obstáculos y un robot, tomando en cuenta las restricciones globales del ambiente y del robot usando la información proporcionada por los sensores en ambientes simples y complejos. La eficiencia de las estrategias varia en tiempo y en la complejidad mapa obtenido.

Trabajos futuros

Los trabajos futuros que pueden derivarse a partir de los resultados obtenidos de nuestro trabajo, principalmente son los siguientes:

- Sería ideal probar las estrategias propuestas con robots reales.
- Probar otras estrategias para la determinación del estado candidato a un nuevo proceso de percepción.
- Debido a que el método adapta los planificadores RRT, los cuales son convenientes para manejar muchos grados de libertad y espacios de altas dimensiones, un trabajo futuro es ampliar las estrategias a otros tipos de robots, con una formación y comportamientos más complejos.

Referencias

[1] P. Bessiére, E. Mazer, and J. M. Ahuactzin. *"The ariadnés clew algorithm: Global planning with local methods"*. Workshop on the Algorithmic Foundations of robotics, 1994.

[2] J. F. Canny. "The complexity of robot motion planning". MIT Press, Cambridge, MA, 1988.

[3] K.Nagatani, Y. Iwai, and Y. Tanaka. *"Sensor based navigation for car-like mobile robots using generalized voronoi graph"*. Proc. of the IEEE International Conference on Intelligent Robots and Systems, 2001.

[4] E. Kruse, R. Gutshe, and F. M. Wahl." *Efficient, iterative, sensor based 3-d map building using rating funtions in configuration space"*. Proceedings of the IEEE, Internacional Conference on Robotics & Automation, pp. 1067–1072, 1996.

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

[6] S. M. LaValle. *"Rapidly-exploring random trees: A new tool for path planning"*. Department of Computer Science, Iowa State University, 1998.

[7] S. M. LaValle and J. J. Kuffner. *"Rapidly-exploring random trees: Progress and prospects"*. Algorithmic and Computational Robotics: New Directions, pp. 293–308, 2001.

[8] S. M. LaValle and J. J. Kuffner. *"RRT-connect: An efficient approach to single-query path planning"*. Proceedings of the IEEE, Internacional Conference on Robotics & Automation, pp. 995–1001, 2000.

[9] S. M. LaValle and J. J. Kuffner. *"Randomized kinodynamic planning. Algorithmic and Computational Robotics: New Directions"*, Vol. 20, No. 5, pp. 378–400, 2001.

[10] T. Lozano-Perez. Spatial planning: A configuration space approach. IEEE Transactions on Computers, Vol. 32, No. 2, pp. 108–120, 1983.

[11] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. *"Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms".* Department of Computer Science, Old Dominion University, 1993.

[12] G. Oriolo, M. Venditelli, L. Freda, and G. Troso. *"The SRT method: Randomized strategies for exploration".* Proceedings of the IEEE, Internacional Conference on Robotics & Automation, 2004.

[13] P. Renton, M. Greenspan, H. A. Elmaraghy, and H. Zghal. "*Plan-N-Scan: A robotic system for collision-free autonomous exploration and workspace mapping".* Journal of intelligent and robotics systems, No. 24, pp. 207–234, 1999.

[14] A. Sanchez and R. Zapata. "*Sensor-based probabilistic roadmaps for car-like robots"*. Fifth Mexican International Conference in Computer Science, IEEE Computer Society, pp. 282-288, 2004.

[15] Y. Yu. *"An information theorical incremental approach to sensor-based motion planning for eye-in-hand system"*. PhD thesis, Simon Fraser University, 2000.

[16] Y. Yu and K. Gupta. "*Sensor-based probabilistic roadmaps: Experiments with an eye-in-hand system. Journal of Advance Robotics"*, Robotics Society of Japan, 2000.

[17] J.Ahuactzin and A. Portilla, *"Abasic algorithm and data structures for sensor-based path planning in unknown environments"*, in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2000, pp.902–908.

[18] B.Yamauchi, *"A frontier-based approach for autonomous exploration",* in 1997IEEE Int. Conf. on Robotics and Automation, 1997, pp.146–151.

[19] L.Freda and G.Oriolo,*"Frontier-based probabilistic strategies for sensor-based exploration"*, in IEEE Int. Conf. on Robotics and Automation, 2005, pp.3892–3898.

[20] Y. YuandK.Gupta, "*C-space en tropy: Ameasure for view planning and exploration for general robot-sensor systems in unknown environments",* Int. J. Robotics Research, vol.23, no. 12, pp.1197–1223, 2004.

[21] Rob Hess, *"Adventures in Voxel Coloring CS 559 Term project*", Department of Computer Science, pag. 7, Oregon State University, March 17, 2005.

[22] Seitz, S., and Dyer, C. "*Photorealistic scene reconstruction by voxel coloring*". Int. Journal of Computer Vision 35, 2 (1999).

[23] A. Sánchez López. *"Geometría solida constructiva"*, Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Abril 2007.

[24] A. Sánchez López. *"Cinemática directa e inversa Parte I"*, Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Abril 2012.

[25] G. Sarabia Cortés. *"Planificación reactiva de movimientos para robots y manipuladores usando técnicas probabilísticas, Tesis de maestría"*, Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Primavera 2012.

[26] Craig, John J. *"Introducción a la robótica"*, 3 Ed, Pearson educación, México 2006.

[27] J. Irving Vásquez Gómez *"Planificación de vistas para reconstrucción tridimensional de objetos con robots móviles"*, Tesis de maestría, Instituto Nacional de Astrofísica, Óptica y Electrónica, Centro de investigación en matemáticas, Agosto 2011.

REFERENCIAS

[28] Yong Yu, *"An information theoretical incremental approach to sensor based motion planning for eye in hand system"*, B.Sc., Xi'an Jiaotong University, China, M.Sc., Northeast University, China 2000.

[29] F. Vecchioli, L.Freda and G.Oriolo,*"Sensor-based exploration for general robotic systems*", Universidad de Roma "La Sapienza", 2008.

[30] Judith Espinoza León, estudiante de maestría en ciencias de la computación, "*Estrategias para la exploración de ambientes desconocidos en robótica móvil"*, facultad en Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Mayo 2006.