



**Benemérita
Universidad Autónoma de Puebla**

Facultad de Ciencias de la Computación

**“Exploración basada en sensores para
robots móviles”**

Tesis de postgrado

Que para obtener el grado de:
Maestro en Ciencias de la Computación

**Presenta:
Omar Torres Acuitlapa**

**Asesor:
Dr. Abraham Sánchez López**

Puebla, Pue.

Mayo 2014

Índice general

1. Introducción	5
1.1. Exploración integrada	6
1.2. Trabajos relacionados	7
1.3. Contribución	10
2. Método SRT	11
2.1. Método SRT	11
2.1.1. Exploración de ambientes desconocidos con el método SRT	13
2.1.2. Hipótesis de trabajo	13
2.1.3. Algoritmo SRT	14
2.2. Corrección de la región segura local	17
2.2.1. Algoritmo	18
2.3. Exploración con SRT-Radial	24
2.4. Conclusiones	27
3. Método SET	28
3.1. Descripción del problema	30
3.2. Robot y modelos del mundo	30
3.3. Modelo del sensor	31
3.4. La tarea de exploración	32
3.5. Estrategias de exploración	33
3.6. Límite libre	34
3.7. El método SET	35
3.7.1. Ganancia de la información	36
3.7.2. Conjunto admisible y definición del límite local libre	37
3.7.3. Estrategia de selección	38
3.7.4. Estrategias de búsqueda	38
3.7.5. Planificación del camino	41

4. Resultados obtenidos	42
4.1. Resultados en ambientes 2D	42
4.2. Discusión	51
5. Conclusiones y trabajo futuro	52
5.1. Conclusiones	52
5.2. Trabajo futuro	53
A. Librería GPC	54
A.1. Descripción	55
A.2. Funciones	55
A.3. Huecos y contornos externos	58
A.4. Asociación de los huecos con el contorno externo	59
A.5. Lados coincidentes y casi-coincidentes	60
Lista de figuras	61
Lista de tablas	63
Bibliografía	64

Dedicada
a mi hija Jade
y mi esposa Hilda

Agradecimientos

En primer lugar a Dios por haberme permitido alcanzar una meta más en mi vida.

A mi esposa Hilda y mi hija Jade por darme la fuerza para seguir cada día.

A mis padres Miguel y Sabela porque siempre me han apoyado, guiado y cuidado.

A mis hermanos Alan y Cesar por su apoyo y cariño.

En especial a mi Asesor de Tesis, Dr. Abraham Sánchez López por su experiencia, paciencia y su motivación brindados hacia mi persona.

Capítulo 1

Introducción

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 preplaneado 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, el cual se sustenta en información local (o parcial) y b) el sensado que es parte integral de la navegación. A causa del primer aspecto, los algoritmos para ambientes desconocidos con frecuencia se les llaman algoritmos en línea. 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. Para una navegación segura es de suma importancia contar con un método de localización.

La necesidad de localizar a un robot en su ambiente se manifiesta en diferentes circunstancias. Un robot no puede alcanzar un objetivo definido a partir de sus propias coordenadas en un sistema relativo a su ambiente si él no conoce sus propias coordenadas respecto a este ambiente.

Ciertas decisiones no pueden ser tomadas sin un conocimiento de la localización del robot. Si un robot ejecuta un movimiento en el cual un obstáculo

imprevisto aparece, el robot para tomar una decisión de movimiento o enviar una señal de alerta al operador debe tener conocimiento de su localización respecto a éste. Con mayor claridad el robot debe responderse: ¿Cuál es mi posición respecto a dicho objeto? Con este conocimiento el robot o el humano podrán entonces tomar una decisión de giro, alto o continuar según las circunstancias.

1.1. Exploración integrada

Un robot móvil que opera en el mundo físico debe conocer su entorno. Una gran parte de este conocimiento es saber donde esta el robot en el mundo (la tarea de *localización*) y donde ha estado (la tarea de *mapeo*). En ausencia de la localización externa, el robot debe ser capaz de construir un mapa y, al mismo tiempo, localizarse en el mismo mapa, aún cuando este mapa sea imperfecto y este parcialmente construido (*localización simultanea y construcción del mapa* o SLAM [22]).

A menudo la creación del mapa se considera una meta (la tarea de *exploración*). Se necesita una *estrategia de exploración* para responder a la pregunta de dónde ir después con el fin de construir el mapa de manera eficiente. Un enfoque adecuado para la exploración requiere una evaluación equilibrada de las medidas alternativas de movimiento desde el punto de vista de la ganancia de la información, los costos de navegación, localización y calidad. Las soluciones existentes alcanzan varios grados de integración entre las tareas de localización, mapeo y control de movimiento. La Figura 1.1 ilustra el ámbito de la exploración robótica, con tres regiones que se superponen parcialmente. La región I representa gráficamente la integración de la localización y el mapeo implementados por varias familias de algoritmos de SLAM [22, 35, 7]. La región II representa la integración del mapeo y del control de movimiento, ejemplificados por casi la totalidad de las estrategias de exploración [21, 37, 31, 33, 26]. La región III, que integra la localización y el control de movimiento en el campo de la navegación activa [35, 11, 30] y la gestión de sensores [25]. La plena integración de los tres componentes [17, 7, 2] en la región IV y se denominará en lo sucesivo *la exploración integrada*.

Se busca que una estrategia de exploración, tenga un balance de la cobertura, precisión y velocidad de exploración, los factores de control básicos en cualquier misión de exploración. Estos controles son a menudo contradictorios y se requiere de un orden para seleccionar una métrica adecuada de acción. Además de los tradicionales parámetros de la información obte-

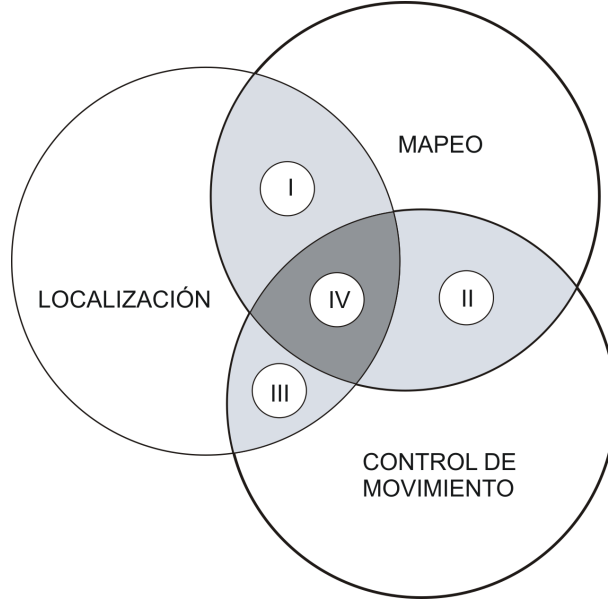


Figura 1.1: El campo de la exploración robótica con las regiones de integración destacadas.

nida y tiempo (o distancia) recorrido, se introduce a menudo una métrica de *localizabilidad*, lo que permite comparar la calidad de la localización en diferentes lugares.

1.2. Trabajos relacionados

Un requisito fundamental para la navegación autónoma es el conocimiento del modelo de un ambiente, usualmente conocido como mapa. La exploración es la tarea básica de un robot que construye este modelo a través de su sistema sensorial para su uso posterior. En cada paso, con el fin de ser capaz de fusionar los datos del sensor consistentemente en el mapa actual, la posición del robot debe ser estimada, para este fin, se debe realizar un proceso de localización simultánea. Por lo tanto, una eficiente estrategia de la exploración debe tomar en cuenta diferentes criterios, posiblemente criterios en conflicto cuando seleccionamos la siguiente acción: su costo (en términos de tiempo o la energía), la información que se espera obtener y la asociación del potencial de localización.

Muchas estrategias de exploración eligen la siguiente acción sobre la base de sólo un subconjunto de los criterios mencionados. Por ejemplo, las técni-

cas propuestas en [37, 12], se pueden clasificar como basadas en fronteras, ya que estas toman en cuenta - aunque en diferentes formas - la información obtenida asociada a cada acción y al costo final de la exploración. Por otro lado, las técnicas de localización activa, como por ejemplo [17], tratan de aumentar la calidad de localización a lo largo del camino elegido sin tener en cuenta el costo asociado o información obtenida. De hecho, la mayoría de los algoritmos de SLAM (por ejemplo, ver [16] y las referencias en él) no se ocupan de la tarea de planificación.

Recientemente, diversos investigadores han enfatizado en la importancia de la integración del mapeo, localización y planificación. Un enfoque común es la asociación de una función de utilidad a cada uno de estos procesos, a fin de evaluar la contribución de las siguientes acciones candidato para el cumplimiento de la tarea correspondiente. Algún tipo de minimización de un criterio mixto (la utilidad total) combina las funciones de utilidad individual, que se usan para seleccionar la siguiente acción, posiblemente sobre un número finito de elecciones. Una estrategia óptima debe maximizar la utilidad esperada durante todo el camino de la exploración, pero la complejidad del problema junto con la falta de información a priori, sugieren un enfoque voraz más efectivo, donde la evaluación de la utilidad de una acción se basa en un paso simple de mirar adelante. Como consecuencia de ello, la selección de acciones en las estrategias de exploración integradas es inherentemente local. Sin embargo, en principio, las decisiones más inteligentes vienen de la mayor información obtenida del ambiente.

El paradigma de la exploración integrada fue por primera vez expuesta explícitamente en [24]: el método propuesto, basado en un enfoque de información teórica [4], asocia a cada posible destino del robot un estimado de la ganancia de la información del mapa obtenida lo mejor posible de la posición determinada que se ha alcanzado hasta ese punto. Estas aproximaciones son respectivamente usadas en las funciones de mapeo y localización, además, una utilidad de navegación se adiciona a la función total, para obtener el camino más corto. Una aproximación similar es usada en [10], donde la decisión local es tomada de tal manera que mejore las estimaciones del robot y la localización de marcas. En [34] otra aproximación de decisión teórica es propuesta, en la cual cada acción es seleccionada de manera que reduzca la incertidumbre posterior acerca de la trayectoria del robot y el mapa resultante. La función de utilidad total adoptada combina el cambio esperado de la entropía y el filtro de partícula Rao-Blackwellized seleccionado y el costo de alcanzar el posible destino. Otra técnica relacionada incluida fue propuesta en [32], el SLAM se usa para dar una clase parametrizada de trayectorias espirales y maximizar una función de utilidad que combina el

costo de navegación y la incertidumbre adoptada del filtro EKF (filtro extendido de Kalman) y en [14], donde la calidad de la localización es obtenida directamente considerando como requisito un mínimo de traslape entre la nueva observación esperada y el mapa disponible.

Ya que existe una gran cantidad de trabajos en la exploración robótica, debido en gran parte al papel central desempeñado por la navegación y mapeo en cualquier aplicación de robótica móvil. Lee [21] hace una revisión de las estrategias de exploración hasta mediados de la década de 1990. Para trabajos más recientes, el lector podría revisar el libro de Probabilistic robotics [36]. Esta sección está organizada de acuerdo a las regiones descritas en la Figura 1.1.

Exploración Clásica (II). Yamauchi [37] introdujo el ahora popular método de exploración de fronteras. Este trabajo es extendido en [31], acoplándolo con un método de localización mejorado, dando como resultado una mayor integración del mapa representado; pero la tarea de localización y exploración siguen siendo completamente desacoplados. La información manejada por la estrategia de exploración es extendida a las múltiples fuentes de información en [26], pero la localización no es considerada una parte integral de la estrategia de exploración.

SLAM y Exploración (I & II). En [33] múltiples robots cooperativamente exploran un ambiente de interiores, cada uno usa un algoritmo SLAM para la localización y mapeo. La estrategia de exploración está basada en el método de frontera, pero no aborda la cuestión de la calidad de la localización.

Localización Activa (III). Un algoritmo voraz se emplea a nivel local en [17] para seleccionar las acciones de control de movimiento que puedan reducir la incertidumbre de la localización. En la navegación por costas, un método de planificación de trayectorias permite aumentar la robustez de la localización en espacios largos con pasajes estrechos [30]. La distribución de las métricas de localizabilidad se calcula fuera de línea usando un mapa de rejilla de ocupación a priori.

SLAM y Localización Activa (I & III). La localización estocástica activa se implementó en [11]. El algoritmo selecciona de una lista de acciones posibles de movimiento ponderando la consiguiente reducción de la incertidumbre de localización con el costo asociado. La tarea de exploración no se considera.

Exploración Integrada (IV). Feder et al. [10] es el trabajo más estrechamente relacionado. El vehículo crea un mapa y se localiza simultáneamente, toma las decisiones locales de donde será el próximo movimiento con el fin de minimizar el error en las estimaciones de la actitud del vehículo y las localizaciones de las marcas. Este principio se aplica para el problema de la exploración submarina en [2]. El manejo para minimizar la actitud del vehículo y los errores del mapa se incorporan en una arquitectura general de comportamiento. En ambos casos la estrategia de exploración es parcialmente local y la métrica de localizabilidad está basada en cálculos sencillos.

1.3. Contribución

Las contribuciones mas importantes de esta tesis son:

- Una estrategia basada en visibilidad para simular de manera más real el sensado del ambiente.
- La implementación de un algoritmo general para exploración de ambientes desconocidos en robótica móvil (SET).
- La integración de dicha estrategia de exploración en la herramienta SRT-BUAP.

El capítulo 2 está dedicado a la corrección de la región segura local obtenida de la información de los sensores. En el capítulo 3 se estudia con detenimiento el método para exploración de ambientes desconocidos usando sensores, el método SRT. En el capítulo 4, se presenta la exploración basada en fronteras para ambientes desconocidos. Los resultados obtenidos con las experimentaciones en simulación se presentan en el capítulo 5. Finalmente, las conclusiones se plantean en el capítulo 6 haciendo una recapitulación del trabajo realizado y un análisis de la técnica propuesta planteando las posibles mejoras para trabajos futuros y posibles ampliaciones del proyecto.

Capítulo 2

Método SRT

Oriolo, Vendittelli, Freda y Troso presentan en [28] 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). Pueden obtenerse diversas estrategias de exploración adaptando al método general, diferentes técnicas de percepción. En [28] 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. La estrategia desarrollada en esta tesis sigue las dos líneas. Judith Espinoza propuso [9] un método llamado SRT-Radial, el cual mejora la eficiencia del algoritmo y es el que tomamos para el desarrollo del presente trabajo.

2.1. 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.

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 un caso de aprendizaje activo de orden-sensitivo, por lo que el flujo de datos es resultado de todas las acciones pasadas del robot. El problema central de la exploración es como 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 última 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.

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 orientados a objetivos utilizando la selección aleatoria de acciones. 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.

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 RSL's 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 usando en misiones planetarias), pero no podemos omitir que tal suposición a menudo es ilógica en ambientes desconocidos y no estructurados. Como veremos en el capítulo de resultados en la práctica, el mapa obtenido sin localización no es el adecuado, se requiere del módulo de localización.

2.1.1. 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 el punto de vista general, es decir, independiente de una estrategia de percepción particular. Finalmente, se presenta la variante adoptada y los resultados obtenidos por el medio de herramienta de simulación desarrollada.

2.1.2. 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, R^2 o subconjunto de 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 esta 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 por que de lo contrario ningún movimiento puede ejecutarse después del primer sensado.

2.1.3. 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 libres. 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 2.1.

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 la Figura 2.2. Una nueva configuración candidata q_{cand} se determina tomando un paso de longitud $\alpha \cdot r$ en dirección a θ_{rand} . La constante $\alpha < 1$ garantiza que q_{cand} se encuentre en el área segura S y puede alcanzarse a través de un camino contenido en S ; valores próximos a 1 incrementan la capacidad de exploración del algoritmo, mientras que valores más pequeños aumentan el margen de seguridad.

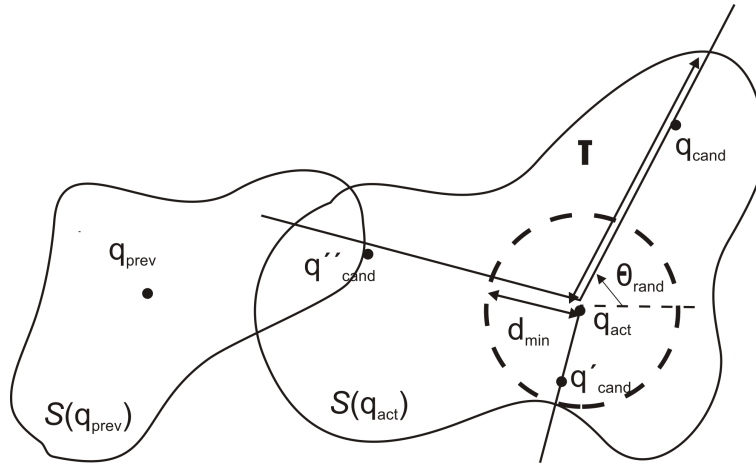
Una vez generada q_{cand} de forma aleatoria en la región segura S , pasa por un proceso de validación efectuado por la función `VALIDA`. Como se muestra en la Figura 2.2, q_{cand} :

```

CONSTRUIR_SRT( $q_{ini}, k_{max}, I_{max}, \alpha, d_{min}$ )
1   $q_{act} = q_{ini}$ ;
2  para  $k = 1$  a  $k_{max}$ 
3     $S \leftarrow \text{PERCEPCION}(q_{act})$ ;
4     $\text{AGREGA}(T, (q_{act}, S))$ ;
5     $i \leftarrow 0$ ;
6    repetir
7       $\theta_{rand} \leftarrow \text{DIR\_ALEATORIA}$ ;
8       $r \leftarrow \text{RADIO}(S, \theta_{rand})$ ;
9       $q_{cand} \leftarrow \text{DESPLAZA}(q_{act}, \theta_{rand}, \alpha \cdot r)$ ;
10      $i \leftarrow i + 1$ ;
11   hasta que ( $\text{VALIDA}(q_{cand}, d_{min}, T)$  o  $i = I_{max}$ )
12   si  $\text{VALIDA}(q_{cand}, d_{min}, T)$  entonces
13      $\text{MOVER\_A}(q_{act})$ ;
14      $q_{act} \leftarrow q_{cand}$ ;
15   sino
16      $\text{MOVER\_A}(q_{act}.\text{padre})$ ;
17      $q_{act} \leftarrow q_{act}.\text{padre}$ ;
18   Regresa  $T$ 

```

Figura 2.1: Algoritmo básico de construcción del SRT.



(a) En este caso, q_{cand} , es valida, mientras q' y q'' no lo son, la primera se encuentra a una distancia menor a d_{min} de q_{act} y q'' se ubica en la región segura local de otro nodo.

Figura 2.2: Generación de configuraciones candidatas con el método SRT.

- (i) Debe estar alejada de q_{act} a una distancia mayor a una distancia mínima prefijada d_{min}
- (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 válida o exceder un número máximo de intentos, I_{max} . En el último caso, el robot regresa al nodo padre de q_{act} , donde se ejecuta nuevamente el ciclo. Típicamente, cuando el espacio libre ha sido explorado completamente, el algoritmo fallará en encontrar una nueva dirección de exploración y el robot hará un proceso automático de retorno a la configuración inicial.

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

- En comparación con el RRT, la estructura de 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 sustancialmente 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 en una consecuencia de la naturaleza del recorrido en profundidad del algoritmo SRT.
- El método SRT 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. Corrección de la región segura local

De acuerdo a los resultados obtenidos con el método SRT sin localización, notamos que hay ciertas partes de la región segura local que sobrepasan los obstáculos, esto se debe a las características propias del método y a la librería que utilizamos para simular los sensores (ver Apéndice A). Para poder contar con un modelado del sensor de manera más adecuada, se implementó un algoritmo para el cálculo de la visibilidad de la región segura local y con esto se asegura que no existan efectivamente zonas más allá de los obstáculos, como sucede al utilizar el método SRT. Podemos afirmar entonces que el nuevo método SRT es efectivamente un método basado en fronteras dado que la región segura local se calcula de manera correcta (ver Figura 2.3).

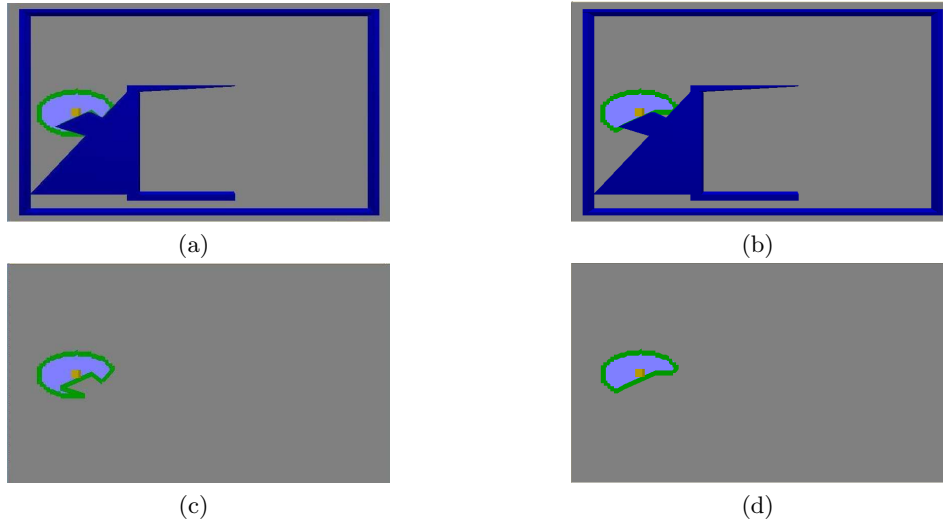


Figura 2.3: Las figuras muestran la comparativa de la RS sin visibilidad a) y la RS con visibilidad b), en las siguientes figuras c) y d) se muestran la misma comparativa ocultando los obstáculos.

La visibilidad es un fenómeno diario en nuestra vida. Observamos objetos a nuestro alrededor y decidimos nuestros movimientos acorde a estos, ver un objeto significa identificar las porciones del objeto visible desde la posición actual de un observador. Todo el objeto puede no ser visible en algunas de sus partes que pueden estar ocultas al observador. El observador también determina las formas y tamaños de las porciones visibles de un objeto. Las porciones visibles de un objeto cambian como el observador se mueva de una posición a otra.

Por otra parte, el observador puede ver varios objetos en diferentes direcciones desde su posición actual, las partes visibles de estos objetos forman la escena alrededor del observador. La construcción continua de la escena es muy natural para un observador humano debido a que el sistema visual humano puede ejecutarlo sin esfuerzo.

Supongamos que un robot quiere moverse desde una posición inicial a una posición meta sin colisionar con algún objeto u obstáculo alrededor. El robot construye la escena alrededor de él desde su posición actual y entonces guía su movimiento en el espacio libre que está entre él y la porción visible de los objetos alrededor de él. La posición del robot y de los objetos puede ser representada en la computadora del robot por sus coordenadas x , y y z y, por lo tanto, la escena consiste de porciones visibles de esos objetos que pueden ser calculados por la posición actual del robot. El problema del cálculo de porciones visibles de objetos dados desde un punto de vista ha sido estudiado extensivamente en gráficas computacionales.

2.2.1. Algoritmo

Un polígono P es definido como una región cerrada R en el plano delimitado por un conjunto finito de segmentos de línea (llamados *borde* de P) de tal manera que existe un camino entre dos puntos de R el cual no intersecta ningún lado de P . Cualquier punto final de un borde de P es llamado un *vértice* de P , el cual es un punto en el plano.

En esta sección presentaremos el algoritmo de Lee [20] para calcular el polígono de visibilidad $V(q)$ de un polígono simple P de n vértices desde un punto q .

El primer paso del algoritmo es determinar si q esta dentro o fuera de P . Si q esta dentro de P , un simple polígono P' es construido desde P tal que $q \in P'$ y $V(q) \subseteq P'$. Entonces, el procedimiento para calcular el polígono de visibilidad desde un punto interno puede ser usado para calcular $V(q)$ en P' tal que $q \in P'$.

Se explicara el procedimiento para construir P' desde P con q dentro de P . Existen dos situaciones dependiendo si q esta dentro de la envolvente convexa de P (ver Figura 2.4). La envolvente convexa de P es el polígono convexo más pequeño que contiene a P y puede ser calculado por el algoritmo de Graham y Yao [15].

Si q esta fuera de la envolvente convexa de P (ver Figura 2.4a), se dibujan dos tangentes (qv_i y qv_j) desde q a la envolvente convexa de P . Sea $bd(P)$ la frontera de P . Observamos todos los puntos de $bd(P)$ visibles desde q que están entre v_i y v_j frente a q . Ahora, $bd(P')$ consiste de las partes de $bd(P)$ entre v_i y v_j y dos tangentes qv_i y qv_j . Ahora q es un punto interno de P' .

Se considera otra situación donde q esta fuera de P pero dentro de la envolvente convexa de P (ver Figura 2.4b). Se dibuja una línea desde q pasando por algún vértice v_k de P (denotado como $\overrightarrow{qv_k}$). Sea q' el punto más cercano a q entre todos los puntos de intersección de $\overrightarrow{qv_k}$ con $bd(P)$. Empezando desde q' a través de $bd(P)$ en sentido de las manecillas del reloj (y en sentido contrario de las manecillas del reloj) hasta un vértice v_i de la envolvente convexa (respectivamente, v_j) es alcanzado. Notamos que v_i y v_j son vértices consecutivos en la envolvente convexa de P . Ahora, $bd(P')$ consiste del $bd(P)$ entre v_i y v_j que contiene q' , y el lado de la envolvente convexa $v_i v_j$. Ahora, q es un punto interno de P' .

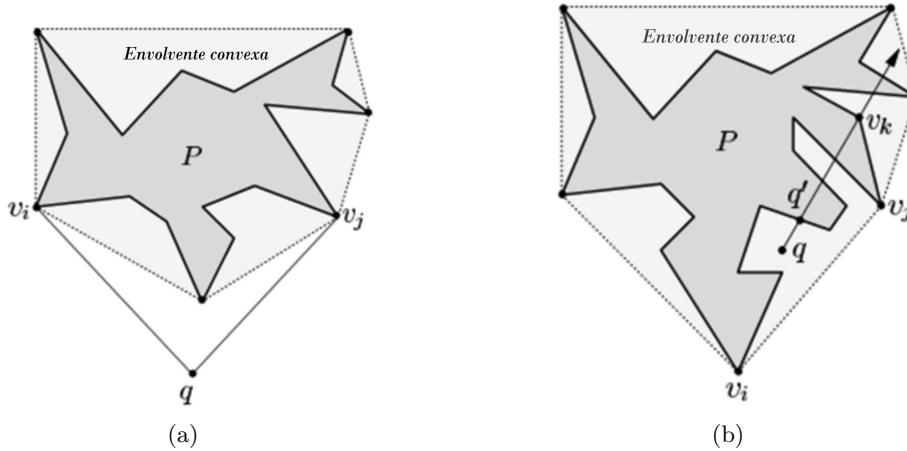


Figura 2.4: (a) El punto q esta fuera de la envolvente convexa de P . (b) El punto q esta fuera de P pero dentro de la envolvente convexa.

Se sabe que $V(q)$ es un polígono en forma de estrella, donde q es un punto en el núcleo de $V(q)$. Sea v_0 el punto más cercano a q de entre los puntos interseccionados de $bd(P)$ con la línea horizontal trazada desde q a la derecha de q (ver Figura 2.5a). Asumimos que los vértices de P son etiquetados como $v_1, v_2, \dots, v_n$ en sentido contrario de las manecillas del reloj con v_1 como el siguiente vértice en sentido contrario de las manecillas del reloj después de v_0 . Ahora, v_1 y v_n están en la izquierda y derecha de $\overrightarrow{qv_0}$, respectivamente.

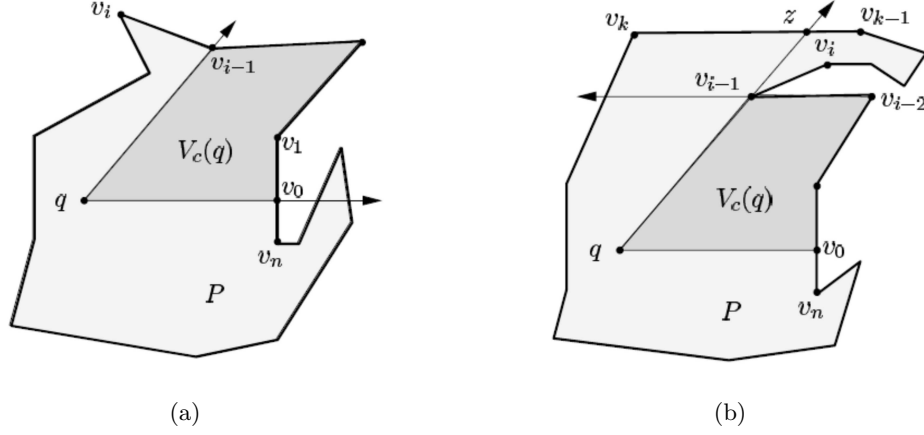


Figura 2.5: (a) El vértice v_i es insertado en la pila. (b) El vértice de $bd(v_i, v_{k-1})$ no es visible desde q .

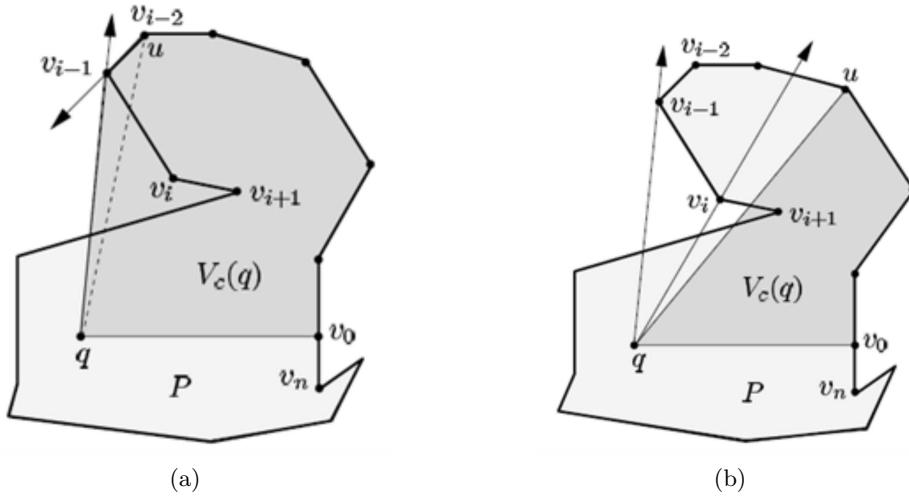


Figura 2.6: (a) El lado $v_{i-1}v_i$ intersecta uq . (b) El lado $v_{i-1}v_i$ no intersecta uq .

Asumimos que el procedimiento para calcular $V(q)$ debe escanearse $bd(P)$ en sentido contrario de las manecillas del reloj desde v_1 a v_{i-1} y v_i es el vértice actual en consideración. El polígono en forma de estrella formados por los vértices y puntos en la pila a lo largo de las fases con q es referido como la región visible actual $V_c(q)$. Sea $bd(v_j, v_k)$ la frontera en sentido contrario de las manecillas del reloj de P desde v_j a v_k .

También asumiremos que los vértices (y los puntos finales de los lados construidos) en $bd(v_0, v_{i-1})$, los cuales fueron encontrados visibles desde q por el procedimiento, son puestos en una pila en el orden que sean encontradas, donde v_0 y v_{i-1} son el fondo y el final de la pila respectivamente. Ahora los vértices y puntos en la pila son clasificados angularmente y ordenados alrededor de q . El procedimiento siempre asegura que el contenido de la pila satisface las propiedades para cualquier estado de la ejecución. Tenemos los siguientes casos:

- caso 1: El vértice v_i está a la izquierda de $\overrightarrow{qv_{i-1}}$ (Figura 2.5a).
- caso 2: El vértice v_i está a la derecha de $\overrightarrow{qv_{i-1}}$ (Figura 2.5b y Figura 2.6a).
 - a El vértice v_i está a la derecha de $\overrightarrow{v_{i-2}v_{i-1}}$ (Figura 2.5b).
 - b El vértice v_i está a la izquierda de $\overrightarrow{v_{i-2}v_{i-1}}$ (Figura 2.6a).

Considerando el caso 1. Desde v_i y el vértice y puntos en la pila están en la clasificación angular ordenados con respecto a q (ver Figura 2.5a), v_i se coloca en la pila.

Considerando el caso 2. Puede verse que v_{i-1} y v_i no son visibles desde q (ver Figura 2.5b y Figura 2.6a), si algún qv_i es intersectado por $bd(v_0, v_{i-1})$ (caso 2a) o qv_{i-1} es intersectado por $bd(v_{i+1}, v_n)$ (caso 2b).

Considerando el caso 2a. El vértice v_i y alguno de los subsecuentes vértices de v_i (aun estando escaneados) no son visibles desde q (ver Figura 2.5b). Sea $v_{k-1}v_k$ el primer lado de v_{i+1} en $bd(v_{i+1}, v_n)$ en sentido contrario a las manecillas del reloj tal que $v_{k-1}v_k$ intersecte $\overrightarrow{qv_{i-1}}$. Sea z el punto de intersección.

Considerando el caso 2b. El vértice v_{i-1} y algunos de los vértices preceden de v_i (actualmente en la pila) no son visibles desde q (ver Figura 2.6a). Se saca de la pila para remover v_i . Sea u el vértice en el tope de la pila. El lado $v_{i-1}v_i$ es llamado un lado delantero. Mientras $v_{i-1}v_i$ intersecta uq y u es un vértice de P , se saca de la pila.

Notese que se sacaron vértices que no son visibles desde q como su visibilidad es bloqueada por $v_{i-1}v_i$. Antes de la ejecución de este paso de retorno, existen dos situaciones que se pueden resaltar: (i) $v_{i-1}v_i$ no intersecta uq (ver Figura 2.6b), y (ii) $v_{i-1}v_i$ intersecta uq (ver Figura 2.8).

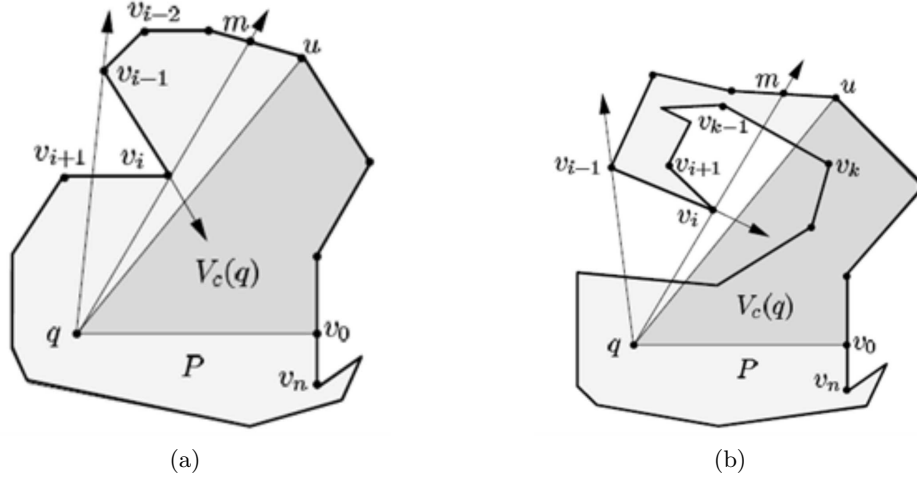


Figura 2.7: (a) El retroceso termina insertando m y v_i en la pila (b) El retroceso continua con $v_{k-1}v_k$ como el lado actual.

En la primera situación el procedimiento decide si un regreso más largo es requerido (ver Figura 2.6b y Figura 2.7). Si v_{i+1} está a la derecha de $\overrightarrow{qv_i}$ (ver Figura 2.6b), el retroceso continua con $v_i v_{i+1}$ como el actual lado delantero. Sea m el punto de intersección de $\overrightarrow{qv_i}$ con el borde poligonal u . Si v_{i+1} está a la derecha de $\overrightarrow{v_{i-1}v_i}$, entonces el retroceso termina (ver Figura 2.7a). Se inserta m y v_i en la pila y v_{i+1} se vuelve el nuevo v_i . Si v_{i+1} está a la izquierda de $\overrightarrow{v_{i-1}v_i}$ (ver Figura 2.7b), se escanea $bd(v_{i+1}, v_n)$ desde v_{i+1} hasta encontrar un vértice v_k tal que el lado $v_{k-1}v_k$ interseque mv_i . El retroceso continua con $v_{k-1}v_k$ como el lado delantero actual.

En la segunda situación, u no es un vértice de P (ver Figura 2.8). Sea w el vértice inmediato debajo de u en la pila. Ahora, uw es un lado construido rápido por el procedimiento en el caso 2. Sea p el punto de intersección de uq y $v_{i-1}v_i$. Si $p \in qw$ (ver Figura 2.8a), la visibilidad de ambos u y w desde q es bloqueada por $v_{i-1}v_i$. Sacar de la pila. Se continua con el retroceso y $v_{i-1}v_i$ sigue siendo el borde actual. Por otra parte, $v_{i-1}v_i$ fue intersectado por uw como p pertenecen a uw (ver Figura 2.8b). Escanear $bd(v_{i+1}, v_n)$ desde v_{i+1} hasta que un vértice v_k sea encontrado tal que el lado $v_{k-1}v_k$ interseque en algún punto (punto, z). Ahora todo el $bd(w, z)$ (excluyendo w y z) no son visibles desde q . Sacar de la pila. Insertar z y v_k en la pila. Entonces v_{k+1} regresa el nuevo v_i .

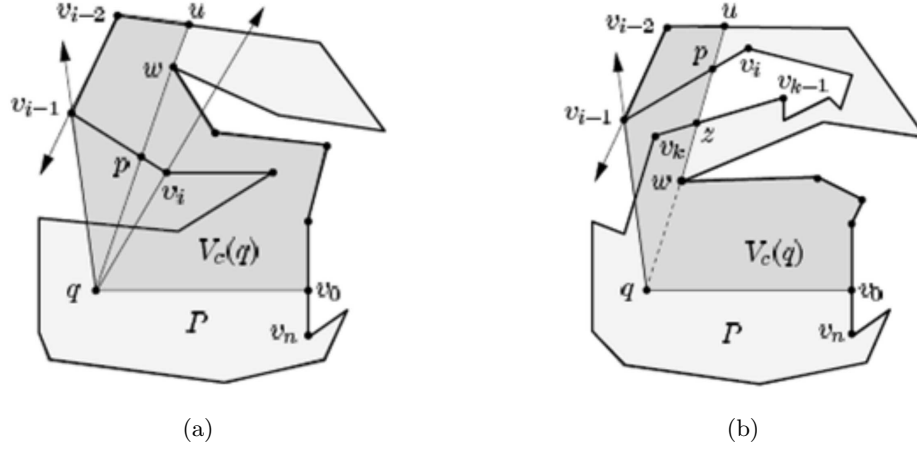


Figura 2.8: (a) El lado $v_{i-1}v_i$ no intersecta el lado construido uw . b) El lado $v_{i-1}v_i$ intersecta el uw en p y el lado $v_{k-1}v_k$ intersecta pw en z .

En los siguientes pasos se presenta formalmente el algoritmo para calcular $V(q)$. Como antes, asumiremos que v_0 es el punto más cercano de $bd(P)$ a la derecha de q y el vértice v_1 es el siguiente vértice en sentido contrario de las manecillas del reloj de v_0 . Se inserta v_0 en la pila y se inicializa i con 1.

1. Inserta v_i en la pila y $i := i + 1$. Si $i := n + 1$ ir al paso 8.
2. Si v_i está a la izquierda de $\overrightarrow{qv_{i-1}}$ (Figura 2.5a), ir al paso 1 (caso 1).
3. Si v_i está a la derecha de ambos $\overrightarrow{qv_{i-1}}$ y $\overrightarrow{v_{i-2}v_{i-1}}$ entonces (caso 2a):
 - a) Escanear desde v_{i+1} en sentido contrario de las manecillas del reloj hasta encontrar un vértice v_k tal que $v_{k-1}v_k$ intersecte $\overrightarrow{qv_{i-1}}$ (Figura 2.5b). Sea z el punto de intersección.
 - b) Insertar z en la pila, $i := k$ e ir al paso 1.
4. Si v_i está a la derecha de $\overrightarrow{qv_{i-1}}$ y a la izquierda de $\overrightarrow{v_{i-2}v_{i-1}}$ (Figura 2.6a) entonces (caso 2b):
 - a) Sea u el elemento tope de la pila. Se saca de la pila.
 - b) Mientras u es un vértice y $v_{i-1}v_i$ intersecta uq , se saca de la pila (Figura 2.6a).
5. Si $v_{i-1}v_i$ no intersecta uq (Figura 2.6b) entonces:
 - a) Si v_{i+1} está a la derecha de $\overrightarrow{qv_i}$ (Figura 2.6b) entonces $i := i + 1$ y ve al paso 4b.

- b) Sea m el punto de intersección de $\overrightarrow{qv_i}$ y el lado contenido en u . Si v_{i+1} está a la derecha de $\overrightarrow{v_{i-1}v_i}$ (Figura 2.7a) entonces insertar m en la pila e ir al paso 1.
 - c) Escanear desde v_{i+1} en sentido contrario de las manecillas del reloj hasta encontrar un vértice v_k tal que $v_{k-1}v_k$ intersecte mv_i (Figura 2.7b). Asignar k a i e ir al paso 4b.
6. Sea w el vértice inmediato debajo de u en la pila. Sea p el punto de intersección entre $v_{i-1}v_i$ y uq . Si $p \in qw$ (Figura 2.8a) o q , w y u no son colineales entonces sacamos de la pila e ir al paso 4b.
 7. Escanear desde v_{i+1} en sentido contrario de las manecillas del reloj hasta encontrar un vértice v_k tal que $v_{k-1}v_k$ intersecta wp (Figura 2.8b). Inserte el punto de intersección en la pila, asignar k a i e ir al paso 1.
 8. La salida $V(q)$ es obtenida de todos los vértices y puntos en la pila y el algoritmo finaliza.

2.3. 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 [28] se presenta una variante del método llamada SRT-Star, la cual involucra una estrategia de percepción que toma completamente 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 2.9. 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 2.10). 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.

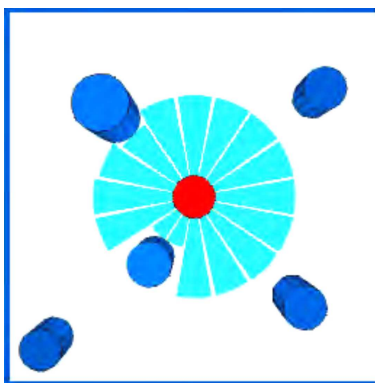


Figura 2.9: Región segura local S obtenida con la estrategia de percepción SRT-Star. Note que la extensión de S en algunos conos es reducida por el rango de alcance del sensor.

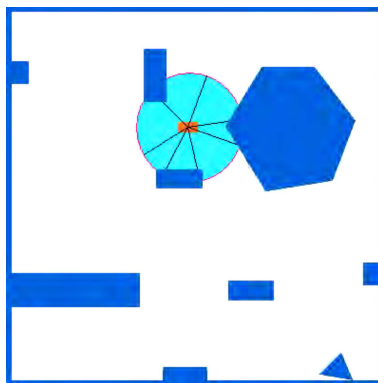
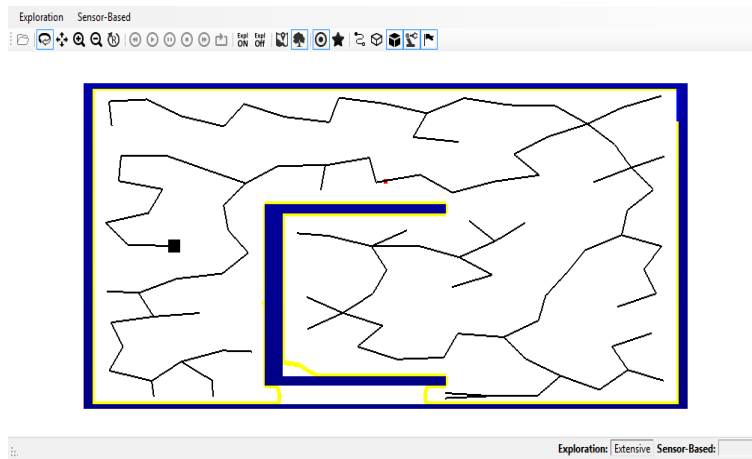
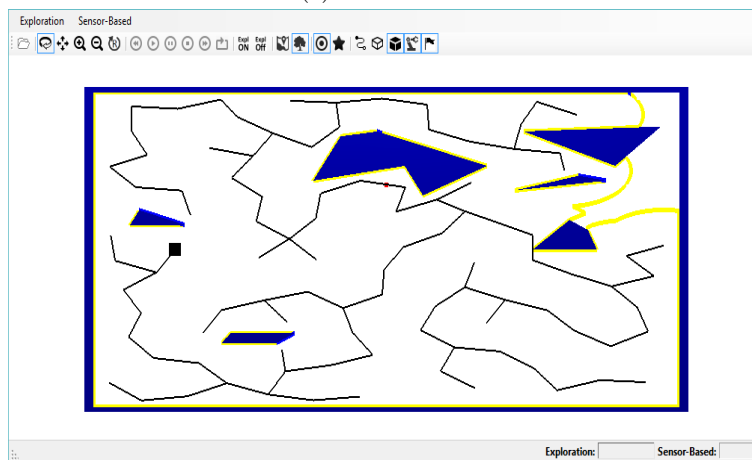


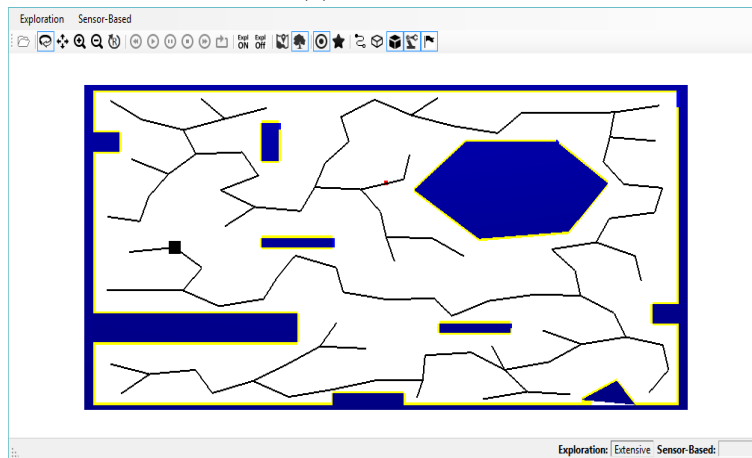
Figura 2.10: Diferentes radios obtenidos en la región segura local S con la estrategia de percepción SRT-Radial y la aplicación de la librería GPC.



(a) Ambiente 1



(b) Ambiente 2



(c) Ambiente 3

Figura 2.11: Árbol resultante utilizando el método SRT en 3 ambientes; en los escenarios, en color negro se puede ver al robot

2.4. Conclusiones

Esta estrategia construye incrementalmente un árbol aleatorio de exploración como el sensor reporta el espacio libre del ambiente. El árbol refleja la conectividad del espacio libre conocido donde el robot móvil se mueve para conocer porciones del espacio físico.

Para explorar el ambiente la estrategia utiliza una simple elección de estados candidatos para un nuevo proceso de sensado guiado con una heurística sencilla sobre la distancia euclidiana que separa al estado candidato del estado meta.

En la Figura 2.11 se muestra el resultado de la simulación de la exploración en 3 ambientes, dichos resultados posteriormente serán comparados con principal método trabajado en esta tesis, el método SET descrito en el siguiente capítulo.

Capítulo 3

Método SET

Aquí se presenta un método para la exploración basada en sensores de entornos desconocidos utilizando un sistema robótico equipado con telémetros. El método se basa en la generación incremental de una estructura de datos del espacio de configuraciones llamado Árbol de Exploración basado en sensores (SET, del inglés Sensor-based Exploration Tree). La expansión del SET es llevado por la información a nivel global, donde el proceso de percepción tiene lugar. En particular, las fronteras de la región explorada se utilizan para guiar la búsqueda de información de las configuraciones de las vistas.

En aplicaciones de servicio, los sistemas robóticos a menudo son requeridos para llevar a cabo determinadas tareas en entornos que son parcial o completamente desconocidos. El sensado, planificación y ejecución del movimiento deben ser entrelazados apropiadamente con el fin de descubrir y navegar por las porciones del medio ambiente que son relevantes para la tarea asignada. Por ejemplo, la planificación de movimientos que utiliza el sensor, aborda el problema de encontrar un camino libre de colisiones desde una configuración inicial a una configuración meta en un entorno desconocido [5] [3].

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

Existe una cantidad constante de literatura que concuerda con la exploración basada en sensores utilizando robots móviles de un solo cuerpo.

Típicamente, se asume que el robot tiene forma de disco y está equipado con un sensor láser omnidireccional. Para este problema, existen muchos algoritmos de exploración que caen en la clase de estrategias basadas en fronteras [27] - [19]. Estos se basan en la idea de que el robot debe acercarse al límite entre el área explorada e inexplorada de los ambientes a fin de maximizar la utilidad esperada de los movimientos del robot.

El problema de la exploración de un mundo desconocido con un sistema robótico con varios cuerpos, tal como un manipulador fijo o móvil, es más desafiante. Esto se debe fundamentalmente al hecho de que el espacio sentido (el mundo) y el espacio de planificación (el espacio de configuraciones) son de naturaleza muy diferente. En particular, el primero es un espacio euclidiano de dimensión 2 o 3, mientras que el segundo es un espacio con la presencia de coordenadas angulares y tiene dimensión igual al número de grados de libertad (dof's) del robot, típicamente 6 o más. Mientras que las fronteras a nivel global conservan claramente su valor informativo, no es sencillo utilizar esta información para planificar acciones en el espacio de configuraciones. En la literatura, existen algunos trabajos que abordan este problema principalmente para los manipuladores de base fija, por ejemplo, ver [18] - [38].

En este trabajo se presenta el método SET (Árbol de exploración basado en sensores) la estrategia de exploración basada en fronteras para sistemas robóticos generales, se puede ver como una extensión del método para robots móviles descrito en [19], [28]. La idea básica es guiar el robot para que el sistema sensorial realice una exploración en profundidad del mundo, sensando progresivamente regiones contiguas desde el punto de vista de la ubicación del sensor. La información recopilada acerca del espacio libre es mapeada al roadmap en el espacio de configuraciones que se construye por un procedimiento de muestreo. Este último es utilizado para seleccionar una nueva configuración de la vista, el cual es agregado al SET. En el proceso de exploración, el robot alterna movimientos hacia adelante y hacia atrás en el SET, el cual esencialmente actúa como un hilo de Ariadna (recordemos que es un método clásico de planificación de movimientos propuesto en los años 90, por J.M. Ahuactzin)[3].

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 método de muestreo global.

En el segundo, el roadmap se construye en la forma de un bosque de árboles conectados, cada uno con raíz en una configuración con un punto de vista diferente y estos crecen a través de un procedimiento "local" de muestreo.

3.1. Descripción 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 para "cubrir" tanto como sea posible del ambiente 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.

3.2. Robot y modelos del mundo

El robot es llamado R . este consiste de una cadena cinemática de r cuerpos rígidos ($r > 1$) interconectado por uniones elementales. Esta descripción incluye: manipuladores de base fija, robots móviles de un cuerpo y robots móviles multicuerpos, como vehículos con remolques, robots con forma de serpiente, humanoides y manipuladores móviles [37].

El *mundo* W es un subconjunto compacto de $\mathbb{R}^N$, 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 , $j = 1, 2, \dots, p$, cada uno de ellos es un subconjunto compacto conectado a W . El límite se supone que actúa como una "cerca" 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 , $j = 1, 2, \dots, p$. El mundo libre es $W_{free} = W \setminus O$.

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

3.3. Modelo del sensor

El robot está equipado con un sistema de m sensores de percepción exterior, cuya operación se formaliza de la siguiente manera. Se asume que el robot está en q , se denota por $F_i(q) \subseteq \mathbb{R}^N$ 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 $\mathbb{R}^2$, $F_i(q)$ puede ser un sector circular con el vértice $s_i(q) \in W$, con ángulo de apertura α_i y radio R_i , en donde este último es el rango de percepción.

El *campo del sensor* (total) en q es $F(q)$, definido como la unión de todos los $F(i)$ para $i = 1, 2, \dots, m$. Dada una configuración q del robot, un punto $p \in W$ se dice que es visible desde el sensor i -ésimo si $p \in F_i(q)$ y el segmento de línea abierta que une p y $s_i(q)$ no se intersectan $\partial O \cup \partial R(q)$.

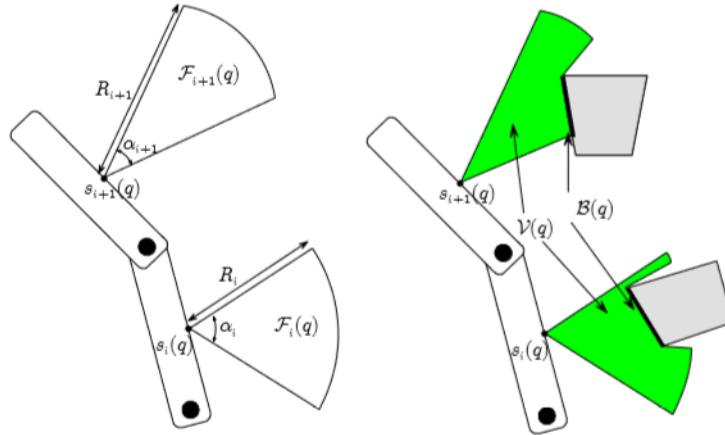


Figura 3.1: Izquierda: sensores centrales $s_i(q)$ y $s_{i+1}(q)$ y el campo de la vista asociada $F_i(q)$ y $F_{i+1}(q)$ cuando el robot esta en la configuración q . Derecha: la vista V_q y el límite visible del obstáculo $B(q)$.

En cada configuración q , el sistema sensorial del robot regresa (ver Figura 3.1):

- o La *región libre visible* (o vista) $V(q)$, el cual colecta todos los puntos de W_{free} que son visibles para por lo menos un sensor.
- o El límite del obstáculo visible $B(q) = \partial O \cup \partial V(q)$, el cual colecta todos los puntos de O que son visibles desde 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 a lo largo de 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.

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

$$\varepsilon^0 = R(q^0) \cup V(q^0) \quad (3.1)$$

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 de q^0 (proporcionados por los sensores exteroceptivos). En el paso $k \geq 1$, la región que es explorada es:

$$\varepsilon^k = V(q^k) \cup \varepsilon^{k-1} \quad (3.2)$$

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

$$\varepsilon^k = r(q^0) \subset \left(\bigcup_{i=0}^k V(q_i) \right) \quad (3.3)$$

Un punto $p \in W_{free}$ se define *explorado* en el paso k , si está contenido en ε^k e *inexplorado* si no. En cada paso k , $E^k \subseteq W_{free}$ 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 colecta 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 expandir ε^k tanto como sea posible a medida que k se incrementa.

3.5. Estrategias de exploración

En esta estrategia asumimos que el robot es capaz de asociar una información obtenida $I(q, k)$ a cualquier q (segura) en el paso k . Esto es una estimación de la información del mundo que puede ser descubierta mediante la adquisición de una vista desde q en el paso actual.

Consideremos el k -ésimo paso de la exploración, el cual inicia con el robot en q^k . Sea $\varrho^k \subset S^k$ 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 $\varrho^k \cap D(q^k, k)$, 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 $D(q^k, k)$ es un pequeño "vecindario" de q^k la búsqueda es de alcance local.

Cuando $\varrho^k \cap D(q^k, k)$ no es vacío, es decir, el conjunto admisible del paso k contiene configuraciones informativas, un movimiento seguro se realiza hacia la nueva configuración de la vista seleccionada (hacia adelante). De lo contrario, el robot vuelve de nuevo hacia una configuración de la vista previamente visitada (hacia atrás).

La exploración puede ser considerada completa en el paso k si $\varrho^k = \emptyset$, es decir, no hay ninguna configuración $q \in S^k$ tal que (1) una nueva visión de q puede ser extendida a la región explorada actual (2) el robot puede alcanzar q desde q^k a través de un camino que es seguro en el paso k .

Para 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 para definir un procedimiento eficiente para calcular $\varrho^k \cap D(q^k, k)$.

3.6. Límite libre

Una herramienta útil, el cual provee información acerca de ϱ^k sin requerir un cálculo explícito, es el límite libre (también llamado frontera en la literatura [37] - [12]) de la región explorada. En el paso k , el límite de la región explorada $\partial\varepsilon^k$ es la unión de dos conjuntos disjuntos:

- o El límite obstáculo $\partial\varepsilon_{obs}^k$ es decir, la parte $\partial\varepsilon$ se realiza mediante la detección de las superficies de los obstáculos.
- o La frontera libre $\partial\varepsilon_{free}^k$ es decir, el complemento de $\partial\varepsilon_{obs}^k$ que conduce a áreas potencialmente explorables.

El límite obstáculo $\partial\varepsilon_{obs}^k$ puede ser calculado como la unión de todos los límites obstáculos visibles $B(q^i)$, $i = 0, 1, \dots, k$, reunidos por el robot hasta el k -ésimo paso. La frontera libre $\partial\varepsilon_{free}^k$ es entonces calculado como $\partial\varepsilon^k \setminus \partial\varepsilon_{obs}^k$.

La frontera libre tiene algunas propiedades útiles. Dada una configuración segura q , se tienen las siguientes implicaciones:

$$F(q) \cap \partial\varepsilon_{free}^k = \emptyset \Rightarrow I(q, k) = 0 \quad (3.4)$$

Es decir, si no se presenta frontera libre en todo el campo de q , entonces nada puede ser descubierto mediante la adquisición de una vista desde q . Además uno tiene

$$\partial\varepsilon_{free}^k = \emptyset \Rightarrow \varrho^k = \emptyset \quad (3.5)$$

De hecho, si la frontera libre está vacía, no hay más puntos inexplorados que permanezcan a W_{free} y la exploración es completada. De lo contrario de (2) y (3) en general no son ciertas. A la luz de las implicaciones anteriores, y gracias al bajo coste computacional de la frontera, la idea central del método SET es guiar al robot hacia aquellas configuraciones $q \in D(q^k, k)$ en la que $F(q) \cap \partial\varepsilon_{free}^k \neq \emptyset$.

3.7. El método SET

En el método SET, el robot construye incrementalmente la estructura de datos "Árbol de exploración basado en sensores (SET)". Cada nodo del SET representa una configuración de la vista, mientras que un arco entre dos nodos representa un camino seguro uniendo dos configuraciones de la vista. Un ciclo de exploración del algoritmo SET se muestra en la Figura 3.2. 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 configuración de la vista ha sido rechazada en la 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 se agrega al modelo del ambiente $(\varepsilon^k, \partial\varepsilon_{obs}^k)$.

Después, se extrae el *límite local libre* LFB (q^k, k) (línea 2). En términos generales, contiene alguna porción de frontera libre que es visible por el sensor desde alguna $q \in D(q^k, k)$. Se define de manera que se puede calcular fácilmente y tiene la siguiente implicación:

$$LFB(q^k, k) = \emptyset \Rightarrow D(q^k, k) \cap \varrho^k = \emptyset \quad (3.6)$$

Esto significa que $LFB(q^k, k) \neq \emptyset$ es una condición necesaria que debe ser verificada antes de la búsqueda de una configuración de la vista en $D(q^k, k) \cap \varrho^k = \emptyset$ (línea 4); si $LFB(q^k, k) = \emptyset$, se obliga un retroceso. La estrategia de selección adoptada (se detalla más adelante) consiste en la maximización de 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, se proponen al final de esta parte. De acuerdo a (4), cuando el límite local libre es vacío, no se lleva a cabo la búsqueda y $U(q^{k+1})$ es puesto en 0 forzando un 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). Esta comprobación es doble: 1) fuerza un paso de retroceso cuando el límite local libre es vacío (es decir, cuando $U(q^{k+1})$ es un conjunto vacío); 2) filtros casi inútiles para las acciones del robot.

Método SET

```

1  actualizar SET y el modelo del medio ambiente
2  se extrae la frontera local libre LFB ( $q^k, k$ )
3  si la frontera local libre es no vacía
4      ( $q^{k+1}, U(q^{k+1})$ )  $\leftarrow$  Buscar configuración con utilidad máxima admisible
        en el interior del conjunto actual  $D(q^k, k)$ 
5  si no
6       $U(q^{k+1}) \leftarrow 0$ 
7  si  $U(q^{k+1}) \geq U_{min}$     % verifica utilidad minima
8      planifica un camino seguro conectando  $q^k$  a  $q^{k+1}$ 
9      se mueve a  $q^{k+1}$  y obtiene la vista del sensor
10 si no
11     se mueve a la configuración padre

```

Figura 3.2: Descripción en pseudocódigo de la iteración k -ésima del método SET en el cual el robot inicia en q^k .

Por lo tanto, si $U(q^{k+1}) > U_{min}$: q^{k+1} se convierte en la siguiente configuración de la vista, el camino planificado es invocado para calcular el camino de q^k a q^{k+1} (línea 8), q^{k+1} es alcanzado y una nueva vista se obtiene (línea 9). De otra manera, el paso de retroceso es realizado y el robot se mueve a la configuración padre (línea 11).

Cuando el robot es incapaz de realizar "un avance hacia adelante" hacia los límites 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).

3.7.1. Ganancia de la información

Sea $V(q, k)$, la *vista simulada*, la cual sería adquirida por el robot en q si el límite del obstáculo fuera exactamente $\partial\epsilon_{obs}^k$. Ya que nuestro objetivo es extender ϵ^k tanto como sea posible, nosotros definimos la ganancia de información $I(q, k)$ como la medida del conjunto de todos los puntos no explorados contenidos en $V(q, k)$. $I(q, k)$ puede ser calculado por el robot a través de un proceso de rastreo.

Claramente, dependiendo de la representación del ambiente y la tarea del robot, son posibles otras definiciones de ganancia de información [13]. Por ejemplo, en la presencia de ruido en el sensado se puede adoptar una representación probabilística del ambiente con sus respectivas definiciones del límite libre basadas en entropía.

3.7.2. Conjunto admisible y definición del límite local libre

Ya que la implicación (4) se requiere para respaldar las definiciones de conjunto admisible y límite local libre, estas deben estar estrictamente relacionadas. En esta sub-sección, primero definiremos $D(q^k, k)$, entonces $LFB(q^k, k)$ es designada de acuerdo 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 como 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 $s(q)$ está dentro de una distancia máxima ρ desde $S(q^k)$, (ii) $s(q)$ y $s(q^k)$, son mutuamente visibles en el paso k , es decir el segmento de línea abierta $(s(q), s(q^k))$, no se intersecta con $\partial\epsilon_{obs}^k$.

El primer requerimiento, según la estructura algorítmica de adelante/atras ó impuesta, sugiere una "Estrategia de búsqueda basada primero en profundidad". Esto es, el centro del sensor es guiado realizando un recorrido primero en profundidad del ambiente, explorando incrementalmente regiones contiguas. La segunda condición es un usual requerimiento en la exploración de ambientes desconocidos. Esto previene a la exploración de ambientes de saltos continuos entre cuartos continuos separados (por ejemplo: cuando $s(q^k)$ y $s(q^{k+1})$ son cercanas pero separadas por una pared).

Por otra parte, una colección de sensados mutuamente visibles localizados en el mundo puede verse como un grafo visible de localidades reconocibles conocidas con especial interés y con ciertas propiedades topológicas.

- 2) *Límite local libre*: $LFB(q^k)$ colecta todos los puntos del límite libre $\partial\epsilon_{free}^k$ los cuales (i) están contenidos en una bola $B(s(q^k), \rho + R)$ con centro en $s(q)$ y un radio $\rho + R$ (ii) puede ser conectado a $s(q^k)$ a través de camino en el ambiente completamente contenido en $\epsilon^k \cap B(s(q^k), \rho + R)$. 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 $LFB(q^k)$ y $D(q^k, k)$ dado anteriormente.

3.7.3. Estrategia de selección

El enfoque más común al evaluar la contribución de la acción de un robot hacia el cumplimiento de una tarea asignada a este, es la asociación de una función de utilidad. Una maximización se usa a menudo para seleccionar la siguiente acción, posiblemente sobre un número finito de elecciones. Los diferentes requerimientos en la tarea pueden ser trasladados 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 en el camino total explorado, pero la complejidad del problema, junto con la falta de una información prioritaria, sugiere un enfoque más efectivo e inteligente, donde la evaluación de la utilidad de una acción este basada en un simple paso hacia adelante. Sin embargo, en principio, las decisiones más inteligentes se vuelven posibles si se obtiene mas información 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 desde q^k a q^{k+1} puede ser incluida en U para reducir en lo posible los comportamientos erróneos. Así que, como muestra a continuación, la definición propuesta de $D(q^k, k)$ y una estrategia de búsqueda adecuada naturalmente limita estos comportamientos erróneos.

3.7.4. Estrategias de búsqueda

Durante la exploración, el modelo del espacio de configuración es actualizado incrementalmente para (i) la búsqueda de nuevas configuraciones de la vista y (ii) la realización de operaciones de planificación. Desde manipuladores que típicamente tienen espacios de configuración de alta dimensión, esto es conveniente en un enfoque de crecimiento incremental de un roadmap, el cual captura la conectividad de la región segura actual.

En particular, sea ϱ^k el roadmap construido en el paso k en la región segura S^k . En ϱ^k , un nodo representa una configuración que es segura en el paso k , mientras un arco entre dos nodos representa un camino local que es seguro en el paso k y que conecta las dos configuraciones.

Una vez que ε^k se calcula fusionando $V(q^k)$ con ε^{k-1} , el roadmap ϱ^k se obtiene expandiendo ϱ^{k-1} . Con el fin de encontrar estas configuraciones se realiza un chequeo de colisión en la reconstrucción del modelo del mundo en el paso k : conforme a este modelo, ε^k es el ambiente libre disponible y $\partial\varepsilon^k$ es el límite del obstáculo. Podemos notar que, en este marco teórico, el SET construido en el paso k representa el camino actual recorrido por el robot en el roadmap ϱ^k .

Dos instancias principales del método SET pueden obtenerse 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 árboles, cada uno crece localmente alrededor de una configuración de la vista almacenada.

1) SET-GG: en esta estrategia el roadmap ϱ^k es expandido globalmente usando un enfoque basado en muestreo tal que (i) se trata de un algoritmo PRM multi-consulta o (ii) un algoritmo de árbol sencillo simple-consulta (RRT o EST). Una descripción en pseudocódigo de la estrategia es mostrada en el algoritmo de la Figura 3.3. En la primera, el roadmap ϱ^{k-1} es expandido globalmente para obtener ϱ^k (línea 1). Aquí, una de las técnicas basadas en muestreo antes mencionadas es usada. Entonces, el subconjunto $\hat{D}$ de configuraciones cae dentro del conjunto admisible actual $D(q^k, k)$ es extraído de ϱ^k (línea 2). Note que $\hat{D}$ representa un estimado de $D(q^k, k)$. En este punto, q^{k+1} es encontrado como una configuración de 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 ϱ^k es construido en la forma de un bosque de árboles conectados. Cada uno de estos árboles está enraizado en una distinta vista de configuración. Por lo tanto, en este caso, un nodo del SET representa una configuración de vista junto con el árbol enraizado a 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, esto permite una búsqueda global (realizando ocasionalmente saltos largos).

Una descripción del pseudocódigo de la estrategia SET-LG se muestra en el algoritmo de la Figura 3.4. Primero, un árbol T^k enraizado en la configuración actual a q^k es expandido (línea 1). Su expansión se lleva

a cabo dentro de un espacio- C en forma de bola con centro en q^k y radio δ . Se puede usar en este caso un algoritmo de simple consulta tal como RRT. Después, el subconjunto $\hat{D}$ de configuraciones caídas dentro del conjunto admisible actual $D(q^k, k)$ es extraído de T^k (línea 2). En este punto, una vista de configuración candidata q^{k+1} es encontrada como una configuración de $\hat{D}$ 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 costo de navegación del siguiente movimiento del robot y evita la definición problemática de un función de utilidad mezclada (donde se debe considerar una penalización explicita en la distancia recorrida para evitar un comportamiento problemático).

Búsqueda con crecimiento global

- 1 $\varrho^k \leftarrow$ expande globalmente el roadmap ϱ^{k-1}
 - 2 $\hat{D} \leftarrow$ extrae de ϱ^k el subconjunto de configuraciones que caen dentro del conjunto admisible actual $D(q^k, k)$
 - 3 $(q^{k+1}, U(q^{k+1})) \leftarrow$ encuentra configuración en $\hat{D}$ con utilidad máxima
-

Figura 3.3: Descripción en pseudocódigo de la estrategia de búsqueda con crecimiento global (GG)

Búsqueda con crecimiento local

- 1 $T^k \leftarrow$ expande el árbol enraizado a q^k con una bola C -espacio con centro q^k y radio δ
 - 2 $\hat{D} \leftarrow$ extrae de T^k el subconjunto de configuraciones que caen dentro del conjunto admisible actual $D(q^k, k)$
 - 3 $(q^{k+1}, U(q^{k+1})) \leftarrow$ encuentra configuración en $\hat{D}$ con utilidad máxima
 - 4 **si** $U(q^{k+1}) < U_{min}$
 - 5 $\zeta^k \leftarrow$ se expande el árbol Lazy enraizado a q^k dentro de $D(q^k, k)$
 - 6 $\hat{D} \leftarrow$ extrae de ζ^k un subconjunto de configuraciones las cuales son seguras en el paso k y tiene una utilidad $U \geq U_{min}$
 - 5 $(q^{k+1}, U(q^{k+1})) \leftarrow$ encuentra una configuración en $\hat{D}$ alcanzable desde q^k
-

Figura 3.4: Descripción en pseudocódigo de la estrategia de búsqueda con crecimiento local (LG)

Después, si $U(q^{k+1}) \geq U_{min}$, q^{k+1} se convierte en la nueva configuración de la vista y retorna la rutina de búsqueda. De otra manera, se efectúa la búsqueda global en el conjunto $D(q^k, k)$. Eso se realiza en tres pasos (línea 5-7). Primero (línea 5), un árbol ζ^k enraizado en q^k es expandido en el conjunto admisible $D(q^k, k)$: durante la restringida expansión se realiza el chequeo de no colisión (expansión Lazy). Aquí, los RRT's se prefieren por su velocidad de expansión C-space. Después (línea 6), un subconjunto $\hat{D}$ de configuraciones las cuales son seguras en el paso k y tienen una utilidad $U \geq U_{min}$ son extraídos de ζ^k . Entonces (línea 7), se invoca un planificador de simple consulta para encontrar una configuración de $\hat{D}$ y la cual es alcanzable a través de un camino que es seguro en el paso k . Si este planificador falla al encontrar configuraciones alcanzables, entonces este 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 a las configuraciones de las vistas pueden expandirse en sobre posición de las regiones del C-space. Este resultado no aceptado, puede ser casi evitado mediante la selección adecuada del radio δ de la bola limitante del C-espacio.

3.7.5. Planificación del camino

Una vez que se ha seleccionado una nueva configuración de la vista q^{k+1} , la planificación del camino debe calcular 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 $\mathcal{Q}^k$. En SET-LG, dos casos son posibles: 1) q^{k+1} pertenece al árbol T^k y 2) q^{k+1} pertenece al árbol Lazy ζ^k . En el primer caso, un camino seguro puede ser encontrado fácilmente en T^k . En el segundo caso la estrategia de búsqueda (algoritmo 5, línea 7) retorna automáticamente un camino que es seguro en el paso k . Por simplicidad, este no ha sido detallado de forma implícita en el pseudocódigo.

Capítulo 4

Resultados obtenidos

Una vez descrita la parte teórica en capítulos anteriores, en este capítulo se reportan los resultados obtenidos de la comparación de las estrategias SRT y SET con las mismas condiciones en un ambiente 2D.

Dicha implementación se ha desarrollado en C# (Microsoft Visual Studio 2010), aprovechando la estructura de la librería MSL (Motion Strategy Library) descrita en el Anexo A; así como un ambiente gráfico en OpenGL para la descripción de la escena, al igual que una interfaz gráfica para visualizar el ambiente de trabajo y el resultado de la exploración.

4.1. Resultados en ambientes 2D

Las simulaciones se realizaron de la siguiente manera: en los escenarios mostrados en la Figura 4.1, la simulación se realizó 10 veces tanto para el algoritmo de exploración SRT (árbol aleatorio basado en sensores), como para la exploración SET (árbol de exploración basado en sensores).

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

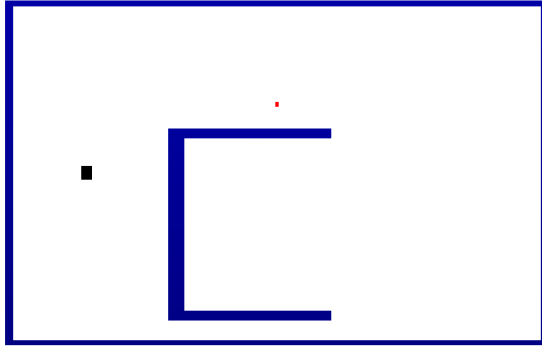
A continuación, en el Cuadro 4.1, 4.2 y 4.3 se presentan los resultados de las simulaciones realizadas, dichos resultados contienen 10 mediciones tanto del tiempo de ejecución como del número de nodos creados para la exploración, así como el promedio de dichas características medidas, además uno de los mapas resultantes del camino recorrido con el método SRT mostrados en las Figuras 4.2, 4.4 y 4.6, y con el método SET mostrados en las Figuras 4.3, 4.5 y 4.7 de cada ambiente utilizado en la simulación.

SRT		SET	
No. de Nodos	Tiempo de exploración (seg)	No. de Nodos	Tiempo de exploración (seg)
214	46.771	152	18.362
234	50.822	154	23.722
240	40.225	164	20.9722
218	48.795	146	15.9883
224	48.497	162	21.5732
226	47.955	154	20.831
214	41.967	148	16.938
210	45.403	142	16.373
220	46.862	154	20.169
218	50.403	154	17.465
promedio:	221.8 46.77	153	19.239

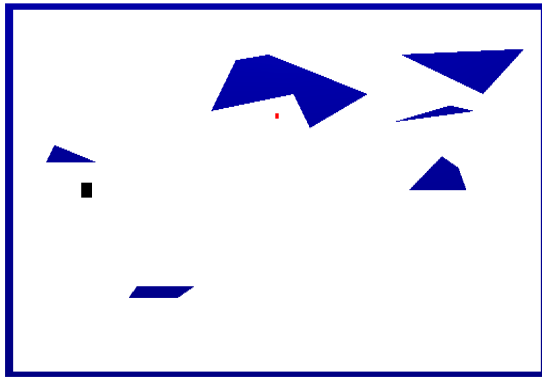
Cuadro 4.1: Resultados para el ambiente 1

SRT		SET	
No. de Nodos	Tiempo de exploración (seg)	No. de Nodos	Tiempo de exploración (seg)
210	45.444	154	20.128
222	46.329	124	15.006
218	46.819	156	18.406
214	46.656	152	17.399
224	45.624	158	18.186
220	47.297	148	16.804
210	43.89	146	17.695
212	43.236	156	17.534
208	38.631	158	18.339
210	39.251	152	20.425
promedio:	214.8 44.317	150.4	17.992

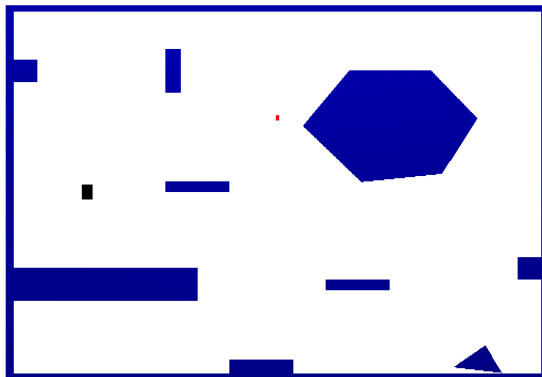
Cuadro 4.2: Resultados para el ambiente 2



(a) Ambiente 1

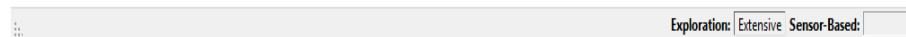
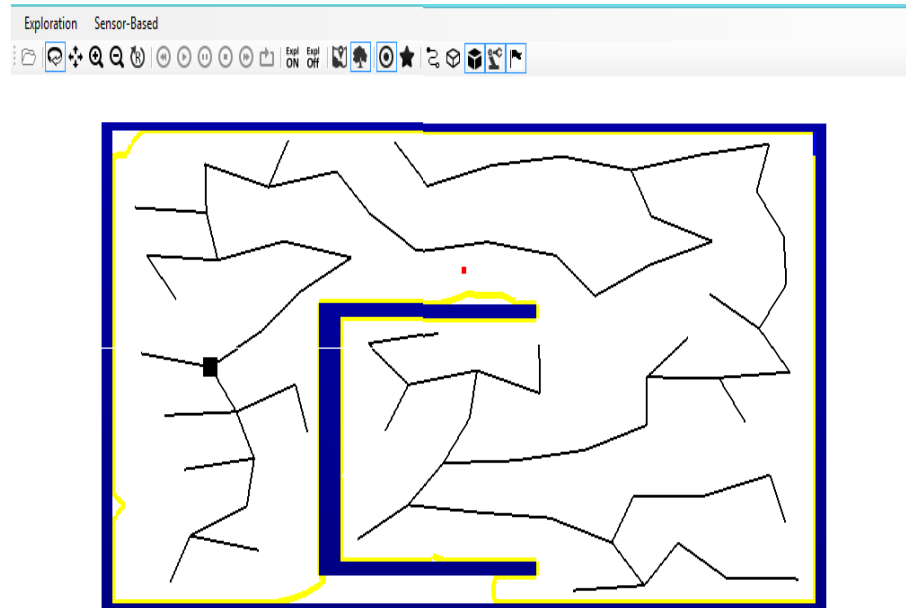


(b) Ambiente 2

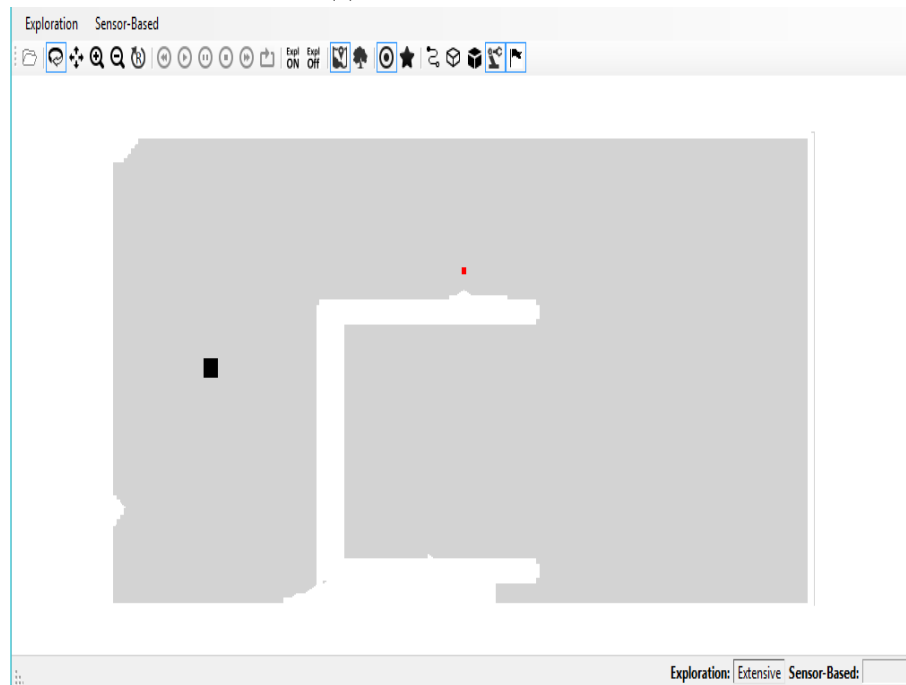


(c) Ambiente 3

Figura 4.1: Escenarios usados para las simulaciones; en los escenarios, en color negro se puede ver al robot

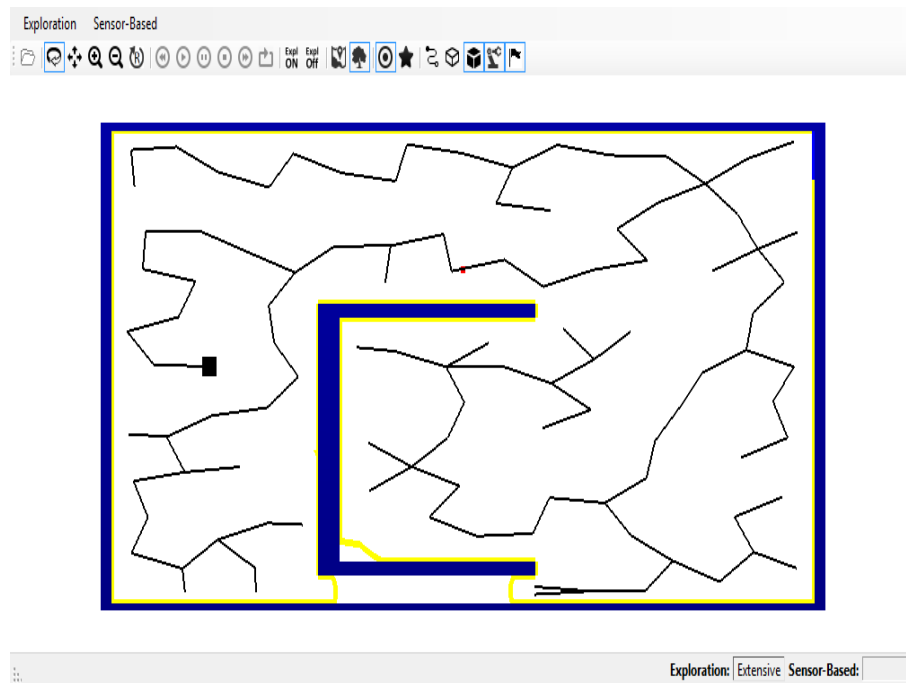


(a) Árbol de exploración

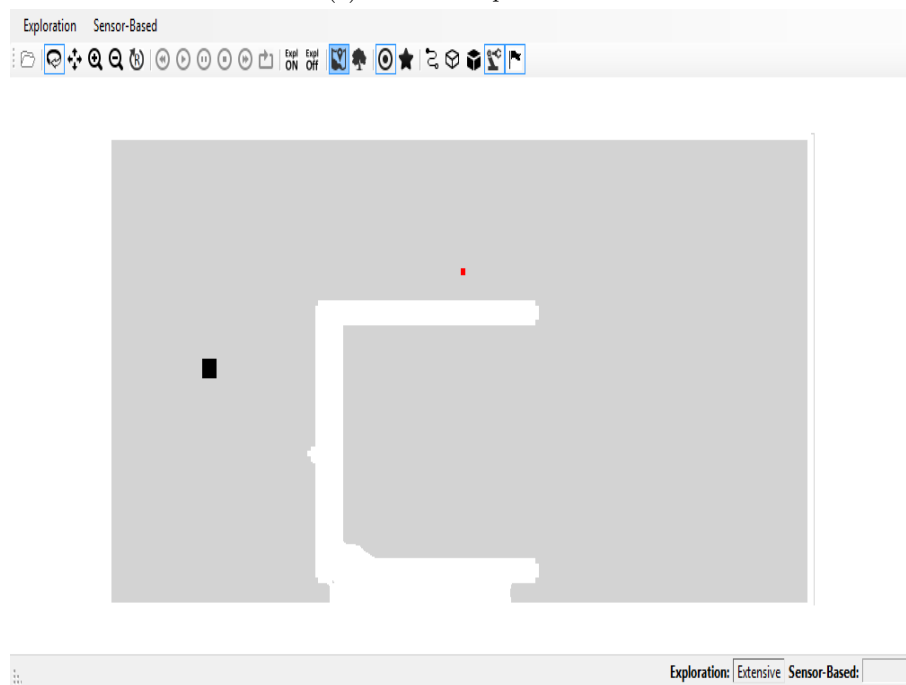


(b) Mapa

Figura 4.2: Las figuras muestran los resultados obtenidos del ambiente 1 con el método SET.

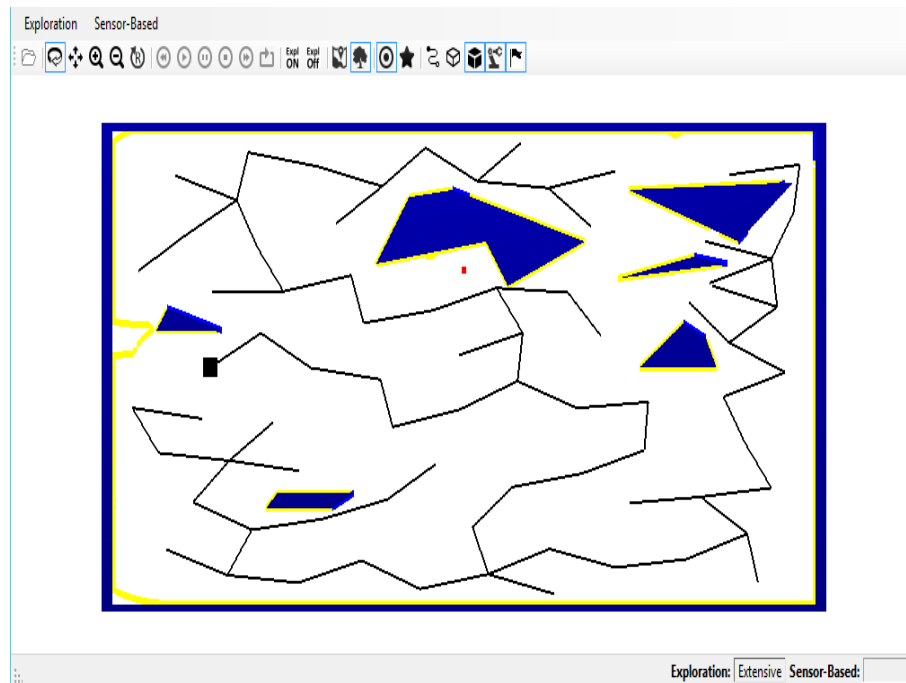


(a) Árbol de exploración

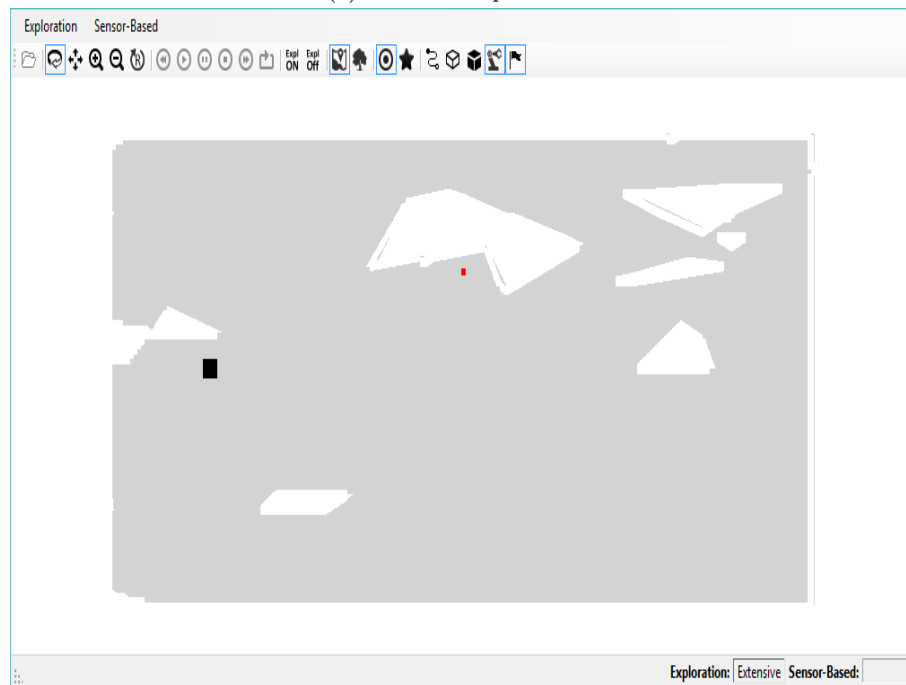


(b) Mapa

Figura 4.3: Las figuras muestran los resultados obtenidos del ambiente 1 con el método SRT.

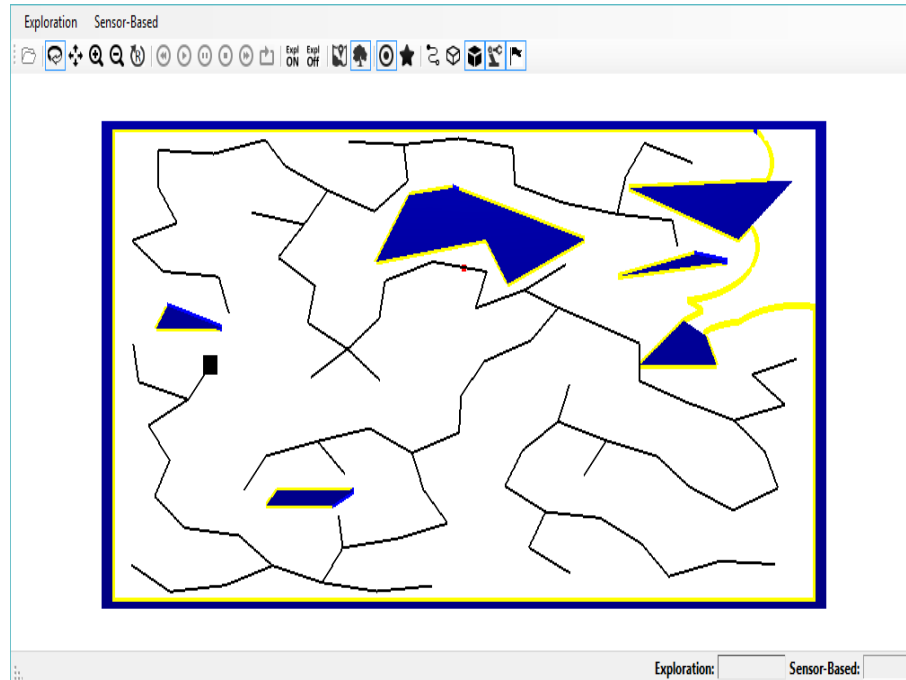


(a) Árbol de exploración

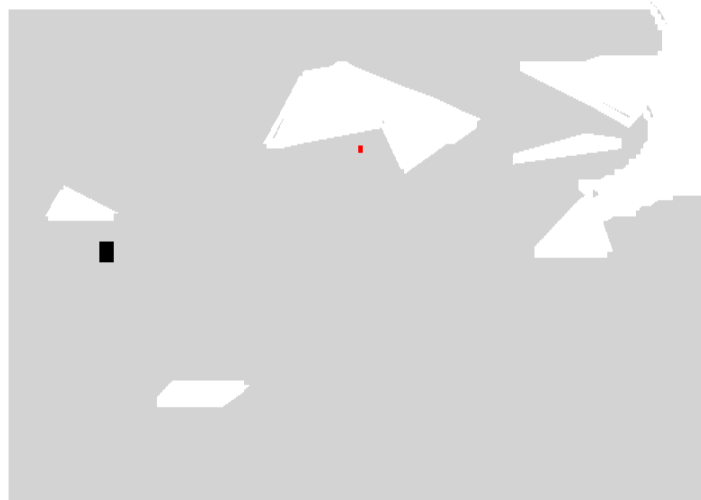


(b) Mapa

Figura 4.4: Las figuras muestran los resultados obtenidos del ambiente 2 con el método SET.

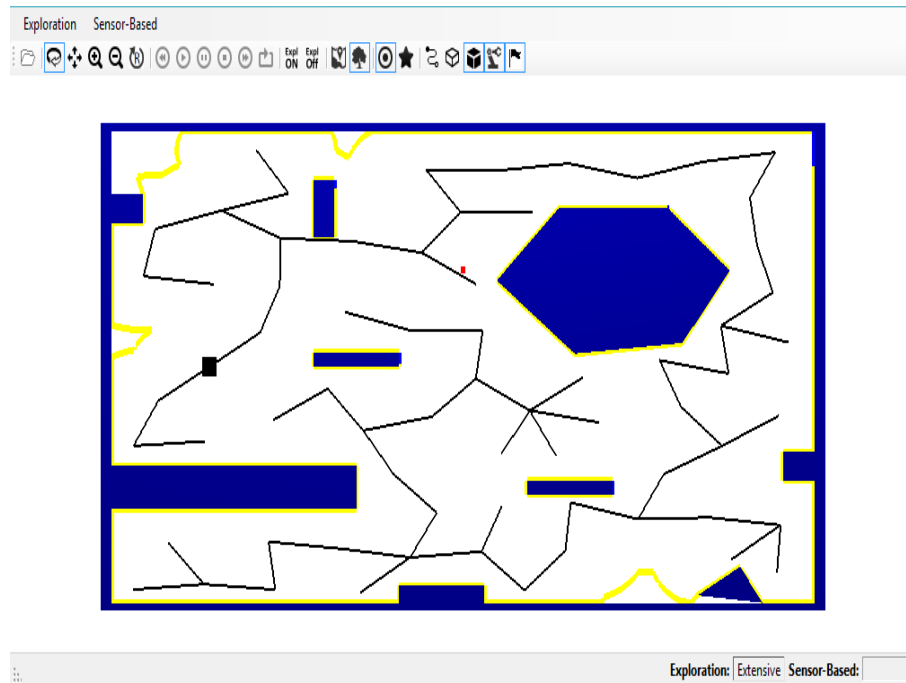


(a) Árbol de exploración

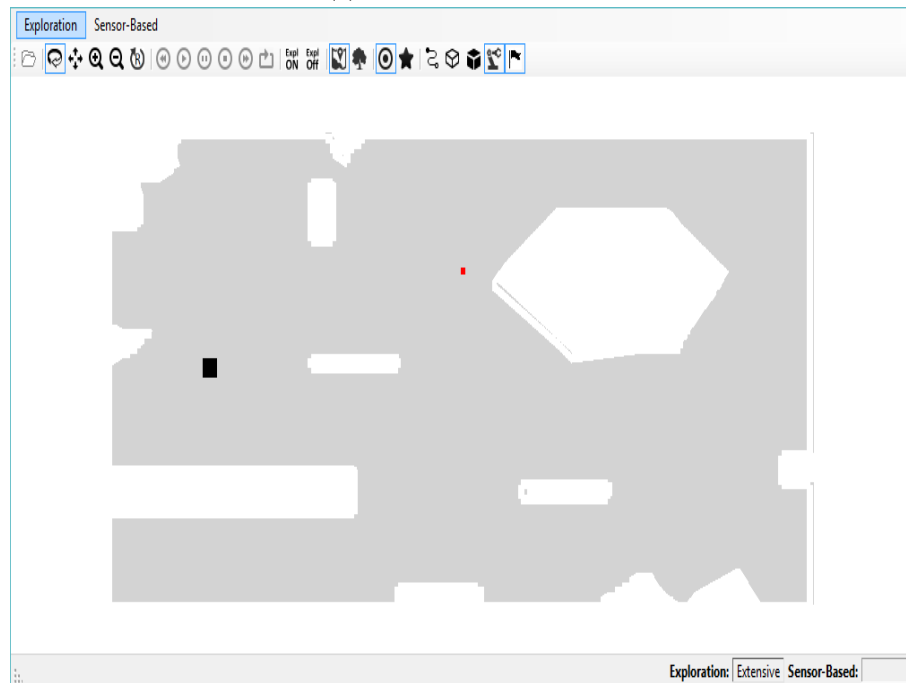


(b) Mapa

Figura 4.5: Las figuras muestran los resultados obtenidos del ambiente 2 con el método SRT.

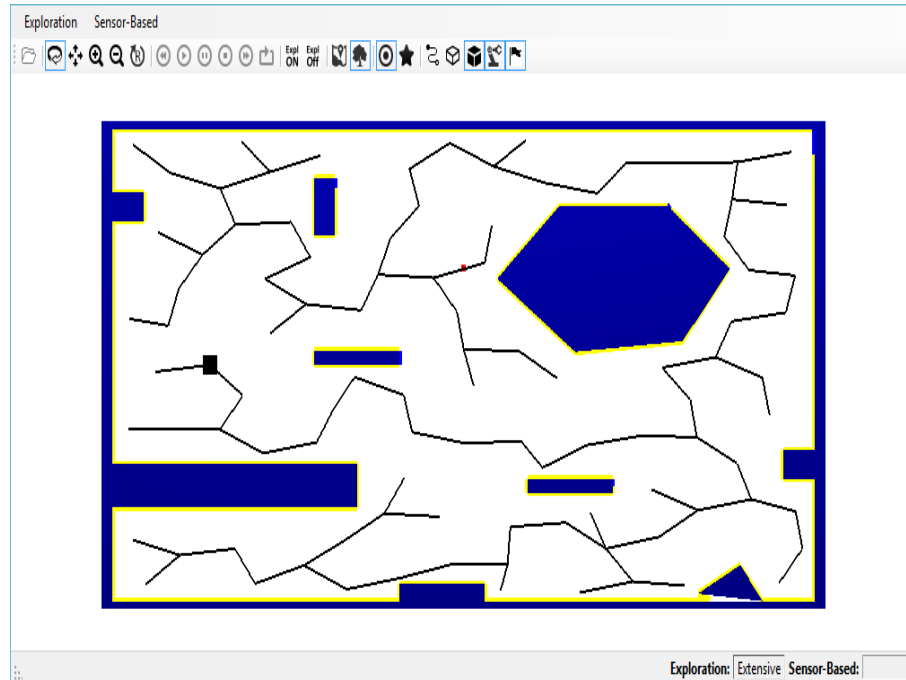


(a) Árbol de exploración

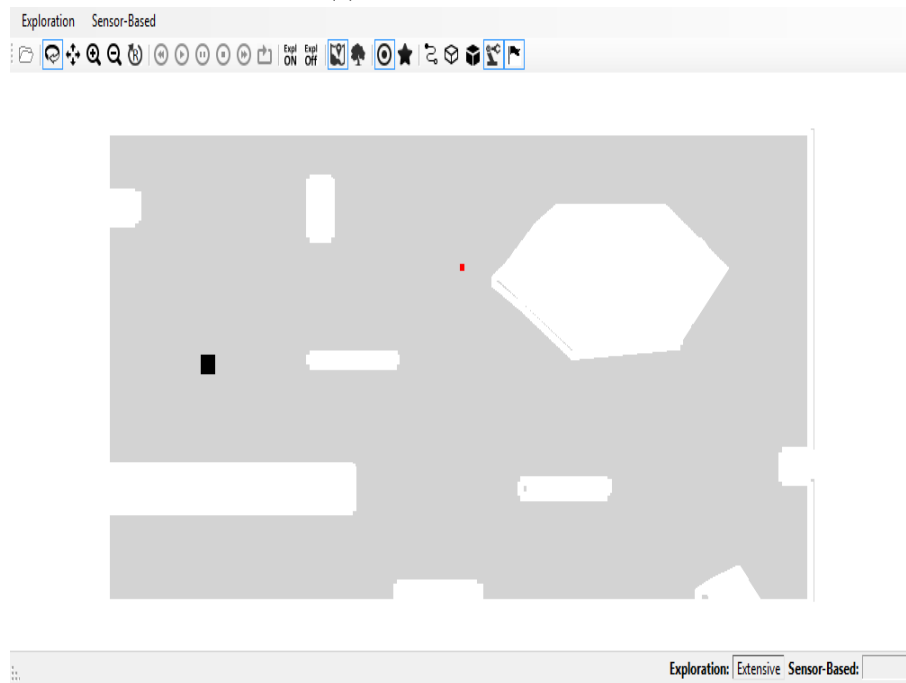


(b) Mapa

Figura 4.6: Las figuras muestran los resultados obtenidos del ambiente 3 con el método SET.



(a) Árbol de exploración



(b) Mapa

Figura 4.7: Las figuras muestran los resultados obtenidos del ambiente 3 con el método SRT.

SRT		SET	
No. de Nodos	Tiempo de exploración (seg)	No. de Nodos	Tiempo de exploración (seg)
198	42.033	136	15.296
210	46.847	140	17.405
198	39.082	144	17.159
192	34.896	140	16.471
204	38.596	152	16.332
210	40.876	138	17.736
214	45.003	160	19.542
188	42.391	130	15.547
196	37.679	148	18.671
214	47.85	138	14.127
promedio:	202.4 41.52	142.6 16.828	

Cuadro 4.3: Resultados para el ambiente 3

4.2. Discusión

Como puede observarse en las Tablas 4.1, 4.2 y 4.3 (que reflejan sólo los resultados obtenidos de las simulaciones presentadas anteriormente), se puede encontrar un comportamiento similar en todas las tablas. En primer lugar, el rendimiento de la exploración con el método SRT es menor al de SET; dicho de otro modo, el número de nodos usados y el tiempo empleado de estos es relativamente superior en SRT.

Lo anterior se debe a que el rendimiento se ve afectado por la forma en que se elige el siguiente punto a explorar, mientras que el método SET para seleccionar un estado candidato hace una seleccion aleatoria dentro de la región segura local, generando una mayor cantidad de nodos al tratar encontrar una dirección para seguir la exploración antes de regresar a un nodo anterior, mientras que el método SET genera un roadmap más simple debido a que toma en cuenta solo el límite del area local de la ultima posición y no toda el área local, así que para regresar a un nodo anterior realiza una búsqueda más eficiente lo cual refleja en una menor cantidad de nodos.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Se presentó un método basado en sensores para la exploración de ambientes desconocidos, el método está basado en la generación incremental de una estructura llamada árbol de exploración basada en sensores (SET), la generación de la siguiente acción está dada por la información del ambiente, donde el proceso de percepción tiene lugar al igual que el método SRT. La diferencia particular es la selección del siguiente estado donde el método SET utiliza la frontera de la región explorada para guiar la búsqueda y como se observó en el capítulo 4 es más eficiente.

Además se realizó la implementación de un algoritmo de visibilidad, el cual permite la corrección de la información obtenida por el sensor telemétrico y que nos ayuda a tener una mejor aproximación con el mundo real, y por lo tanto también del resultado de la exploración.

Los resultados obtenidos a través de simulaciones muestran que esta estrategia resuelve 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 varía en tiempo y en la complejidad del mapa obtenido.

5.2. Trabajo futuro

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

- Integrar un método de localización a la simulación de la aplicación.
- Una vez teniendo el punto anterior, sería ideal probar la estrategia propuesta con robots reales.
- Probar otras estrategias para la determinación del estado candidato a un nuevo proceso de percepción.

Apéndice A

Librería GPC

Tradicionalmente el recorte de polígonos se ha usado para recortar las porciones de un polígono que se encuentran fuera de la ventana del dispositivo de salida (por ejemplo la pantalla) para prevenir efectos indeseables. El recorte de un polígono arbitrario contra otro polígono arbitrario ha sido una tarea compleja. Las soluciones existentes son limitadas a cierto tipo de polígonos o tienden a ser muy complejas y consumir demasiado tiempo de ejecución.

En la librería GPC (General Polygon Clipping) se implementa un nuevo algoritmo de recorte de polígonos. Las técnicas usadas se basan en el método de recorte de polígonos de Bala R. Vatti. Tanto el polígono o conjunto de polígonos a recortar (polígono sujeto) como el polígono utilizado para el corte, (polígono de corte), pueden ser convexos o cóncavos, interceptarse a sí mismos, contener huecos, o estar comprendidos en varios contornos disjuntos.

La librería extiende el algoritmo de Vatti para permitir lados horizontales y manejar de manera robusta la coincidencia de lados en los polígonos. Las operaciones que se pueden realizar con dicha librería son: intersección, or-exclusivo, unión y diferencia del polígono sujeto con el polígono de recorte. La salida puede ser el contorno de otro polígono o una lista de triángulos (tristrip).

A.1. Descripción

Un polígono genérico (o un 'conjunto polígono') consiste de cero o más límites disjuntos de una composición arbitraria. Cada límite se le denomina "contorno", y puede ser como ya se menciono, convexo, cóncavo o interceptarse a sí mismo. Los huecos internos pueden formarse debido a los contornos. Ver la Figura A.1, donde se muestra un conjunto polígono constituido por cuatro contornos. A la izquierda tenemos un contorno cóncavo de siete lados que contiene otro contorno de cuatro lados, el cual forma un hueco en la figura envolvente. Un tercer contorno triangular, intercepta el límite del primero. Finalmente a la derecha hay un contorno disjunto que se intercepta a sí mismo de cuatro lados.

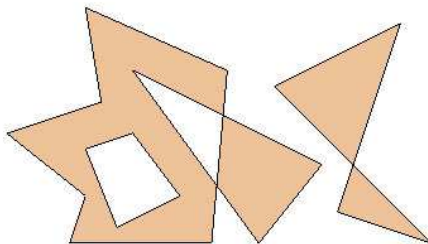


Figura A.1: Polígono genérico con cuatro contornos.

La librería soporta cuatro tipos de operaciones de recorte: la diferencia, intersección, or-xclusivo o unión de dos polígonos genéricos. La Figura A.2 muestra los tipos de operación, en cada caso el polígono resultante es resaltado con un color de relleno.

A.2. Funciones

La librería proporciona ocho funciones. Dos de ellas están dedicadas a la lectura y escritura de datos entre los archivos de los polígonos y las estructuras propias de la librería.

```
void gpc_read_polygon(FILE *fp,  
int read_hole_flags,  
gpc_polygon *polygon);
```

```
void gpc_write_polygon(FILE fp,
```

```
int write_hole_flags,  
gpc_polygon *polygon);
```

El manejo de polígonos es por medio de un archivo ASCII con el siguiente formato:

```
(número - contornos)  
(número - vértices - primer - contorno)  
[(bandera - huecos - primer - contorno)]  
(lista - vértices - primer - contorno)  
(número - vértices - segundo - contorno)  
[(bandera - huecos - segundo - contorno)]  
(lista - vértices - segundo - contorno)  
.  
.  
.
```

La bandera de huecos es opcional, las operaciones de recorte establecerán correctamente las banderas de huecos en el polígono resultante.

La función `gpc_add_contour()` facilita la fusión de un nuevo contorno con un polígono existente. El parámetro final indica si el contorno sería considerado como hueco o no, los valores que utiliza son 1 (true) o 0 (falso) según corresponda.

```
void gpc_add_contour(gpc_polygon *p,gpc_vertex_list *c, int hole);
```

La siguiente función desarrolla las operaciones de recorte,

```
void gpc_polygon_clip(gpc_op operation,  
gpc_polygon *subject_p,  
gpc_polygon *clip_p,  
gpc_polygon *result_p);
```

Si el resultado se requiere en un trisrip se utiliza la siguiente función,

```
void gpc_trisrip_clip(gpc_op operation,  
gpc_polygon *subject_p,  
gpc_polygon *clip_p,  
gpc_trisrip *result_t);
```

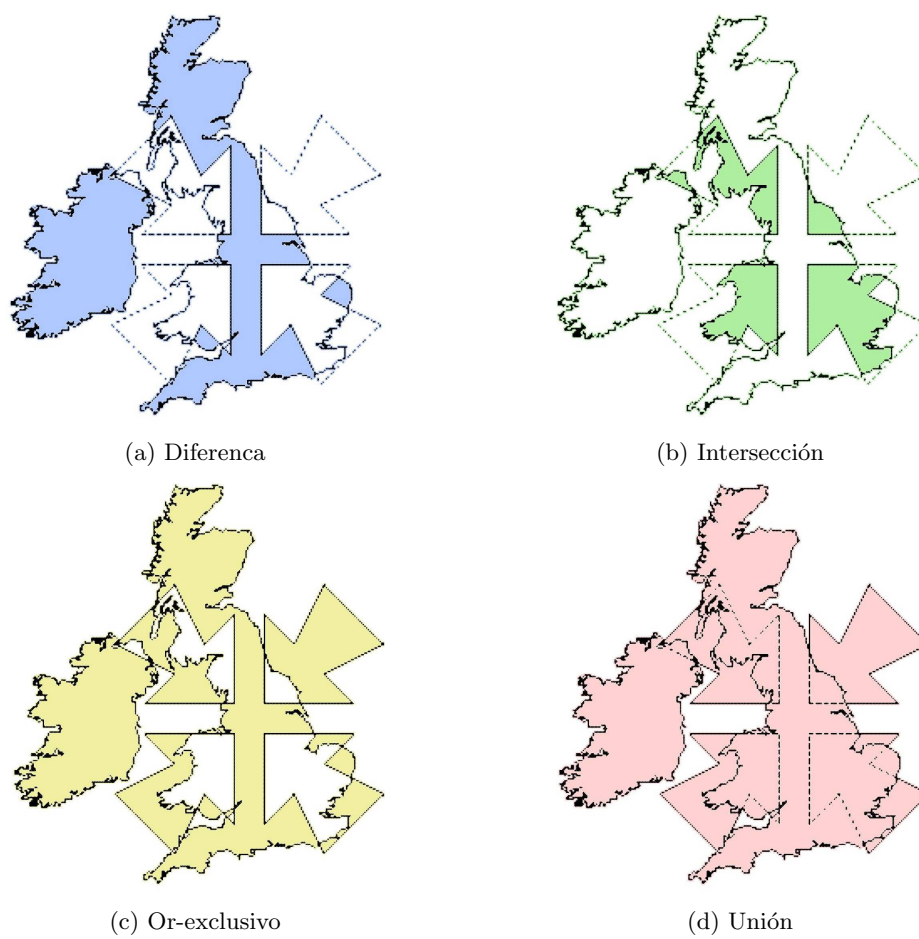


Figura A.2: Operaciones de recorte que maneja la librería GPC.

En ambos casos el primer parámetro especifica la operación de recorte (GPC DIFF, GPC INT, GPC XOR, GPC UNION). Los siguientes parámetros son el polígono sujeto y el de corte. En el parámetro final se almacenará el resultado, ya sea como polígono o una colección de trisrip.

Un polígono puede convertirse en su equivalente trisrip con la función,

```
void gpc_polygon_to_trisrip(gpc_polygon *source_p,  
gpc_trisrip result_t);
```

Las últimas dos funciones son para liberar la memoria utilizada por las estructuras de datos para los polígonos y para los trisrips.

```
void gpc_free_polygon(gpc_polygon *p);  
void gpc_free_trisrip(gpc_trisrip *t);
```

A.3. Huecos y contornos externos

Un contorno se clasifica como un contorno externo si su vértice más alto (o más a la derecha, cuando el contorno es horizontal) forma un máximo local externo (MLE). Si este vértice forma un máximo local interno (MLI) el contorno se clasifica como un contorno de un hueco.

Cuando los lados del contorno se cruzan (por ejemplo en figuras que se interceptan a sí mismas) siempre generan un vértice máximo local debajo de la intersección (o cuando uno de los lados es horizontal, a la izquierda de la intersección) lo cual conecta las partes de los dos contornos que se encuentran por debajo o a la izquierda del punto de intersección. Las partes de los lados que surgen en el costado opuesto al punto de intersección originan un nuevo vértice mínimo local. El contorno cerrado generado nunca se interceptará a sí mismo.

Estas reglas tienen implicación con respecto a cómo descomponer las figuras que se interceptan a sí mismas en un conjunto de contornos cerrados que no se intercepten. La Figura A.3a muestra ejemplos de cuadrados interceptándose, cada uno creará un contorno en su interior. En cada caso, el vértice máximo arriba o a la derecha del contorno interno forma un máximo interno, por tanto, el contorno es clasificado como un hueco (línea punteada). El correspondiente contorno exterior es considerado externo ya que termina con un vértice máximo externo.

Los ejemplos en la Figura A.3b, muestran figuras similares que se intersectan a sí mismas, sin embargo, cada una crea dos contornos externos tocándose en el punto de intersección. En resumen, los contornos generados por el recortador no sólo son afectados por la forma de los polígonos entrantes sino también por su orientación.

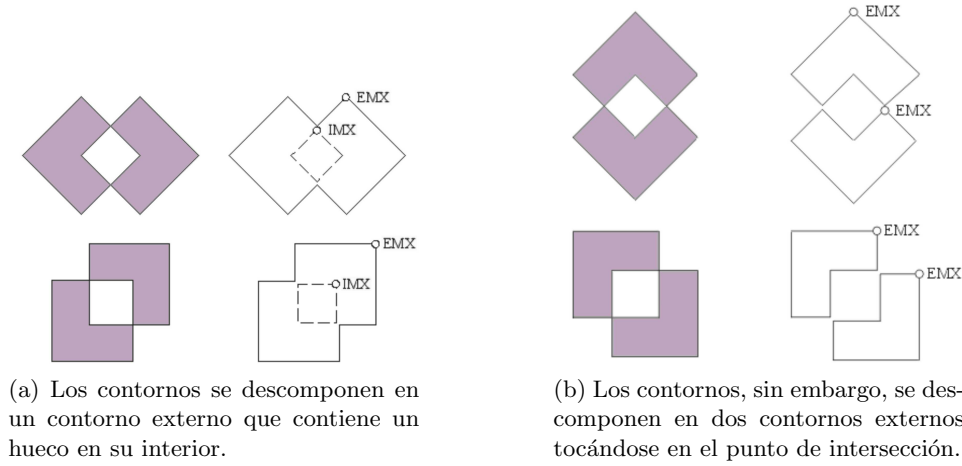


Figura A.3: Contornos que se interceptan a sí mismos.

A.4. Asociación de los huecos con el contorno externo

La actual versión simplemente utiliza las banderas para identificar cuales contornos pueden ser considerados huecos y cuales externos. Descubriendo cuales huecos se sitúan en que contornos externos toma un poco más de trabajo por parte del usuario, una forma de asociar los huecos $H_1, H_2, \dots, H_n$ con los contornos externos $E_1, E_2, \dots, E_n$ es usar la librería para calcular la diferencia del i -ésimo hueco con cada contorno externo E_j , para todo j de 1 a m . Cualquier diferencia que obtenga un resultado vacío indica que el i -ésimo hueco se encuentra en el contorno externo j .

A.5. Lados coincidentes y casi-coincidentes

El cortador fusiona los lados que son coincidentes. Los contornos adyacentes del polígono sujeto y del cortador que comparten un lado en común los fusionará para formar un simple contorno bajo la operación de unión, y producirá un resultado nulo a lo largo del límite compartido con la operación de intersección. La precisión de los límites numéricos es probablemente la causa de la leve pérdida de registro de los lados coincidentes, resultando en un error en la fusión. Incrementando el valor de la constante `GPC_EPSILON` en `gpc.h` incluirá en la fusión de lados casi-coincidentes. Sin embargo, pueden resultar figuras poligonales incorrectas si se da a `GPC_EPSILON` un valor bastante grande.

Índice de figuras

1.1. El campo de la exploración robótica con las regiones de integración destacadas.	7
2.1. Algoritmo básico de construcción del SRT.	15
2.2. Generación de configuraciones candidatas con el método SRT. . . .	15
2.3. Las figuras muestran la comparativa de la RS sin visibilidad a) y la RS con visibilidad b), en las siguientes figuras c) y d) se muestran la misma comparativa ocultando los obstaculos.	17
2.4. (a) El punto q esta fuera de la envolvente convexa de P . (b) El punto q esta fuera de P pero dentro de la envolvente convexa. . . .	19
2.5. (a) El vértice v_i es insertado en la pila. (b) El vértice de $bd(v_i, v_{k-1})$ no es visible desde q	20
2.6. (a) El lado $v_{i-1}v_i$ intersecta uq . (b) El lado $v_{i-1}v_i$ no intersecta uq	20
2.7. (a) El retroceso termina insertando m y v_i en la pila (b) El retroceso continua con $v_{k-1}v_k$ como el lado actual.	22
2.8. (a) El lado $v_{i-1}v_i$ no intersecta el lado construido uw . b) El lado $v_{i-1}v_i$ intersecta el uw en p y el lado $v_{k-1}v_k$ intersecta pw en z	23
2.9. 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.	25
2.10. Diferentes radios obtenidos en la región segura local S con la estrategia de percepción SRT-Radial y la aplicación de la libreria GPC. . . .	25
2.11. Árbol resultante utilizando el método SRT en 3 ambientes; en los escenarios, en color negro se puede ver al robot	26
3.1. Izquierda: sensores centrales $s_i(q)$ y $s_{i+1}(q)$ y el campo de la vista asociada $F_i(q)$ y $F_{i+1}(q)$ cuando el robot esta en la configuración q . Derecha: la vista V_q y el límite visible del obstáculo $B(q)$	31
3.2. Descripción en pseudocódigo de la iteración k -ésima del método SET en el cual el robot inicia en q^k	36
3.3. Descripción en pseudocódigo de la estrategia de búsqueda con crecimiento global (GG)	40

3.4. Descripción en pseudocódigo de la estrategia de búsqueda con crecimiento local (LG)	40
4.1. Escenarios usados para las simulaciones; en los escenarios, en color negro se puede ver al robot	44
4.2. Las figuras muestran los resultados obtenidos del ambiente 1 con el método SET.	45
4.3. Las figuras muestran los resultados obtenidos del ambiente 1 con el método SRT.	46
4.4. Las figuras muestran los resultados obtenidos del ambiente 2 con el método SET.	47
4.5. Las figuras muestran los resultados obtenidos del ambiente 2 con el método SRT.	48
4.6. Las figuras muestran los resultados obtenidos del ambiente 3 con el método SET.	49
4.7. Las figuras muestran los resultados obtenidos del ambiente 3 con el método SRT.	50
A.1. Polígono genérico con cuatro contornos.	55
A.2. Operaciones de recorte que maneja la librería GPC.	57
A.3. Contornos que se interceptan a sí mismos.	59

Índice de cuadros

4.1. Resultados para el ambiente 1	43
4.2. Resultados para el ambiente 2	43
4.3. Resultados para el ambiente 3	51

Bibliografía

- [1] J. Ahuactzin and A. Portilla, A basic 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.
- [2] A. Bennett and J.J. Leonard. A behavior-based approach to adaptive feature mapping with autonomous underwater vehicles. IEEE J. of Oceanic Engineering, 25(2):213-226, 2000.
- [3] P. Bessière, E. Mazer, and J.M. Ahuactzin. "The ariadnés clew algorithm: Global planning with local methods". Workshop on the Algorithmic Foundations of robotics, 1994.
- [4] F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, and H. Durrant-Whyte, Information based adaptive robotic exploration, in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2002.
- [5] [2] J.F. Canny. "The complexity of robot motion planning". MIT Press, Cambridge, MA, 1988.
- [6] I.J. Cox and G.T. Wilfong, editors. Autonomous Robot Vehicles. Springer-Verlag, USA, 1990.
- [7] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csobor. A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. on Robotics and Automation, 17(3):229-241, 2001.
- [8] S.K. Ebiyara, T. Otani, and E. Kume. Position localisation for mobile robots using a colour image of equipment at nuclear plants. Robotica, 14(6):677-685, 1996.
- [9] Espinoza León Judith. Estrategias de exploración de ambientes desconocidos en robótica móvil. Tesis de maestría FCC-BUAP (2006).
- [10] H. Feder, J. Leonard, and C. Smith, Adaptive mobile robot navigation and mapping, Int. J. Robotics Research, vol. 18, no. 7, pp. 650-668, 1999.

- [11] D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 1998.
- [12] 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.
- [13] L. Freda, F. Loiudice, and G. Oriolo, A randomized method for integrated exploration, *Tech. Rep.*, Università di Roma La Sapienza, 2006.
- [14] H.H. Gonzalez-Banos and J.C. Latombe, Navigation strategies for exploring indoor environments, *Int. J. Robotics Research*, vol. 21, no. 10, pp. 829-848, 2002.
- [15] R.L. Graham and F.F. Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4:324-331, 1983.
- [16] D. Hähnel, W. Burgard, D. Fox, and S. Thrun, An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003.
- [17] L. Kaelbling, A. Cassandra, and J. Kurien, Acting under uncertainty: Discrete bayesian models for mobile robot navigation, in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1996.
- [18] E. Kruse, R. Gutschke, and F. Wahl, Efficient, iterative, sensor based 3-d map building using rating functions in configuration space, in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1996, pp. 1067-1072.
- [19] [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.
- [20] D.T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22:207-221, 1983.
- [21] D. Lee. *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Robot : An Experimental, Quantitative Evaluation*. Distinguished Dissertations in Computer Science. Cambridge University Press, Cambridge; New York, 1996.
- [22] J.J. Leonard and H.F. Durrant-Whyte. Simultaneous map building and localisation for an autonomous robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '91)*, pages 1442-1447, Osaka, Japan, 1991.

- [23] J.J. Leonard and H.F. Durrant-Whyte. Directed Sonar Sensing for Mobile Robot Navigation. Kluwer Academic Publishers, 1992.
- [24] A. Makarenko, S.B. Williams, F. Bourgault, and H. F. Durrant-Whyte, An experiment in integrated exploration, in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2002, vol. 1, pp. 534-539.
- [25] P.S. Maybeck. Stochastic Models, Estimation, and Control, volume 1. Academic Press, 1979.
- [26] S. Moorehead, R. Simmons, and W.R.L. Whittaker. Autonomous exploration using multiple sources of information. In IEEE International Conference on Robotics and Automation (ICRA '01), 2001.
- [27] 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.
- [28] G. Oriolo, M. Venditelli, L. Freda, and G. Troso. The SRT method: Randomized strategies for exploration. Proceedings of the IEEE, International Conference on Robotics & Automation, 2004.
- [29] R.F. Poppen. Using partial or nonstandard GPS solutions in automotive navigation. In Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation, pages 59-62, 1993.
- [30] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In the IEEE International Conference on Robotics and Automation (ICRA '99), volume 1, Detroit, MI, USA, 1999.
- [31] A. Schultz, W. Adams, and B. Yamauchi. Integrating exploration, localization, navigation and planning with a common representation. Autonomous Robots, 6(3):293-308, 1999.
- [32] R. Sim, G. Dudek, and N. Roy, Online control policy optimization for minimizing map uncertainty during exploration, in IEEE Int. Conf. on Robotics and Automation, 2004.
- [33] R.G. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In AAAI National Conference on Artificial Intelligence, pages 852-858, Austin, TX, USA, 2000.
- [34] C. Stachniss, G. Grisetti, and W. Burgard, Information gain-based exploration using rao-blackwellized particle filters, in IEEE Int. Conf. on Robotics and Automation, 2005.

-
- [35] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots*, 31(1-3):29-53, 1998.
 - [36] S. Thrun, Wolfram Burgard and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.
 - [37] B. Yamauchi, A frontier-based approach for autonomous exploration, in 1997 IEEE Int. Conf. on Robotics and Automation, 1997.
 - [38] Y. Yu and K. Gupta, C-space entropy: A measure 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.