Navegación óptima basada en discontinuidades de distancia para un robot de manejo diferencial con forma de disco

por

M. en I. Rigoberto López Padilla

Universidad Guanajuato (2007)

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

Doctor en Ciencias de la Computación

en el

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS A.C.

Abril 2014

Firma del autor.....
Departamento de Ciencias de la Computación
Marzo del 2014

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

Certificado por .....
Dr. Victor Ayala Ramírez
Universidad de Guanajuato — , Profesor Titular B
Codirector de Tesis

Aceptado por .....
Dr. Rogelio Hasimoto Beltrán
Coordinador del Postgrado Ciencias de la Computación

1

Navegación óptima basada en discontinuidades de distancia para un robot de manejo diferencial con forma de disco

por

M. en I. Rigoberto López Padilla

Sometida a revisión al Departamento de Ciencias de la Computación
en Marzo del 2014, en el cumplimiento parcial de los
requisitos para obtener el grado de
Doctor en Ciencias de la Computación

1

Navegación óptima basada en discontinuidades de distancia para un robot de manejo diferencial con forma de disco

por

M. en I. Rigoberto López Padilla

Sometida a revisión al Departamento de Ciencias de la Computación
en Marzo del 2014, en el cumplimiento parcial de los
requisitos para obtener el grado de
Doctor en Ciencias de la Computación

Resumen

Esta tesis considera el problema de la navegación globalmente mínima, con respecto a la distancia Euclidiana, para un robot de manejo diferencial con forma de disco colocado en una región poligonal simplemente conectada desconocida. Suponemos que el robot es incapaz de construir mapas geométricos precisos del entorno. La mayoría de la información proviene del sensor de brechas, el cual indica discontinuidades en profundidad, permitiendo que el robot se mueva hacia ellas. Se presenta una estrategia de movimiento que permite que el robot navegue óptimamente hacia cualquier landmark en la región. La optimalidad es probada y el método es ilustrado en simulación.

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

Codirector de Tesis: Dr. Victor Ayala Ramírez
Título: Universidad de Guanajuato — , Profesor Titular B

1

Agradecimientos

A mis padres, porque me han apoyado en cada proyecto que he decidido emprender y por lo que me han enseñado con su buen ejemplo.

A mi esposa Ma. Consuelo Ayala Rodriguez, por estar conmigo incondicionalmente. Me has dado palabras de aliento en los momentos más difíciles. Has sido un gran impulso en mi vida. Te amo.

A mi asesor el Dr. Rafael Murrieta Cid, por la paciencia que me ha tenido y por todo el trabajo invertido en esta tesis.

Al Dr. Steven M. LaValle, por sus aportaciones a esta tesis y por el apoyo recibido para realizar una estancia en la Universidad de Illinois en Urbana-Champaign como parte de su grupo de investigación.

A mi co-asesor el Dr. Victor Ayala Ramírez, porque ha sido mi guía durante todo el proceso de mi formación profesional.

1

Agradecimientos institucionales

Quiero agradecer a las siguientes instituciones por hacer posible esta obra:

- Al consejo Nacional de Ciencia y Tecnología por el apoyo mediante la asignación de la beca nacional y beca mixta No. 160296.
- Al proyecto planificación de movimientos y percepción para robots: Mundos tridimencionales y manejo de incertidumbre. Proyecto de ciencia básica CONACyT 106475.
- Al Centro de Investigación en Matemáticas CIMAT por haberme brindado la oportunidad de integrarme en uno sus posgrados. Además, por la Beca de elaboración de tesis de Doctorado durante el periodo del 1 de junio del 2013 al 30 de septiembre del 2013.
- A la Universidad de Illinois en Urbana-Champaign por brindarme la oportunidad de realizar una estancia académica durante el periodo de marzo a septiembre del 2011.

1

Índice general

1

Índice de figuras

1

Índice de tablas

Capítulo 1

Introducción

Se espera que en un futuro cercano, los robots formen parte de nuestro entorno. En la actualidad, podemos encontrar robots en la industria, en el hogar, en laboratorios dentro de universidades o centros de investigación, etc. Existen robots que en el hogar ayudan a la limpieza, juegan el rol de la mascota, cortan el césped, etc. Los robots son ampliamente utilizados para realizar tareas de forma más exacta o más barata que los humanos. En la industria podemos encontrarlos en las plantas de manufactura, montaje y ensamblaje, en transporte, etc. Se espera que en los próximos años los robots puedan realizar tareas cada vez más complicadas como manejar un automóvil, prevenir situaciones de riesgo en las carreteras, brindar autonomía a personas discapacitadas que carecen de movilidad, etc. Las capacidades de los robots van ligadas a la complejidad de los modelos del ambiente que el robot usa. Es importante desarrollar modelos eficientes, para realizar tareas cada vez más complicadas, procesando y seleccionando información cada vez en forma más eficiente. Para esto, un robot debe saber cual es la información útil del ambiente donde está navegando y cuales son las operaciones que debe realizar para obtener esa información.

1.1. Campo de estudio y motivación

Dentro de la robótica podemos encontrar varias disciplinas que colaboran con el objetivo de dar autonomía a diversos tipos de robots, por ejemplo, la inteligencia artificial [75], visión por computadora [32], planificación de movimientos [51], etc. Diversas disciplinas requieren de modelos para representar el ambiente en el cual se encuentra el robot. Algunas suponen que el modelo es conocido y que éste cuenta con los elementos suficientes para que un robot

pueda realizar tareas en su ambiente. Sin embargo, esta suposición no siempre es valida ya que existen varios factores, como la incertidumbre en la localización y en el sensado, que pueden afectar el buen desempeño de las tareas del robot. Actualmente existen técnicas bastante estudiadas que permiten tratar con problemas de la incertidumbre, como es el caso de la localización y construcción de mapas simultánea (SLAM[1]).

Trabajos previos se han enfocado primeramente en problemas relacionados con la incertidumbre en el sensado y en el control. Las primeras investigaciones sobre SLAM usaron un enfoque basado en filtros de Kalman para lidiar con las incertidumbres que se acumulaban durante el movimiento del robot, proporcionando simultáneamente una estimación de la posición del robot y la estimación de la posición de las referencias perceptuales (landmarks). Recientemente, enfoques Bayesianos generalizados han sido propuestos para el problema de SLAM, relajando las condiciones restrictivas impuestas por los métodos de filtrado de Kalman [24].

En esta tesis se abordan temas relacionados con SLAM. Sin embargo, nuestro interés principal es el problema de planificación de estrategias de navegación robótica óptima usando representaciones topologicas del ambiente. Este problema ha sido menos estudiado que el problema de SLAM.

1.2. Objetivo general

Nuestro principal objetivo es extender el enfoque del árbol de navegación basado en brechas (Gap Navigation Tree o GNT) [93] a robots tipo disco (con área no nula), buscando conservar la propiedad de que esta estructura después de ser construida permite navegar de forma óptima en términos de distancia euclidiana, entre cualquier dos localidades del ambiente. Esto requiere un análisis entre la diferencia del espacio de configuraciones[2] del robot puntual y el del robot disco. Es conveniente señalar que una localidad de sensado en el ambiente se caracteriza por la información obtenida por los sensores, por ejemplo, "mover al robot de la pelota roja al cubo azul", permitiendo navegación basada en observaciones más que en posiciones geométricas.

[1]Por sus siglas en inglés Simultaneous Localization and Mapping

[2]La definición está en el Apéndice A

1.3. Planteamiento del problema

El robot es un sistema de manejo diferencial (un robot con dos ruedas, donde cada rueda tiene un motor independiente) que tiene definida la dirección hacia adelante (heading). El robot tiene la forma de un disco con radio r que se mueve en un entorno plano y poligonal, el cual puede ser cualquier conjunto compacto $E \subset \mathbb{R}^2$ para los cuales el interior de E es simplemente conectado. La frontera, ∂E, de E es la imagen de una curva cerrada analítica a trozos. Sin embargo, se supone que el subconjunto libre de colisión del espacio de configuraciones del robot está simplemente conectado o puede tener varios componentes conectados.

El espacio de configuraciones obstáculo proyectado en el plano corresponde a la de un disco crecido un radio r, esto es, la frontera extendida de E lo cual es debido al radio del robot[2].

Sea Λ una landmark estática ubicada en el entorno que tiene la forma de un disco con el mismo radio que el robot. Suponemos que Λ está pintada en el suelo, por lo tanto, no tiene volumen y no produce discontinuidades en distancia (brechas). Esta suposición es hecha ya que el robot tiene como meta estacionarse sobre la landmark.

Un objetivo principal de este trabajo es que el robot navegue con una trayectoria hacia Λ que minimiza la distancia Euclidiana viajada por el centro del robot. Además, estamos interesados en conocer si tal trayectoria existe o no.

1.4. Contenido del documento

El contenido del documento está organizado como sigue. En el Capítulo 2 se presenta el estado del arte de las estrategias de exploración y construcción de representaciones de ambientes para robots. También se presentan métodos de navegación robótica, principal tema de esta tesis. Los enfoques de construcción de representaciones topológicas que sustentan esta tesis son presentados en el Capítulo 3. La navegación óptima para un robot con forma de disco es presentada en el Capítulo 4. En el Capítulo 5 se presentan las conclusiones generales del trabajo y se presenta posible trabajo futuro. En el Apéndice A se incluyen definiciones de conceptos utilizados en todo el documento y en el Apéndice B se presentan marcos de referencia locales donde se ubican vértices del ambiente poligonal.

[2]Note que esto es el espacio de configuraciones correspondiente al de un disco expandido el radio r y no el de cuerpo rígido, debido a la simetría rotacional.

La investigación realizada en esta tesis se ha reportado en las siguientes publicaciones:

- R. Lopez-Padilla, R. Murrieta-Cid, and S. M. LaValle. Optimal Gap Navigation for a Disc Robot. Proc. Tenth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2012, MIT Cambridge MA, USA, E. Frazzoli et al (Eds.) Springer Tracts in Advanced Robotics, STAR 86, pp. 123-138, 2013.
- G. Laguna, R. Murrieta-Cid, H. M. Becerra, R. Lopez-Padilla and Steven M. LaValle. Exploration of an Unknown Environment with a Differential Drive Disc Robot. *Accepted IEEE Int. Conf. on Robotics & Automation*, 2014.

Considero importante mencionar que además del proyecto de investigación doctoral que reporto en esta tesis, durante mi estancia en el CIMAT participé en el proyecto de generación de estrategias de exploración para construcción de mapas geométricos con múltiples robots heterogéneos, investigación que se ha reportado en las siguientes publicaciones:

- Motion Strategies for Exploration and Map-Building under Uncertainty with Multiple Heterogeneous Robots, L. Valentin, R. Murrieta-Cid, L. Muñoz, R. Lopez-Padilla and Moises Alencastre, Accepted to Journal Advanced Robotics, 2013.
- Exploration and Map-Building under Uncertainty with Multiple Heterogeneous Robots, L. Muñoz, M. Alencastre, R. Lopez-Padilla and R. Murrieta-Cid *IEEE International Conference on Robotics and Automation*, pages 2295-2301, Shanghai China, ICRA 2011.

Capítulo 2

Estado del Arte

La construcción automática de un modelo del ambiente ha sido un tópico ampliamente explorado en la investigación relacionada a robots móviles autónomos [20] [87] [17] [30] [89] [3]. Los modelos del entorno han sido utilizados para realizar otras tareas, por ejemplo, navegación (moverse de un lugar a otro sin chocar con obstáculos), búsqueda de objetos, seguimiento de blancos, etc.

En general, un sistema para la construcción automática del modelo del ambiente, debe considerar tres problemas que se relacionan entre sí: i) la construcción incremental del modelo del entorno, ii) la estimación de la configuración del robot con referencia al modelo del entorno, y iii) la definición de una estrategia de movimiento del robot.

2.1. Modelos del entorno

Primeramente debemos escoger el modelo que queremos utilizar para representar el entorno. En [20] se proponen tres tipos de modelos: topológicos, geométricos y semánticos. En [94] se mencionan los mapas topológicos [23], las rejillas de ocupación y los mapas basados en primitivas geométricas [48] [86] como los modelos que han sido principalmente utilizados para la construcción automática de representaciones del ambiente.

2.1.1. Modelo geométrico

El modelo geométrico es directamente deducido de los datos de percepción. Se trata con objetos considerados como obstáculos.

En [20], los objetos tienen formas poliédricas y son proyectados en el plano para obtener un modelo en 2D. Este modelo básicamente contiene la posición en un marco de coordenadas absolutas de los vértices de la proyección del objeto. Esta representación puede ser manipulada para deducir los modelos topológicos y semánticos.

Las rejillas de ocupación usan un arreglo 2D para representar el entorno, donde cada celda toma uno de tres valores: espacio libre, espacio ocupado y espacio desconocido. Los algoritmos basados en rejillas han probado ser muy simples y bastante fáciles de utilizar para evitar colisión entre el robot y los obstáculos durante la navegación del robot [30]. Sin embargo, existen varias desventajas importantes en estos modelos. Su utilización se vuelve compleja en ambientes grandes o 3D y es poco practica con el uso de robots con muchos grados de libertad o sistemas multi-robot.

Los mapas basados en primitivas geométricas [48] pueden representar un modelo 2D [20] o un modelo 3D [86], son otra forma de representar el entorno usando primitivas geométricas. La primitiva geométrica más popular es el segmento de línea, el cual puede ser extraído de información de sensores de ultrasonido [26], información de telémetros láser [36] o información visual [5] [43]. Resultados de otros equipos de investigación son mostrados en la Figura 2.1.

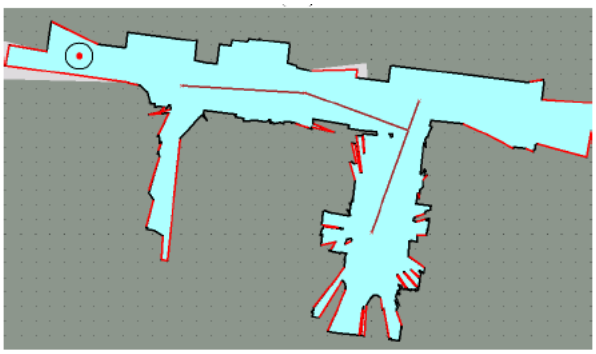


Figura 2.1: Modelo Geométrico [37].

2.1.2. Modelo topológico

Los modelos topológicos pueden ser expresados como grafos, donde los nodos representan lugares y las aristas representan adyacencia o conectividad [20].

El modelo topológico es una jerarquía de varios niveles. El concepto de **lugar** es definido

como una área que es una unidad topológica. Por ejemplo, un cuarto o un corredor son unidades topológicas. El concepto de lugar está definido a varios niveles: un cuarto, un piso, o un edificio completo son lugares. En un entorno de exterior, el concepto de lugar puede ser definido también. Los conectores enlazan lugares: puertas, corredores, escaleras o elevadores son por ejemplo conectores en ambientes de interior. Algunos lugares también pueden ser conectores ellos mismos. En el nivel más detallado, la topología del espacio es típicamente deducida del modelo geométrico estructurando el espacio libre. La estructuración del espacio libre puede ser obtenida construyendo polígonos convexos, llamados celdas basados en los vértices y aristas de obstáculos. Una celda es un lugar, y un conjunto de celdas puede ser otro lugar en una jerarquía más alta. Las aristas comunes entre celdas son conectores. El grafo de conectividad de celdas constituye la primera representación topológica de la estructura del espacio. Mientras el robot explora su entorno y descubre nuevas áreas, la estructuración del espacio es realizada, y el grafo es aumentado y modificado.

En [93] se presenta una estructura topológica de navegación el Gap Navigation Tree (GNT) que no se basa en ninguna métrica. Esta estructura se basa en el seguimiento de brechas (discontinuidades en distancia) para la construcción de la estructura topológica.

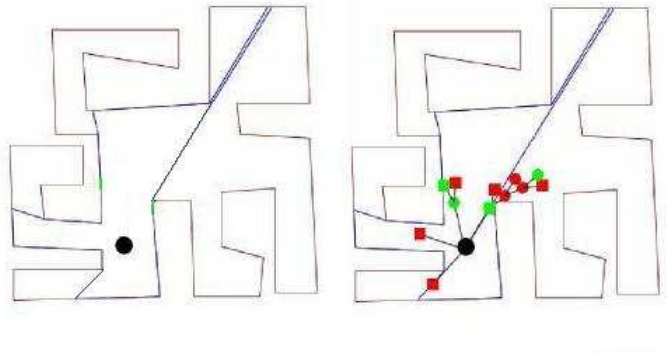


Figura 2.2: Modelo topológico. Figura obtenida del trabajo presentado en [93].

2.1.3. Modelo semántico

El modelo semántico contiene información acerca de objetos, por ejemplo su clase o categoría y propiedades del espacio y sus relaciones. Este modelo está parcialmente distribuido entre el modelo geométrico y el modelo topológico [20]. En [84], se obtienen etiquetas semánticas de lugares (corredores y cuartos) basadas en la información de un

telémetro láser.

Figura 2.3: Modelo Semántico. Figura obtenida del trabajo presentado en [69].

2.2. Construcción incremental de mapas geométricos y topológicos

La construcción de mapas es un proceso incremental. Dado que el rango de alcance de los sensores está limitado, pueden ocurrir oclusiones durante la exploración y construcción del modelo. Además, las medidas pueden ser inexactas. La construcción de mapas es usualmente desarrollada tomando varias lecturas del entorno a diferentes posiciones e integrando estas medidas en un mapa global.

Idealmente, un modelo 2D de un entorno de interior es la proyección, sobre un plano horizontal, de la porción de objetos en el entorno que son obstáculos para el movimiento del robot y/o obstruyen el campo de vista de sus sensores. Sin embargo, debido a las limitaciones de los sensores, los modelos son a menudo aproximados como una sección transversal del entorno a una altura dada.

Un enfoque comúnmente utilizado es la integración de la información de un telémetro láser. En la Figura 2.4 se muestra la información obtenida por un telémetro láser. En [37] el robot construye un modelo poligonal local (lo que ve en ese momento) moviéndose a posiciones sucesivas del entorno, las cuales son seleccionadas por un planificador. Estas posiciones también pueden ser escogidas por un usuario humano. La Figura 2.5 muestra cuatro modelos parciales de un ambiente. El robot se encuentra en una localidad donde gira

a cuatro orientaciones sucesivas espaciadas de 90 grados. El modelo en (a) fue construido en la primera iteración. El modelo en (b) fue obtenido uniendo el modelo de (a) con el modelo local generado en la segunda orientación y así sucesivamente.

Es además necesario para la construcción de un mapa geométrico global un proceso de alineamiento o registro entre los modelos locales y el modelo global, que como ya se mencionó es construido de forma incremental (ver Figura 2.4).

Adicionalmente, los datos crudos obtenidos por el sensor pueden ser agrupados en otras entidades, por ejemplo los puntos obtenidos por un láser pueden ser agrupados en segmentos de línea por medio de agrupamiento ("clustering") y de un ajuste de curvas. Los segmentos de línea resultantes a su vez pueden ser agrupados en listas de segmentos que dan origen a polígonos.

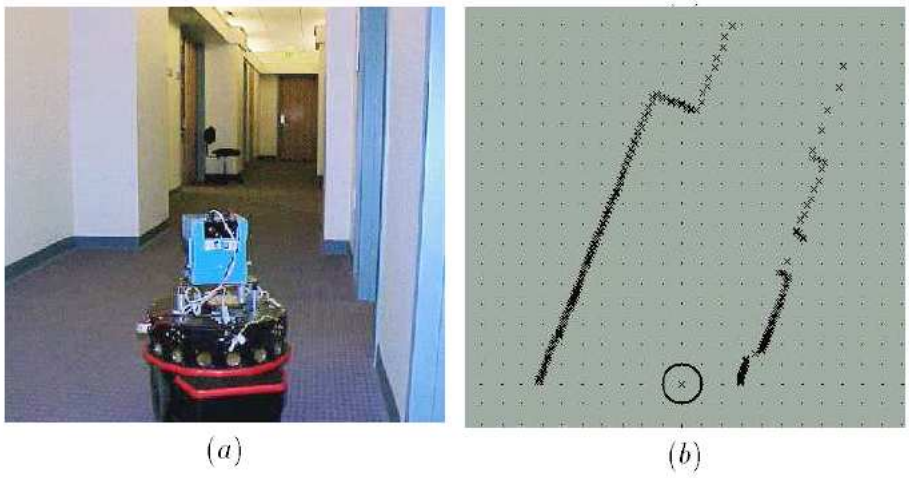


Figura 2.4: Lecturas del telémetro láser. (a) Escena real (b) Puntos capturados [37].

En algún momento durante el proceso de construcción incremental de mapas, las estrategias de movimiento para exploración manejan la selección de posiciones de observación, sobre la base del mapa global construido hasta ese momento. Este punto hace el problema diferente en general de la tarea de encontrar una ruta libre de colisión para ir de una configuración inicial a otra final, dado que las sub-metas (configuraciones de sensado) deben determinarse automáticamente. Excluyendo los casos en los cuales el robot es manejado manualmente, en lo siguiente se mencionan algunas estrategias de exploración que han sido presentadas en la literatura.

Existen sistemas que emplean una camino fijo para explorar un entorno. Por ejemplo,

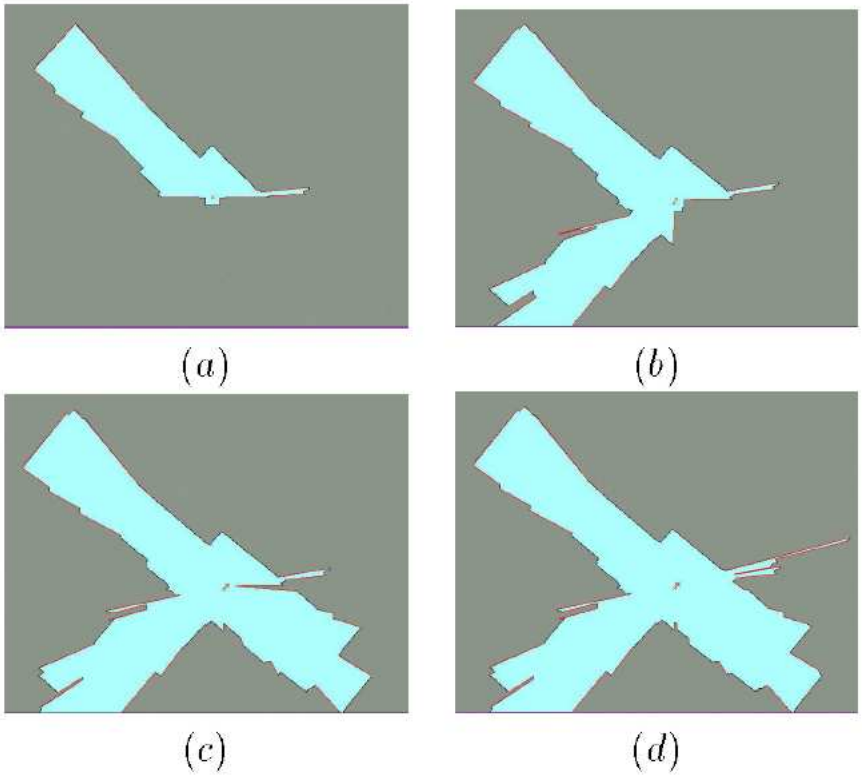


Figura 2.5: Unión de cuatro modelos parciales en una posición dada [37].

en la estrategia de exploración propuesta en [78], el robot traza círculos concéntricos sucesivamente empezando de su posición inicial. Este enfoque es extendido en [79] usando trayectorias en espiral parametrizadas para dirigir la exploración del robot en un entorno. Otras estrategias de exploración hacen que un robot se mueva en forma aleatoria a posiciones de observación sin una evaluación explícita de la importancia de estas posiciones.

En [93] se presenta una estrategia de movimiento que se basa en el seguimiento de brechas (discontinuidades espaciales) para la construcción de una estructura topológica dinámica, llamada GNT[1], que caracteriza el entorno de un robot. Una vez que el GNT es construido, codifica trayectorias desde la posición del robot hasta cualquier lugar en el entorno. Mientras el robot se mueve, el GNT es actualizado para mantener la información de las trayectorias más cortas desde la posición actual del robot. Las trayectorias son globalmente óptimas en distancia euclidiana si el entorno está simplemente conectado[2], incluso si no se cuenta con información geométrica. Murphy y Newman [68] extienden el GNT para detectar las brechas con un modelo probabilístico logrando mayor robustez.

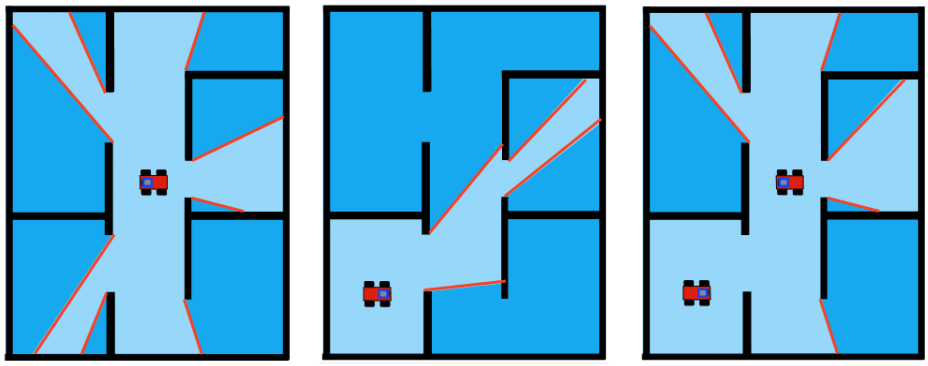
Las estrategias de exploración más complejas tratan de determinar la mejor posición de observación para mejorar la eficiencia del proceso de exploración, es decir, utilizan la planificación de movimientos para decidir la siguiente posición de observación. En general, todas las estrategias que tratan de determinar una localidad de sensado para explorar el ambiente envían al robot a la frontera entre lo que el robot ha percibido y el mundo aún no explorado (ver Figura 2.6). En [1] [94], se describen y comparan algunas estrategias de exploración recientes. En la siguiente sección, se presenta un reporte de estrategias de exploración que utilizan la planificación de movimientos para determinar las configuraciones más convenientes de futura observación del robot, en un entorno parcialmente conocido.

2.3. Estrategias de movimiento para exploración y construcción de representaciones del ambiente

La mayoría de los trabajos previos en exploración y construcción de mapas se han enfocado a desarrollar técnicas para extraer información relevante a partir de los datos en bruto e integrar los datos conseguidos dentro de un solo modelo. Sin embargo, típicamente no se desarrolla una planificación de movimientos para la exploración, que además facilite la construcción del modelo del entorno.

[1]Por sus siglas en inglés Gap Navigation Tree

[2]La definición está en el Apéndice A



(a) Mapa Pasillo (b) Mapa Oficina (c) Fusión de mapas

Figura 2.6: (a) y (b) Visibilidad del robot y fronteras de lo conocido/desconocido en el mapa local, (c) Lo conocido por el robot (o robots) a partir de dos localidades de sensado y fronteras de lo conocido/desconocido en el mapa global.

Para ser eficiente en la exploración y construcción de mapas, es posible utilizar la planificación de movimientos para determinar las configuraciones más convenientes de futura observación del robot en un entorno parcialmente conocido.

La planificación de movimientos para la exploración trata de determinar la configuración más conveniente de la futura observación basado en un modelo del entorno parcialmente conocido. En particular, el objetivo es reducir el tiempo de exploración minimizando los pasos de observación y la distancia viajada, mientras se construye una representación exacta del entorno.

2.3.1. Enfoque Glotón

El enfoque glotón evalúa una posición de observación candidata p sólo en base a su ganancia inmediata, donde la ganancia es comúnmente definida como la cantidad de nueva información del entorno que el robot espera obtener desde p. Además, el horizonte de planificación es de un solo paso en el futuro, es por eso que es llamado glotón.

2.3.2. Enfoque de González-Baños

En [37] [38], se presenta una estrategia de planificación de movimiento para la construcción de mapas. La estrategia presentada en [38] está basada en el cálculo de la

siguiente mejor vista (NBV[3]) y el uso de planificación de movimientos aleatorizada. Esa investigación ha mostrado que es posible encontrar una función que refleja intuitivamente como debe el robot explorar el espacio. En un esquema simple, la función de evaluación debe asignar un valor grande a la posición que mejor se ajuste al compromiso entre la posible reducción del espacio no explorado y la distancia recorrida. Así en [38] se toma en cuenta tanto el costo asociado al movimiento hacia la siguiente localidad de sensado, como la ganancia de la misma en términos de nueva área percibida. La estrategia evalúa una posición de observación candidata p mediante:

$$g(p) = A(p) \cdot \exp(-\lambda L(p)), \tag{2.1}$$

donde $A(p)$ es una estimación del área visible desde p, $L(p)$ es la longitud de la trayectoria que conecta la posición del robot y la posición p, y λ pesa el costo de viajar para alcanzar la posición.

2.3.3. Enfoque de Makarenko

El enfoque presentado en [61] propone una estrategia de exploración para construcción de mapas y localización. El método de la frontera [98] es utilizado para proponer futuras localidades de sensado que tienden a estar en la frontera entre lo conocido y lo no conocido. La estrategia de exploración hace uso de una función de utilidad para evaluar las futuras localidades de sensado. Esta función toma en cuenta tres elementos: la ganancia de información, la distancia a la localización de sensado (costo) y la utilidad de la habilidad para localizarse, basada en una matriz de covarianza.

Utilidad de la ganancia de información. Esta utilidad está diseñada para favorecer localidades que ofrecen una ganancia de información alta. La utilidad de información es obtenida a partir de un mapa de rejillas de ocupación. Una celda C puede tomar dos valores espacio ocupado OCC o espacio vacío EMP, es decir, $s(C) = OCC \mid EMP$. Para medir la cantidad de información disponible para cada celda, la entropía H de la distribución binaria $s(C)$ es calculada como:

$$H(s(C)) = -\sum_{s(C)} P(s(C)) \log P(s(C)) \tag{2.2}$$

[3]Por sus siglas en inglés Next Best View

La utilidad de hacer una observación desde la localidad destino x_i es definida como la entropía promedio de la región sensada W_i al rededor de la localidad destino

$$U_i^I = -\sum_{C \in W_i} H(s(C)) \tag{2.3}$$

donde la entropía H es calculada para cada celda en la región W_i. Mientras mayor sea la entropía, menor información sobre la región está disponible y es más atractiva para su exploración.

Utilidad de navegación. Largos viajes entre localidades reducen la eficiencia de la exploración. La utilidad de la navegación es usada para hacer viajes cortos más atractivos. Una función de navegación encapsula el costo de alcanzar cualquier localidad en el mapa desde la posición actual (basada en la información de mapas de rejillas de ocupación). La utilidad de navegación para una localidad destino x_i, es simplemente el negativo de la función de navegación V a esa localidad.

$$U_i^N = -V(x_i) \tag{2.4}$$

Utilidad de la habilidad para localizarse. Esta utilidad es usada para distinguir entre localidades con diferente calidad de localización. La calidad de la localización en un lugar dado x_i está determinada por la incertidumbre en la configuración de un vehículo que se genera hasta llegar a ese lugar. La incertidumbre es descrita por una matriz de covarianza del estado $\mathbf{P}_{vv}(x_i, t)$ en la localidad x_i al tiempo t. La entropía H para una distribución Gaussiana en la localidad del vehículo a una localidad destino x_i es:

$$U_i^L = -H(\mathbf{P}_{vv}^k) = -\frac{1}{2}\log\left((2\pi e)^n \mid \mathbf{P}_{vv}^k \mid\right) \tag{2.5}$$

donde n es el número de estados del vehículo y $\mathbf{P_{vv}^k}$ la covarianza del estado del vehículo después de k observaciones.

La **utilidad total** de las localidades destino es la suma pesada de las utilidades individuales

$$U^{TOT} = w_I U^I + w_N U^N + w_L U^L \tag{2.6}$$

donde los pesos relativos w_I, w_N y w_L dependen de la misión y pueden ser ajustados de

forma abrupta o variar continuamente. La localidad con la utilidad mayor es seleccionada como la siguiente localidad a visitar:

$$x^* = \arg \max_x (U^{TOT}) \tag{2.7}$$

2.3.4. Enfoque de Amigoni

En [2], se presenta una estrategia de exploración basada en información que ha sido derivada usando el concepto de entropía relativa. La función usada para comparar posiciones de observación es

$$f(p) = \frac{1}{N+A} \sum_{i \in \mathcal{A} \bigcup \mathcal{N}} \ln(\frac{\sigma_{unc,i}}{\sigma}) + N \ln(\frac{\sigma}{p}) + \sum_{i \in \mathcal{A}} \ln(\frac{\sigma}{\sigma_{p,i}}) + N \ln(\frac{2\pi \times c}{\sigma}) \tag{2.8}$$

donde $N = |\mathcal{N}|$ es el número de nuevos puntos sensados (información de un telémetro láser) desde p, $A = |\mathcal{A}|$ es el número de puntos ya sensados que son sensados nuevamente desde p, $\sigma_{unc,i}$ es la desviación estándar de la contribución de la medida de error debido a la incertidumbre en la posición del robot, σ es la desviación estándar en la exactitud del sensor, P es el perímetro esperado del área a ser añadida al mapa, $\sigma_{p,i}$ es la desviación estándar previa del punto i ya sensado, y c es la distancia entre la posición del robot y p. En este caso, el valor menor de $f(\cdot)$ identifica las mejores posiciones de observación. El primer término de la Ecuación 2.8 es la contribución de la entropía de los puntos sensados desde p, el segundo término de la Ecuación 2.8 es la contribución de la entropía de los puntos sensados por primera vez desde p, el tercer término de la Ecuación 2.8 es la contribución de la entropía de los puntos ya sensados que son nuevamente sensados desde p, y el último término es la contribución de la entropía del costo de alcanzar p. En general, el primero de los cuatro términos incrementa la entropía mientras el segundo y el tercer término reducen la entropía.

Para determinar la mejor posición de observación siguiente, esta estrategia de exploración mezcla la información esperada adquirida por los sensores y la distancia viajada por el robot para alcanzar la posición. La ganancia de información es derivada de los puntos que se vuelven visibles en la posición de observación y de la reducción de incertidumbre en la localización de los puntos previamente observados. Esta estrategia de exploración está teóricamente fundamentada en Teoría de la Información.

2.3.5. Enfoque de Tovar

En [94] [91] [92] se presenta una función que es usada para evaluar la calidad de las propuestas de futuras configuraciones de sensado. Se presentan algunos atributos que deben tener las configuraciones de sensado y se describe como deben ser evaluadas por una función de utilidad. Por ejemplo, la función de evaluación debe preferir: configuraciones futuras que tengan marcas visuales con una mayor probabilidad de reconocimiento, configuraciones futuras con el mayor número de primitivas geométricas distintivas, configuraciones futuras cercanas a los límites entre lo conocido y desconocido (bordes libres), trayectorias que minimicen la incertidumbre en la localización, trayectorias que requieran del menor número de configuraciones de sensado, y trayectorias que minimicen las distancia total que el robot debe viajar.

La función de utilidad propuesta en [94] tiene una forma multiplicativa para descartar configuraciones que tengan un valor muy pequeño en alguna de las medidas. Formas similares han sido presentadas en [92] [91]. Esta función tiene la ventaja de que puede ser utilizada para una sola configuración del robot o para un camino asociado con una secuencia compuesta por varias configuraciones de sensado. La utilidad de una secuencia de configuraciones es simplemente la suma de las configuraciones sub-meta del camino recorrido.

2.3.6. Enfoque de Newman

En [70], se propone un algoritmo de exploración basado en primitivas geométricas. En este enfoque los candidatos de la siguiente localidad del robot (sub-meta) están asociados con todas las primitivas geométricas del entorno. Un objetivo en [70] es explorar localmente regiones libres. Para lograr ese objetivo, se propone el siguiente procedimiento. Primero, para cada meta generada, puntos muestra son regularmente diseminados a su alrededor a un radio constante β. Se dibuja un círculo de radio α centrado en cada muestra. El tamaño final del conjunto muestra es el número de puntos muestra para la meta que satisface las siguientes condiciones: i) cada punto debe tener una línea de vista libre al objetivo, y ii) no tiene línea de vista a cualquier otro círculo de radio α. La puntuación $\eta \in [0, 1]$ para evaluar la localización meta se fija para los tamaños inicial y final del conjunto de muestras.

2.3.7. Enfoque de Laguna

En [49] se desarrolló una estrategia de exploración para un robot de manejo diferencial con forma de disco que es representada como una Maquina de Moore. El robot tiene capacidades de sensado limitado, ya que es incapaz de medir ninguna distancia o ángulo, o realizar autolocalización. La estrategia de exploración garantiza que el robot descubrirá la región más grande posible de cualquier entorno poligonal simplemente conectado. La estructura topológica GNT presentada en [93] es usada para representar el entorno y codificar una landmark que está ubicada en el entorno.

2.4. Estrategias de exploración para múltiples robots

El uso de múltiples robots tiene muchas ventajas sobre utilizar un solo robot. La cooperación de robots permite realizar una tarea más rápido que, un solo robot.

Algunos trabajos han sido propuestos para la exploración y construcción de mapas utilizando múltiples robots [81] [18] [97]. En algunos trabajos, la ganancia de la información y el costo de la exploración son considerados simultáneamente para definir las siguientes localidades de sensado para cada robot del équipo [81] [18]. En [18], el mapa es representado usando una rejilla de ocupación. Las posibles localidades para la siguiente exploración son definidas sobre las celdas que se encuentran en el límite entre el espacio conocido y el espacio desconocido. El enfoque anterior es extendido en [84] tomando en cuenta información semántica de la localidad de sensado. En [97], los autores proponen estrategias para múltiples robots usando una segmentación del entorno para determinar el objetivo de cada robot. Esta segmentación mejora la distribución de los robots sobre el entorno. En [94], se presenta una técnica que permite que uno o múltiples robots exploren eficientemente y construyan un modelo del entorno. Se utiliza una función de utilidad para medir la calidad de las configuraciones de sensado propuestas y un algoritmo genera y selecciona configuraciones para la siguiente configuración de sensado. Este trabajo considera un equipo de robots con las mismas capacidades de sensado y movimiento. Algunos trabajos han propuesto una exploración con múltiples robots que tienen diferentes capacidades de sensado y movimiento. El trabajo presentado en [82] considera un equipo de robots con diferente tamaño. Durante la exploración, si un robot es demasiado grande para navegar entre obstáculos y alcanzar una localidad de sensado, entonces pregunta a robots más pequeños si pueden realizar la tarea. En [25], los robots tienen uno de dos roles: navegador o cartógrafo. Los navegadores

se mueven aleatoriamente en el entorno hasta que encuentran una localidad objetivo para el cartógrafo. Después el cartógrafo se mueve a la localidad objetivo. En los trabajos mencionados previamente cada robot sigue un rol específico y predefinido de acuerdo a su tipo.

El trabajo presentado en [81] trata con el problema de exploración y construcción de mapas de un entorno desconocido con múltiples robots. Una rejilla probabilística es usada para representar el entorno. Las celdas de la rejilla son de tres tipos: obstáculo (probabilidad de ocupación arriba de un umbral p_o dado), despejado (probabilidad menor a un umbral p_c) y desconocido (ya sea por no haber sido sensado o con probabilidad entre p_o y p_c). La distancia entre el robot y la frontera entre lo conocido y desconocido del entorno es usado como un costo de exploración. El número de celdas desconocidas que están dentro del rango de sensado del robot (para una localidad siguiente posible) es usado como la ganancia de información para enviar al robot a esa localidad. La utilidad de una localidad de sensado candidata es igual a la ganancia de información menos el costo de exploración. Un ejecutivo central trata de maximizar la utilidad total. Para coordinar a los robots (relacionar robots con localidades de sensado) se usa la ganancia de información.

2.5. Planificación de movimientos para navegación

En [64], se describe un método de navegación reactiva que usa una estrategia de "divide y vencerás" basada en situaciones para simplificar la dificultad de la navegación. El diseño del método es realizado a un nivel simbólico, donde se define un conjunto de situaciones y las acciones asociadas con cada situación. Durante la fase de ejecución, la percepción es usada para identificar la situación actual y se realiza la acción asociada. Además, se propone una implementación basada en geometría llamada la Navegación con el Diagrama de Proximidad (Nearness Diagram Navigation).

Algunos métodos utilizan una analogía física para calcular los comandos de movimiento donde ecuaciones matemáticas tomadas de la física son aplicadas a la información sensorial y la solución es transformada en comandos de movimiento. Por ejemplo: Métodos de campos potenciales [46] [47] [90] [45] [12] [67], la analogía del perfume [6], y la analogía del fluido [62], entre otras.

Algunos métodos calculan un conjunto de comandos de movimiento adecuados para seleccionar un comando basado en una estrategia de navegación. Algunos métodos calculan

un conjunto de ángulos de dirección [31] [13] [95] [42], y otros calculan un conjunto de comandos de velocidad [80] [33] [15] [4].

Otros métodos calculan una forma de describir la información de alto nivel de la información sensorial para obtener un comando de movimiento [45] [71] [16]. El método de la Navegación con el Diagrama de Proximidad pertenece a este grupo de enfoques, ya que algunas entidades intermedias son calculadas para seleccionar una situación concreta, y después se ejecuta una acción que calcula el movimiento.

Nuestro trabajo está relacionado con planificación de movimientos con restricciones de visibilidad [44] [63], planificación de movimientos con restricciones no holonómicas [53] [11] [83] [52] [7] y navegación basada en referencias (landmarks) [56] [74] [69] [72] [60] [41].

La planificación de movimiento con restricciones no holonómicas ha sido un campo de investigación muy activo (un buen panorama es presentado en [52]). Los resultados más importantes en este campo han sido obtenidos abordando el problema con herramientas de geometría diferencial y la teoría de control. Laumond fue pionero de esta investigación y produjo el resultado de que una trayectoria libre para un robot holonómico moviéndose entre los obstáculos en un espacio de trabajo 2D, siempre puede ser transformado en una trayectoria factible para un robot tipo automóvil (no holonómico) haciendo maniobras [53]. El trabajo en [11] usa primitivas de movimiento para construir un diagrama de trayectoria básico, similar al grafo de visibilidad [51], pero para vehículos no holonómicos.

En [28], los autores presentan un algoritmo para encontrar trayectorias factibles para un sistema no holonómico en presencia de obstáculos. Primero, el problema de planificación de la trayectoria sin presencia de obstáculo es transformado en un problema de mínimos cuadrados no lineal en un espacio aumentado. La evasión de obstáculos es incluida como restricciones de desigualdad y se presentan los resultados en simulación para el caso de un tractor con remolque.

Uno de los primeros trabajos de la navegación robótica basada en referencias es el de Lazanas [56]. La visibilidad geométrica no está explícitamente integrada; en su lugar, cada referencia define una "zona segura" circular en la cual se supone que el robot sensa y se mueve sin incertidumbre. El algoritmo polinomial completo usa encadenamiento hacia atrás de retroproyecciones omnidireccionales para calcular planes en la presencia de incertidumbre. En [14], las referencias son vistas como sub-metas a alcanzar y la planificación utiliza un marco probabilístico para calcular los caminos más cortos esperados en el grafo de referencias. En [74], ha sido propuesto un enfoque de navegación basado en referencias

que genera trayectorias de "navegación costera". En una tarea de localización robótica probabilística, una ventaja importante de un enfoque basado en referencias es que si las referencias son detectadas, éstas detienen el crecimiento incremental de la incertidumbre en la posición del robot [74] [69] [72] [41] [77].

Como se muestra en los trabajos antes mencionados, el uso de referencias para la navegación y localización se ha generalizado en robótica [63] [56] [74] [69] [41] [76] [10] [58]. En cualquiera de estos contextos, el primer requerimiento básico para usar referencias es ser capaz de percibirlas, especialmente durante la ejecución del movimiento del robot, a pesar de las limitaciones del campo de vista del sensor. En [39] [40] se estudia la integración de restricciones no holonómicas y de visibilidad usando un robot de manejo diferencial (DDR) que tiene que mantener en vista referencias estáticas en un entorno con obstáculos. Primero determinan las condiciones necesarias y suficientes para la existencia de una trayectoria de tal forma que el sistema sea capaz de mantener la visibilidad hacia una referencia dada. Después extienden este resultado al problema de planificación de trayectorias garantizando la visibilidad a través de un conjunto de referencias.

Los trabajos presentados en [7] [83] [96] pertenecen al grupo de métodos de control óptimo que son ejecutados en lazo abierto. Tales métodos podrían beneficiarse de los resultados de la investigación sobre control visual de robots móviles con ruedas [59] [8] [9], de tal forma que los planes puedan ser ejecutados en lazo cerrado.

Capítulo 3

Algoritmo de navegación para un robot puntual

Los algoritmos de construcción de representaciones topológicas que sustentan esta tesis son presentados en este capítulo.

3.1. Modelo Topológico: Gap Navigation Tree

En este capítulo describiremos brevemente el árbol de navegación basado en brechas (Gap Navigation Tree o GNT), para un robot puntual. Se presentarán los principales conceptos utilizados para su desarrollo y los principales resultados obtenidos en trabajos previos, los cuales extendemos en esta tesis para un robot con forma de disco.

El algoritmo del GNT construye una representación topológica del entorno en forma de un árbol obtenido por medio de un sensor omnidireccional (con un campo de vista de 360^0).

3.1.1. Modelo del robot

El robot es modelado como un punto que se mueve en un espacio desconocido.

Notación

R es el ambiente donde el robot se mueve.

∂R es la frontera del ambiente.

x es la posición del robot.

$G(x)$ es una secuencia de brechas detectadas por el sensor de brechas.

El sensor de brechas

El robot sólo cuenta con un sensor, el sensor de brechas, que tiene la habilidad de detectar y seguir discontinuidades en la información de profundidad. El sensor de brechas puede ser implementado con un telémetro láser o con un puntero láser y una cámara omnidireccional. Cada discontinuidad es llamada brecha y corresponde a una región no visible por el robot. Se supone que el robot puede seguir y distinguir las brechas en todo momento y almacenar cualquiera de sus cambios combinatorios.

El sensor de brechas regresa únicamente una lista ordenada cíclicamente de etiquetas asignadas de forma única a cada brecha, es decir, $G(x) = [g_1, ..., g_k]$ es la secuencia de brechas que regresa el sensor de brechas cuando el robot está en un lugar del entorno $x \in R$. Para cualquier posición del robot en el entorno, cada brecha está etiquetada de forma única y no contiene información de distancias o ángulos (vea la Figura 3.1).

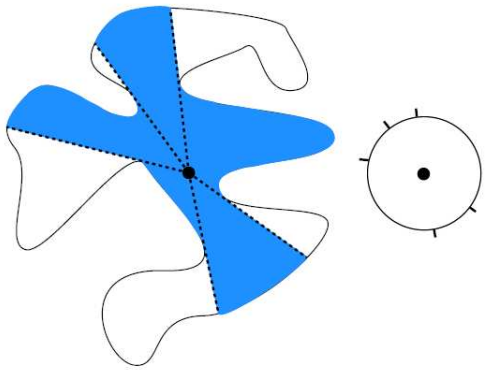


Figura 3.1: El entorno visto por el robot. La posición del robot es mostrada como un punto negro. A la izquierda, se muestra el entorno y la visibilidad del robot. A la derecha, se muestra el orden angular de las brechas detectadas en la región de visibilidad. Figura obtenida del trabajo presentado en [93].

Primitivas de movimiento

Los movimientos del robot son expresados como una secuencia de primitivas de movimiento, las cuales son descritas solamente en términos de la información del sensor. Para un brecha $g \in G(x)$, una primitiva de *cazar (moverse hacia) la brecha* es denotada por chase(g); hay que notar que no hay una referencia a x porque su posición x es desconocida para el robot. En esta primitiva de movimiento, el robot rota hasta alinearse con la brecha y después avanza con velocidad unitaria. Hay que hacer notar también que la primitiva chase(g) puede causar seguimiento de frontera. Es decir, un movimiento en donde el robot se mueve tocando la frontera de un obstáculo.

La primitiva de movimiento puede ser considerada como una acción en un enfoque jerárquico donde una observación dispara una acción y donde hay una capa de planificación y otra de control. Por lo tanto es importante especificar las condiciones bajo las cuales la primitiva de movimiento termina. La primitiva chase(g) termina cuando g desaparece.

3.1.2. Árbol de navegación basado en brechas (GNT)

La construcción del GNT empieza añadiendo n nodos hijo a la raíz de un árbol, donde n es el número de brechas que regresa el sensor de brechas en su observación inicial del ambiente. Las brechas son etiquetadas consecutivamente empezando con g_1. Para cualquier brecha nueva que aparezca, se le asigna el siguiente entero no utilizado para asegurar identificación única. Por ejemplo, si el sensor del robot inicialmente detecta tres brechas, el sensor de brechas regresaría la siguiente lista ordenada $[g_1, g_2, g_3]$ y la raíz del GNT tendría tres nodos hijo.

Eventos críticos y construcción incremental del GNT

Mientras el robot se mueve en el entorno pueden ocurrir cambios en la información del sensor de brechas debido a la ocurrencia de *eventos críticos*. Cuando ocurre un evento crítico el GNT es actualizado. Existen cuatro diferentes tipos de eventos críticos:

1. Una nueva brecha aparece: Un nuevo nodo g es añadido como hijo de la raíz, mientras se preserve el orden angular de la lista del sensor de brechas.

2. Las brechas g_1 y g_2 se unen en g: Los nodos g_1 y g_2 se vuelven hijos del nuevo nodo g, el cual es añadido como hijo de la raíz, mientras se preserva el orden cíclico.

3. Una brecha desaparece: El nodo g, el cual debe ser una hoja, es suprimido.

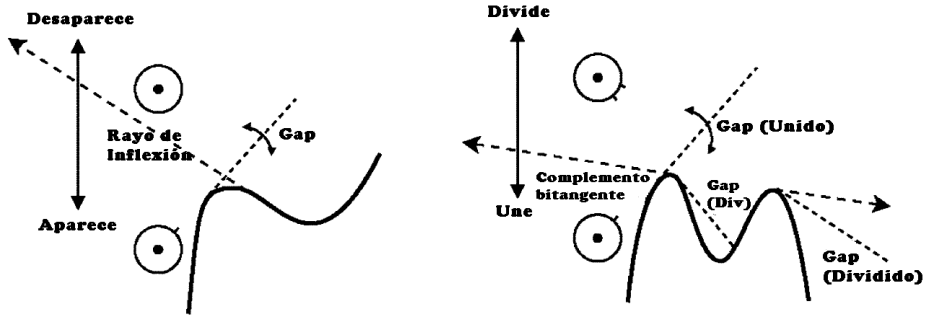
4. La brecha g se divide en g_1 y g_2. Si g es un nodo hoja, entonces g_1 y g_2 se vuelven nuevos nodos; de otra manera, ya existían como hijos de g. Ambos g_1 y g_2 son conectados a la raíz, preservando el orden angular.

Interpretación geométrica del GNT

La relación entre los eventos críticos y la geometría del entorno está determinada por los cruces del robot por inflexiones generalizadas o bi-tangentes generalizadas de la frontera del entorno ∂R [93] (como se ilustra en la Figura 3.2). La aparición o desaparición de brechas ocurre cuando el robot cruza un rayo de inflexión, y la unión o división de brechas ocurre cuando el robot cruza complementos bi-tangentes.

Lema 3.1.1 *Sean g_1 y g_2 dos brechas que se unen en la brecha g_3. Cuando g_3 se divide, g_1 y g_2 aparecen en la misma posición angular al tiempo de la unión, independientemente del movimiento del robot.*

La prueba del Lema 3.1.1 aparece en [93].



(a) (b)

Figura 3.2: Eventos críticos. (a) La aparición y desaparición de brechas ocurre cuando el robot cruza una línea de inflexión. (b) División y unión ocurren al cruzar complementos bi-tangentes. Figura obtenida del trabajo presentado en [93].

Mapa de caminos basado en brechas

Las trayectorias seguidas por el robot utilizando cualquier primitiva de movimiento después de un evento crítico corresponden a los límites de las celdas del grafo de aspectos [93]. El conjunto de puntos de R visitados por el robot al realizar las trayectorias descritas arriba es llamado el *mapa de caminos*, y es denotado por S.

Construcción de un GNT completo

Los nodos en el árbol pueden ser de dos tipos: primitivos y no primitivos. Un nodo primitivo codifica que el espacio detrás de la oclusión que lo genera, ya ha sido percibido y un nodo no primitivo codifica que aún no ha sido percibido. Inicialmente todos los nodos son no primitivos y los nodos primitivos son añadidos al árbol como resultado de la aparición de brechas (oclusión de regiones visibles previamente). Un árbol es incompleto cuando tiene nodos hoja que son no primitivos. Por lo tanto, el GNT es forzado a ser completo cazando nodos hoja no primitivos iterativamente. Cuando el nodo hoja que se está siguiendo (cazando) desaparece, entonces otro nodo hoja no primitivo es seleccionado para ser cazado. El orden en que los nodos hoja no primitivos son cazados no es importante. Eventualmente todas las hojas serán primitivas, en cuyo caso el GNT es completo.

Como un ejemplo de la construcción de un GNT completo, suponga que el robot está en el entorno mostrado en la Figura 3.3. En la Figura 3.3(a), se muestra los límites de las celdas del grafo de aspectos (líneas punteadas). La raíz del GNT es el disco negro. Los nodos que no son conocidos como primitivos son mostrados como círculos, y los nodos que son primitivos son cuadros. El robot empieza la construcción del GNT como se muestra en la Figura 3.3(a). Allí el robot primero ejecuta chase(g_1). Cuando esta brecha es seguida, al cruzar un rayo de inflexión una nueva brecha aparece, lo que dispara un evento de aparición utilizando la información de sensado, y la brecha g_3 es añadida al árbol (ver la Figura 3.3(b)). Después, las brechas g_1 y g_3 se unen, y se vuelven hijos de un nuevo nodo g_4 (ver la Figura 3.3(c)). Cuando g_1 desaparece (ver Figura 3.3(d)), g_2 es la única brecha no primitivo que queda, y el robot ejecuta **chase**(g_2), el cual genera la ejecución de la secuencia de primitivas [chase(g_4), chase(g_2)], es decir, para que el robot pueda seguir a g_2 primero debe seguir a g_4. El robot sigue (caza) g_4 hasta que se divide, y después, g_2 es seguido (ver Figura 3.3(e)). Finalmente, cuando g_2 desaparece, todos los nodos hoja son primitivos (ver Figura 3.3(f)).

Lema 3.1.2 *El procedimiento de seguir hojas no primitivas iterativamente termina con un*

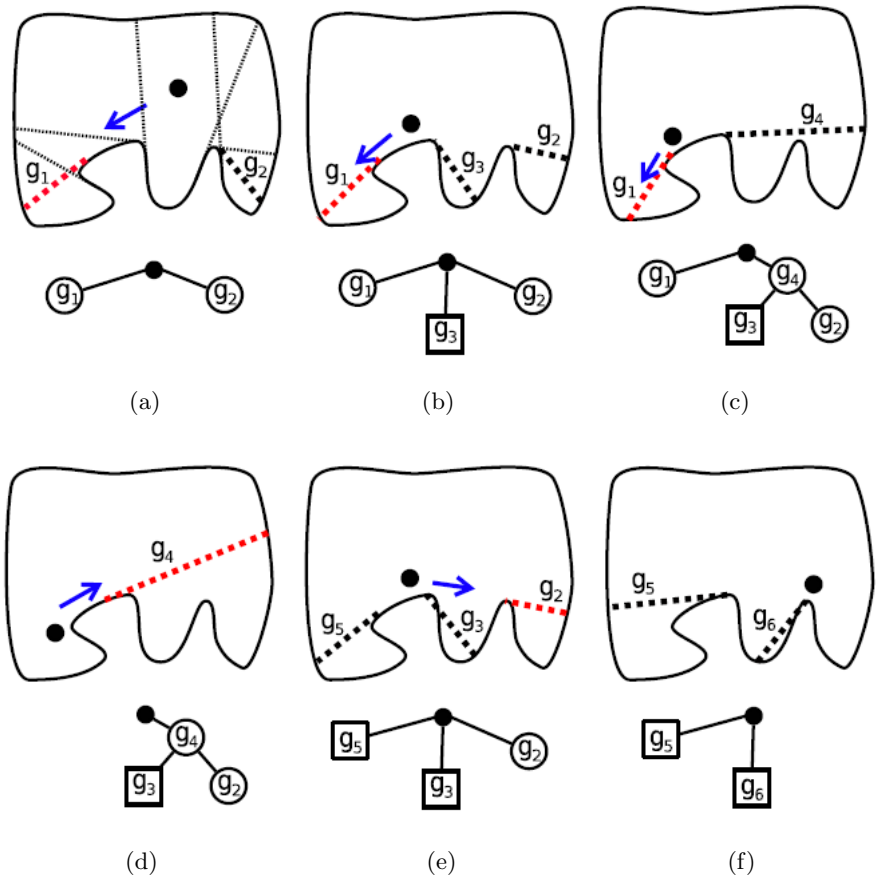


Figura 3.3: Construyendo un GNT. Figura obtenida del trabajo presentado en [93].

GNT completo como resultado.

La prueba del Lema 3.1.2 aparece en [93]. Note que incluso cuando el GNT está completo, aún así el GNT cambia cuando el robot se mueve en su entorno. Esto pasa porque el árbol siempre expresa su entorno, relativo a un marco local del robot.

3.1.3. Navegación Óptima

Dado que el GNT se construye a partir de eventos críticos, y que estos corresponden a los límites de las celdas del grafo de aspectos, puntos dentro de la misma celda tienen el mismo GNT. Una vez que el GNT es construido, las trayectorias generadas siguiendo una secuencia de brechas en el GNT son óptimas en distancia euclidiana.

Movimiento a lo largo del mapa de caminos basado en brechas

En lo siguiente, se discute sobre la optimalidad de las trayectorias generadas usando secuencias de brechas en el GNT, en términos de distancia euclidiana, usando las primitivas de movimiento chase(·).

- Sean p y q puntos en el mapa de caminos basado en brechas.
- Sea $U = [u_1, u_2, \ldots, u_n]$ la secuencia de intervalos máximamente conectados de ∂R que el robot atraviesa (en orden) en la trayectoria más corta de p a q.

Lema 3.1.3 *Sea $H = [g_1, g_2, \ldots, g_n]$ la secuencia de brechas en la cual g_i es la brecha seguida cuando el robot atraviesa el intervalo $u_i \in U$. La trayectoria generada siguiendo iterativamente la secuencia H es la trayectoria más corta entre p y q.*

Teorema 3.1.4 *Si R es simplemente conectado y el robot está en un punto en S, entonces la trayectoria codificada en el GNT entre la raíz y cualquier punto $q \in S$ es globalmente óptima en distancia euclidiana.*

La prueba del Teorema 3.1.4 y del Lema 3.1.3 aparecen en [93].

Hay que notar que la optimalidad surge únicamente de los eventos críticos, y que el robot no necesita realizar mediciones precisas de distancia. Para polígonos, algunos de los

intervalos definidos en U pueden degenerarse solamente en puntos y el mapa de caminos basado en brechas corresponde al grafo de visibilidad reducido[1].

Viajando a cualquier lugar en el entorno

El robot no está confinado al grafo de visibilidad reducido. Una meta en términos de coordenadas (es decir, (x, y)) no tiene ningún significado en el contexto del GNT. Sin embargo, se supone que el entorno del robot contiene referencias visuales (p. ej., un cuadro azul) y objetos que el robot puede mover (p. ej., una pelota roja). Con esta suposición se pueden realizar tareas como "buscar la pelota roja y llevarla a donde se encuentra el cuadro azul".

Se supone que cada objeto y referencia visual son identificados de forma única y pueden estar en cualquier lugar en R. Un objeto $o_i \in O$ se dice que es reconocido cuando el robot está en $x \in R$ si y sólo si $o_i \in V(x)$, donde $V(x)$ es la región de visibilidad del robot en la posición x (es decir, el objeto está dentro del rango de vista del sensor). El reconocimiento de referencias visuales es definido de manera similar.

Se pueden aumentar las capacidades del sensor de brechas para reconocer objetos y referencias. La primitiva chase$(\cdot)$ puede ser adaptada para realizar tareas de recuperar objetos o moverse hacia referencias. En este caso, un quinto evento crítico es incluido, el cual corresponde a la aparición de un objeto o referencia visual en $G(x)$. Por lo tanto, chase$(\cdot)$ termina si la brecha se divide o desaparece mientras es cazada, si un objeto aparece o si una referencia aparece. El robot también debe contar con la habilidad de seguir objetos o referencias visuales, lo cual se denota respectivamente como chase(o) o chase(l).

La construcción del GNT se realiza de la misma forma que en la sección 3.1.2; sin embargo, se almacena nueva información en el árbol. La definición del GNT es extendida para permitir que los objetos o referencias visuales aparezcan como nodos. Si un objeto o referencia visual es ocluido por un vértice que genera la brecha g (puede verse como una unión), entonces es añadido como hijo de g (Ver la Figura 3.4). El robot puede regresar a cualquier referencia visual l siguiendo (cazando) g hasta que l aparezca en $G(x)$. Por lo tanto, se puede definir la primitiva **chase(l)** como una secuencia de primitivas de movimiento que llevan a l. Un objeto puede ser manejado de la misma manera en el GNT, dando como resultado **chase(o)**. Los objetos difieren de las referencias en que el robot puede llevar objetos a otra parte de R para dejarlos allí. Mientras el robot se mueve, los objetos son

[1]La definición está en el Apéndice A

incorporados en el GNT en caso de que se requiera regresar al lugar donde se dejó el objeto.

Teorema 3.1.5 *El GNT extendido codifica las trayectorias de cualquier objeto o referencia visual en el entorno desde la posición actual del robot.*

La prueba del Teorema 3.1.5 aparece en [93].

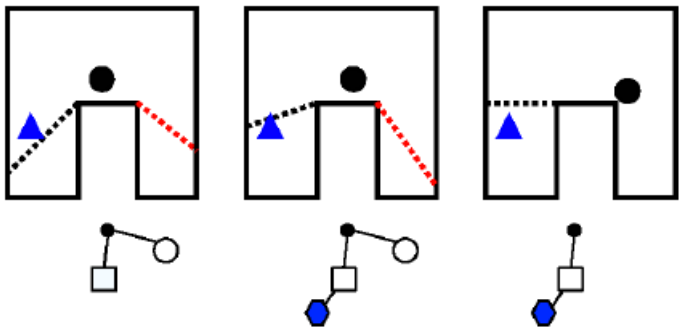


Figura 3.4: Codificando los objetos en el GNT. Cuando el objeto triangular se esconde detrás de la brecha, se asocia ese objeto con la brecha. La brecha codifica la última vez que el objeto fue visible (el objeto está escondido detrás de la brecha). Figura obtenida del trabajo presentado en [93].

3.1.4. Entornos múltiplemente conectados

En el caso de entornos múltiplemente conectados[1], ∂R tiene varios componentes. Se supone que cada componente de ∂R está delimitado y es la imagen de una curva cerrada analítica a partes. La construcción y uso del GNT fue basada en la primitiva de movimiento de seguir brechas hasta que ocurren eventos críticos. Aunque esto ofrece un control simple y limpio para el robot, no es suficiente para entornos múltiplemente conectados. Se establece formalmente este resultado negativo en el siguiente teorema.

Teorema 3.1.6 *La terminación de las primitivas de movimiento no está garantizada en entornos múltiplemente conectados.*

Además, usando sólo el sensado de brechas, la optimalidad global en entornos múltiplemente conectados no puede ser alcanzada.

[1]La definición está en el Apéndice A

Teorema 3.1.7 *La optimalidad de la trayectoria global en general no es posible usando únicamente el sensado de brechas.*

La prueba de los Teoremas 3.1.6 y 3.1.7 aparece en [93].

Nuevas suposiciones para entornos múltiplemente conectados

Algunos eventos críticos nuevos deben ser introducidos de tal manera que se garantice que una primitiva de movimiento termine. La forma de realizar esto es proveer al robot con la capacidad de reconocer una localidad vista anteriormente. Esto puede ser hecho proveyendo al robot con marcadores. Cuando el robot hace contacto por primera vez con la frontera de un componente b mientras sigue (caza) una brecha se suelta un marcador. Después, si al seguir la misma brecha, el marcador es encontrado, un evento crítico es disparado. Esto indica que la frontera del componente ha sido rodeada completamente una vez. Se le da una nueva primitiva al robot, surround(b), la cual comanda al robot para recorrer completamente la frontera del componente b una vez.

El número de marcadores necesitados depende del algoritmo de construcción del GNT en particular. El algoritmo rodea cada componente de frontera, uno por uno, guardando todos los eventos críticos basados en brechas.

En resumen, los nuevos requerimientos para entornos múltiplemente conectados son: sólo un marcador debido a que el robot volverá a encontrarlo y la habilidad de identificar cada componente de frontera (por ejemplo, por color). Así como los objetos y las referencias visuales, los componentes de frontera de ∂R son asociados a una brecha. Particularmente, el comienzo y fin de una brecha son asociados con respecto a los componentes de frontera.

Construcción del GNT para entornos múltiplemente conectados

Al comenzar la exploración, todas las hojas del GNT son marcadas como no primitivas. Para garantizar que el robot verá todo el entorno, una primitiva de rodear es ejecutada por cada componente de ∂R. El robot escoge seguir arbitrariamente un componente de frontera que no ha rodeado antes, y una vez que éste ha sido rodeado, otro nuevo componente es seleccionado.

Teorema 3.1.8 *En el GNT para entornos múltiplemente conectados las primitivas* ***chase(****l****)***

y ***chase(o)*** *resultan en movimientos localmente óptimos (hasta la clase homotópica*[1]*) en distancia a la referencia* (l) *u obstáculo* (o)*, entre cualquier par de posiciones posibles en* R*.*

La prueba del Teorema 3.1.8 aparece en [93].

3.1.5. Implementación

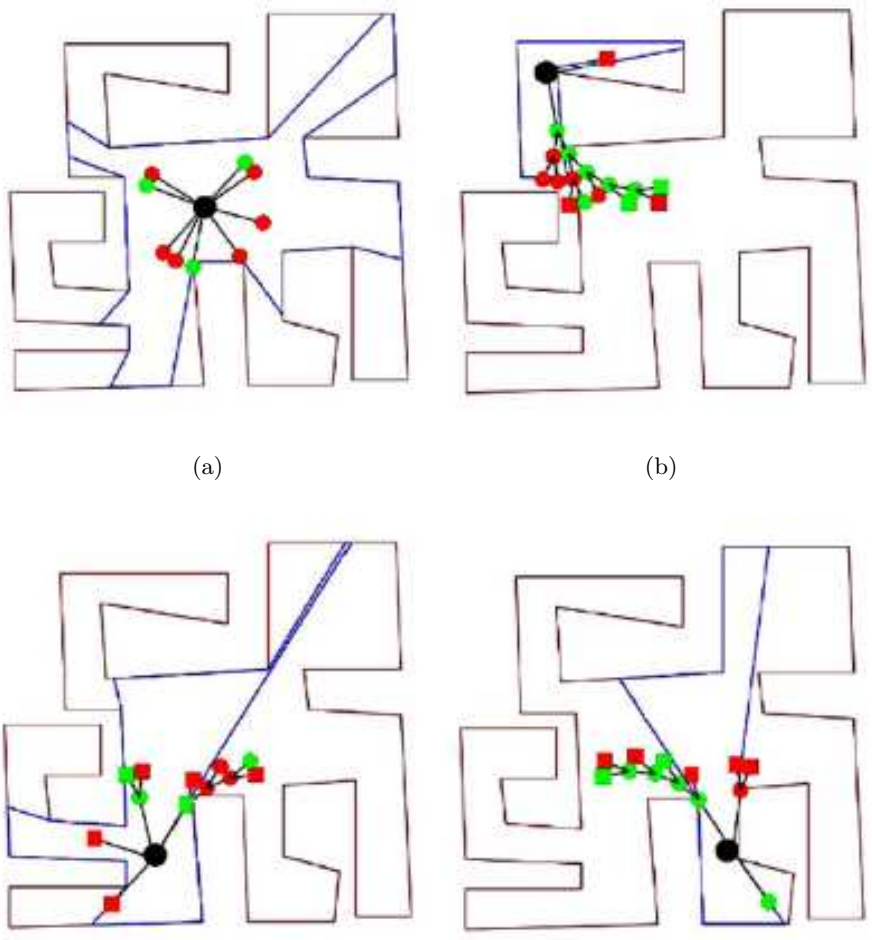
Simulación

En [93], se muestran simulaciones de los algoritmos del GNT para entornos simplemente conectados y múltiplemente conectados. La Figura 3.5 muestra una simulación de la construcción del GNT que es realizada en [93]. La posición del robot está marcada con un disco negro, el cual sirve como raíz de la representación gráfica del GNT. Los nodos primitivos se muestran como cuadros y los nodos no primitivos como círculos. Los colores oscuro (rojo) y claro (verde) de los nodos representan regiones ocluidas en el entorno a la derecha y a la izquierda respectivamente. Las ramas del GNT están alineadas a las brechas que representan, pero esto es sólo por claridad de la presentación. Hay que recordar que no se usa información angular exacta para la construcción del GNT. La posición inicial del GNT es mostrada en la Figura 3.5(a). Debido a que ninguna brecha ha sido explorada, todos los nodos del árbol son círculos (brechas no primitivas). Las Figuras 3.5(b) y 3.5(c) muestran diferentes estados del GNT, mientras brechas no primitivas son exploradas. La Figura 3.5(d) muestra el instante después de que la última brecha no primitiva ha desaparecido. Puede verse que todos los nodos hoja del GNT son cuadros.

Experimentos con un robot móvil Pioneer

En [93], se presentan experimentos en un robot Pioneer 2-DX. El robot fue equipado con dos telémetros láser para obtener un campo de vista omnidireccional. La Figura 3.6 muestra algunos instantes durante la construcción del GNT.

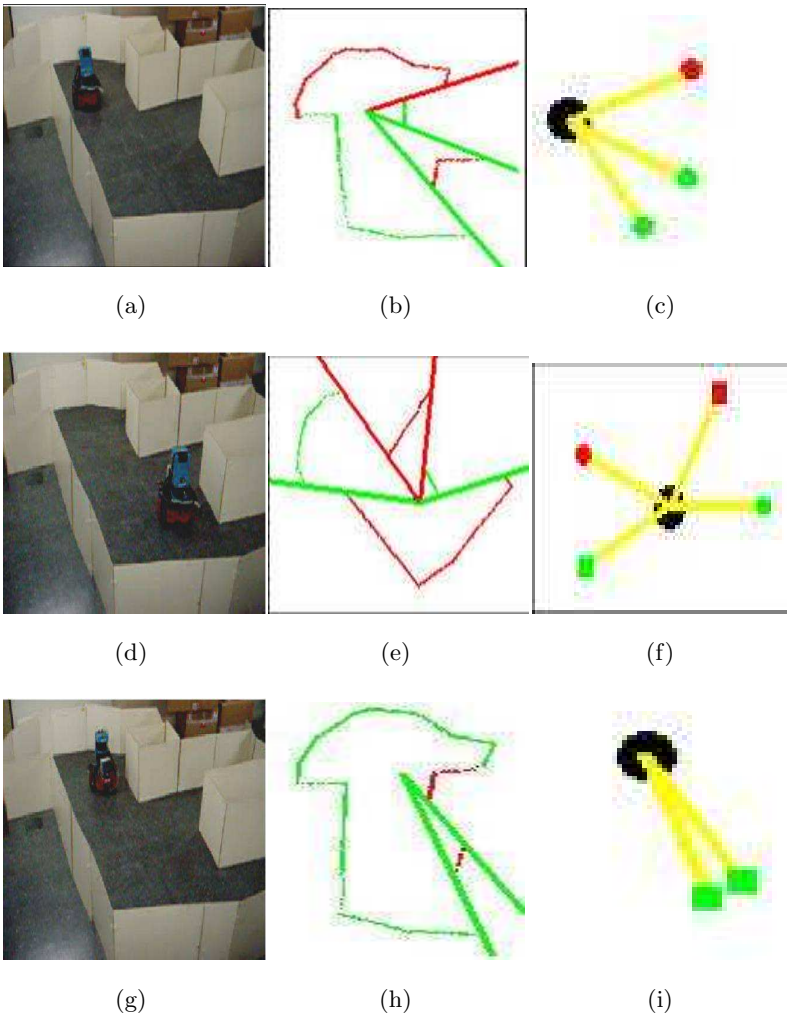
[1]La definición está en el Apéndice A



(a) (b)

(c) (d)

Figura 3.5: Simulación del GNT para un robot puntual en un ambiente simplemente conectado. Figura obtenida del trabajo presentado en [93].



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

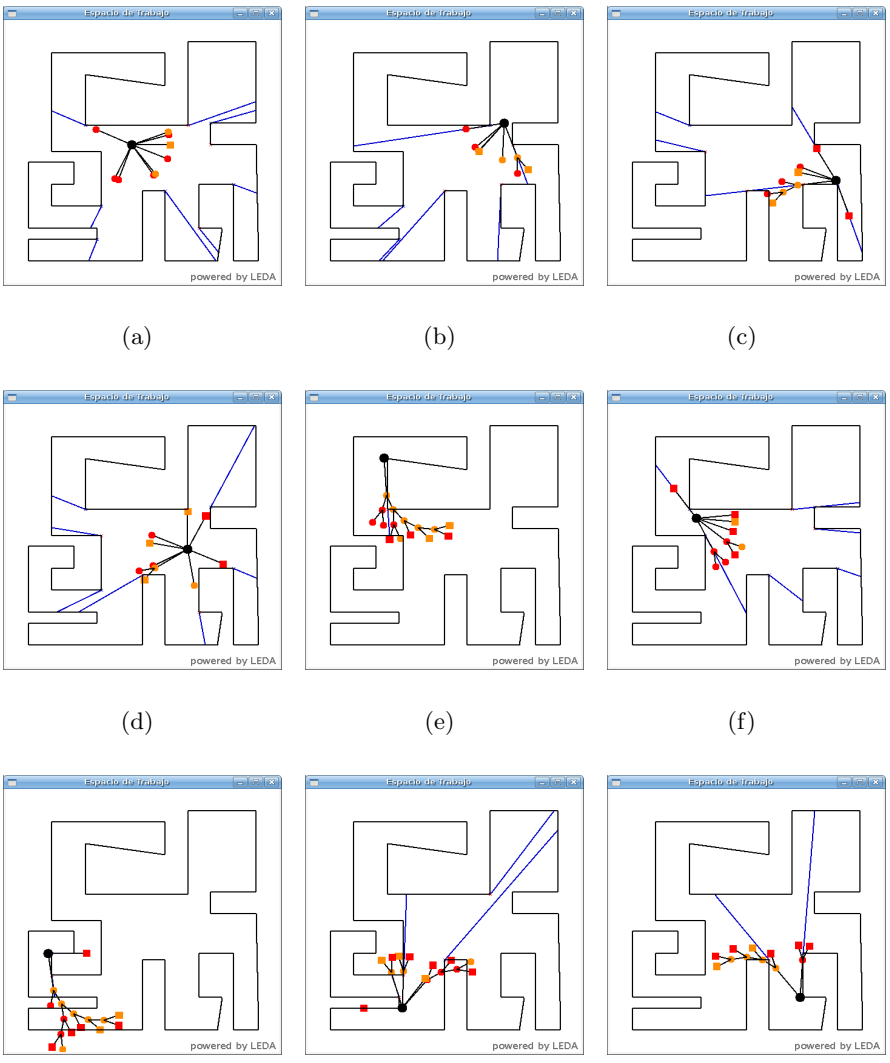
Figura 3.6: Implementación del GNT en un robot Pioneer 2-DX. Figura obtenida del trabajo presentado en [93].

3.2. Mi implementación del GNT para un robot puntual

El primer paso para extender el GNT para un disco fue implementar el GNT original para un punto. El GNT fue implementado en simulación utilizando una representación poligonal del entorno.

Se realizaron algunas pruebas para ambientes simplemente conectados como los mostrados en la Sección 3.1.5. En la Figura 3.7, se muestran varias capturas de pantalla de la simulación del robot puntual. La estructura del GNT se presenta en la Figura 3.7(e).

También se realizaron pruebas en entornos más complejos como los mostrados en la Figura 3.8. En esta figura, se muestra un entorno cuatro veces más grande que el de la Figura 3.7. En la Figura 3.8(a) se muestra al GNT poco después del inicio de la exploración. Se puede ver como una rama del GNT codifica toda la estructura de la parte superior izquierda del entorno. Al pasar de la parte inferior izquierda a la parte superior izquierda en la Figura 3.8(b) se puede ver como un árbol de dos ramas codifica lo parcialmente explorado. Casi al final de la construcción del GNT, en la Figura 3.8(c) se observa que la estructura del GNT ha crecido considerablemente. El GNT completo es mostrado en la Figura 3.8(d).

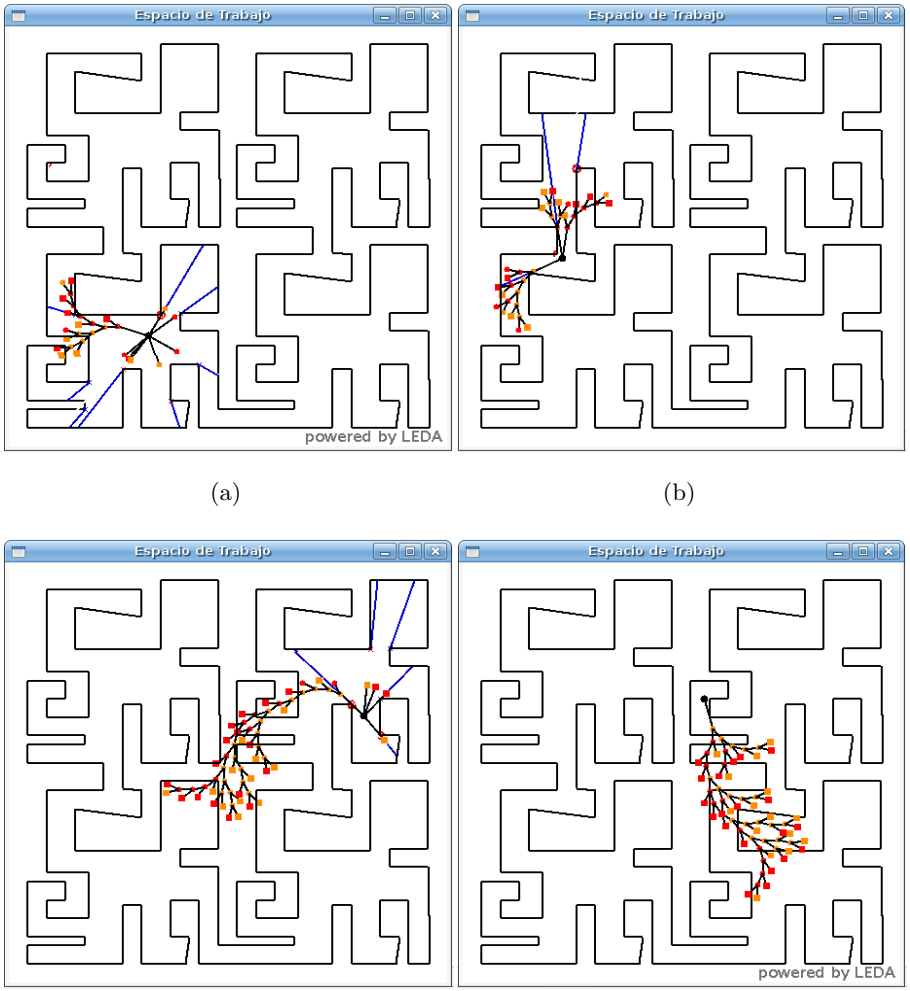


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 3.7: Mi simulación del GNT para un punto en un entorno como el mostrado en la Sección 3.1.5



(a) (b)

(c) (d)

Figura 3.8: Mi simulación del GNT para un punto en entornos más complejos

Capítulo 4

Navegación óptima para un robot de manejo diferencial con forma de disco

4.1. Introducción

Si un robot puntual es colocado dentro de una región poligonal conocida, entonces calcular las trayectorias más cortas es fácil. El enfoque más común es calcular un grafo de visibilidad que incluye solamente los bordes bitangentes, el cual es realizado por un algoritmo de barrido radial en un tiempo $O(n^2 \lg n)$ [27] (existe también un algoritmo $O(n \lg n + m)$, en el cual m es el número de bitangentes [35]). Una alternativa es el *método Dijkstra continuo*, el cual propaga combinatoriamente un frente de onda a través de la región y determina la trayectoria más corta en un tiempo $O(n \lg n)$. Existen numerosas variaciones del problema. Calcular las trayectorias más cortas en regiones poliédricas tridimensionales es NP-hard [19]. Si se permite que los costos varíen sobre las regiones, el problema se complica considerablemente [66, 73]. Vea [34, 65] para estudios de algoritmos de las trayectorias más cortas. Para esfuerzos recientes en obstáculos curvos, vea [21]. En este trabajo consideramos un robot de manejo diferencial DDR con forma de disco y suponemos que el mapa geométrico del ambiente es desconocido.

El uso de un robot puntual es poco realista en la mayoría de las aplicaciones prácticas. Por lo tanto, es interesante estudiar el caso de un robot disco, el cual puede corresponder, por ejemplo, a la plataforma Roomba. Una vez que el robot tiene dimensiones no triviales,

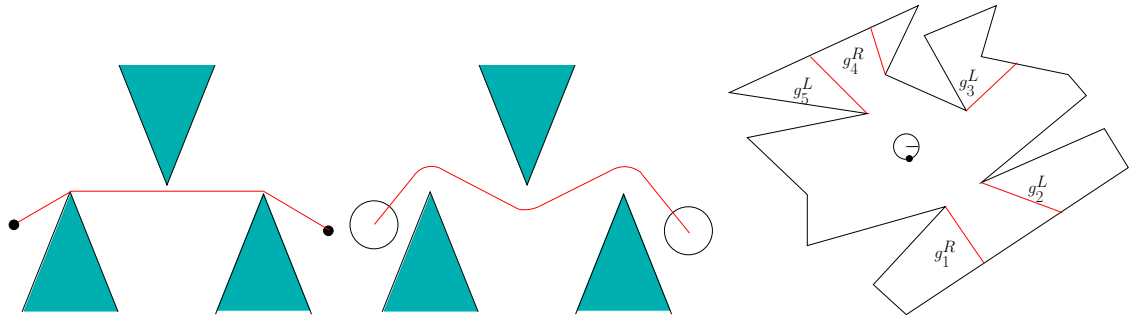


Figura 4.1: a) La trayectoria óptima para un robot puntual, b) La trayectoria óptima para un robot disco, c) El sensor de brechas (el pequeño disco solido en la frontera del robot) detecta la secuencia de brechas $G = [g_1^R, g_2^L, g_3^L, g_4^R, g_5^L]$, en la que g_1^R y g_4^R son brechas derechas (de cerca a lejos) y g_2^L, g_3^L, y g_5^L son brechas izquierdas (de lejos a cerca).

el problema puede ser expresado en términos de obstáculos en el espacio de configuraciones. En el caso del espacio de configuraciones, se presentan soluciones en [22, 57].

Ahora suponga que el mapa del entorno no le es dado al robot. En ese caso el robot debe usar sus sensores para explorar y construir el mapa del entorno para desarrollar estrategias de navegación. Dados sensores poderosos y buena odometría, enfoques estándar de SLAM [29, 88] pueden ser aplicados, para obtener un mapa que puede ser usado como entrada para los métodos previamente mencionados. Sin embargo, en este trabajo no permitimos que el robot pueda ubicarse a sí mismo con respecto a un marco de referencia *global* o construir un mapa geométrico. En lugar de ello, el robot observa el mundo usando principalmente un *sensor de brechas*, introducido en [93], el cual permite determinar las direcciones de discontinuidades en profundidad (distancia hacia la frontera del entorno) y moverse hacia cualquiera de esas direcciones. Bajo este modelo pero para un robot puntual, un filtro combinatorio llamado el árbol de navegación basado en brechas (GNT) fue introducido, de forma que codifica precisamente la parte de la trayectoria más corta en el grafo de visibilidad, que es necesario para la navegación óptima [93]. La estructura de datos aprendida corresponde exactamente al árbol de la trayectoria más corta [34] desde la ubicación del robot. Esto le permite al robot navegar hacia cualquier landmark previamente vista siguiendo la trayectoria más corta en distancia, aún cuando no puede medir distancias de forma directa. El GNT fue extendido a detección probabilística de brechas y aplicado a la exploración en [68]. El GNT fue extendido a modelos de nubes de puntos en [50]. Una familia amplia de sensores de brechas es descrito en [55].

El caso de un robot disco es importante debido a que los robots reales tienen una anchura diferente de cero. Desafortunadamente, el problema es considerablemente más retador debido a que sin información sensorial adicional, el robot puede colisionar accidentalmente con obstáculos que se asoman dentro de su región de barrido mientras se mueve a lo largo de una bitangente. Vea la Fig. 4.1 (a) y (b). En navegación, el robot debe en cambio ejecutar desvíos de la bitangente. Sensar, caracterizar y navegar óptimamente alrededor de estas obstrucciones es una de las contribuciones de esta tesis. El método para generar movimientos de navegación óptimos es modelado como una Máquina de Estados Finitos de Moore (FSM). Antes de proceder con el modelo detallado y la estrategia de movimiento, es importante tener en mente varios puntos:

1. El robot es colocado dentro de un entorno, pero no le es dada la ubicación de los obstáculos ni su propia ubicación y orientación. El robot *observa* esta información en marcos de referencia locales. El propósito es mostrar cómo la navegación óptima es sorprendentemente posible sin el ordinario SLAM. Sin embargo, suponemos que el GNT que codifica el entorno ha sido construido usando la estrategia de exploración presentada en [49].

2. Se proporciona una estrategia de navegación que guía al robot hacia cualquier landmark colocado en el entorno usando el GNT aprendido. Bajo las condiciones dadas por nuestro planteamiento del problema (vea más adelante) la estrategia de navegación propuesta es óptima y se demuestra esta afirmación.

3. Creemos que aún bajo errores de control y sensado, los cuales no son considerados en esta tesis, la estrategia de navegación propuesta estaría cercana al óptimo. Por lo tanto, puede ser útil en muchas aplicaciones prácticas para que los robots naveguen eficientemente con retroalimentación de sensores.

El contenido de este capítulo es organizado como sigue: La Sección 4.2 presenta formalmente el modelo del robot incluyendo las capacidades sensoriales y de movimiento. La Sección 4.3 presenta un vector de observación que es usado para decidir la acción del robot. La Sección 4.4 describe la máquina de estados finitos que es usada para generar la navegación óptima el robot. La Sección 4.5 presenta una política de movimiento basada en retroalimentación que mapea la observación con los comandos del robot. La Sección 4.6 discute la optimalidad de la estrategia de movimiento y la Sección 4.7 presenta la implementación en simulación.

4.2. Planteamiento del problema

El robot es un sistema de manejo diferencial (un robot con dos ruedas, donde cada rueda tiene un motor independiente) que tiene definida la dirección hacia adelante (heading). El robot tiene la forma de un disco con radio r que se mueve en un entorno plano y poligonal, el cual puede ser cualquier conjunto compacto $E \subset \mathbb{R}^2$ para los cuales el interior de E es simplemente conectado. La frontera, ∂E, de E es la imagen de una curva cerrada analítica a trozos. Sin embargo, se supone que el subconjunto libre de colisión del espacio de configuraciones del robot está simplemente conectado o puede tener varios componentes conectados.

El espacio de configuraciones obstáculo proyectado en el plano corresponde a la de un disco crecido un radio r, esto es, la frontera extendida de E lo cual es debido al radio del robot[1].

Sea Λ una landmark estática ubicada en el entorno que tiene la forma de un disco con el mismo radio que el robot. Suponemos que Λ está pintada en el suelo, por lo tanto, no tiene volumen y no produce discontinuidades en distancia (brechas). Esta suposición es hecha ya que el robot tiene como meta estacionarse sobre la landmark.

Un objetivo principal de este trabajo es que el robot navegue con una trayectoria hacia Λ que minimiza la distancia Euclidiana viajada por el centro del robot. Además, estamos interesados en conocer si tal trayectoria existe o no.

4.2.1. Modelo de sensado

El robot tiene un sensor omnidireccional, el cual es utilizado para sensar el entorno. El sensor omnidireccional es capaz de detectar y rastrear discontinuidades en la información de profundidad (brechas). Por lo tanto, sobre el sensor omnidireccional, es posible construir un detector de brechas, en lo futuro referido como el sensor de brechas.

1) Sensor de brechas: El sensor omnidireccional [55, 93] es capaz de detectar y seguir (cazar) dos tipos de discontinuidades en la información de profundidad (considerando una dirección en sentido contrario a las manecillas del reloj a lo largo de ∂E): discontinuidades de lejos a cerca y discontinuidades de cerca a lejos (vea la Fig. 4.1(c)). Sea $G = [g_1^t, ..., g_k^t]$ la secuencia circular de brechas observadas por el sensor. Usando esta notación, t representa

[1]Note que esto es el espacio de configuraciones correspondiente al de un disco expandido el radio r y no el de cuerpo rígido, debido a la simetría rotacional.

el tipo de discontinuidad, en la que $t = R$ significa una discontinuidad de cerca a lejos (la porción oculta está a la derecha), llamada una *brecha derecha*, y $t = L$ significa una discontinuidad de lejos a cerca (la porción oculta está a la izquierda), llamada una *brecha izquierda*. Por ejemplo, el sensor de brechas en la Fig. 4.1(c) detecta brechas de diferente tipo: $G = [g_1^R, g_2^L, g_3^L, g_4^R, g_5^L]$.

Usamos el Árbol de Navegación basado en Brechas (GNT) [55, 93] para representar el entorno. El GNT es una estructura de datos eficiente que cambia dinámicamente de acuerdo a algunos eventos críticos, como lo vimos en el Capitulo 3. El sensor de brechas es capaz de identificar cualquiera de los cuatro eventos críticos posibles relacionados a las brechas: una brecha aparece, desaparece, se divide y dos brechas se unen [93]. La trayectoria hacia la landmark está codificada como una rama en el GNT, la cual es una secuencia de brechas a seguir (cazar). Esta rama en el GNT tiene como nodo hoja, el nodo que codifica la landmark. Como es mencionado antes, suponemos que el GNT que codifica el entorno ya ha sido construido usando la estrategia de exploración presentada en [49]. En consecuencia, el GNT es usado para propósitos de navegación.

Colocamos el sensor de brechas sobre la frontera del robot y definimos primitivas de movimiento que *durante la navegación* mueven al robot sobre trayectorias libres de colisión que posiblemente estén en contacto con los obstáculos (moverse a lo largo de la frontera del subconjunto del espacio de configuraciones, esto es trayectorias semi-libres, es necesario para la mayoría de las trayectorias óptimas). Estas primitivas de movimiento, descritas en detalle en la Sección 4.2.3, permiten que el robot rote sobre su centro de modo que quede alineado con una brecha deseada para moverse hacia delante mientras sigue (caza) una brecha, y para seguir ∂E mientras el sensor está alineado a una brecha.

Se supone que el sensor de brechas puede moverse a dos diferentes posiciones fijas sobre la frontera del robot: los extremos del lado izquierdo y derecho con respecto a la dirección de las ruedas hacia delante. Una forma de implementar esto, es con una torreta que permite que el robot mueva el sensor de brechas de su lado izquierdo a su lado derecho y viceversa. La Fig. 4.3(c) muestra al sensor alineado hacia una brecha derecha, en la cual el sensor de brechas está en el lado derecho del robot. Para alinear al sensor hacia una brecha izquierda, el robot mueve el sensor de brechas hacia su lado izquierdo. El sensor omnidireccional es capaz de medir distancias y ángulos a los vértices que generan brechas. Sea d_u la distancia entre el sensor omnidireccional y el vértice u_i que origina la brecha g_i (en la Fig. 4.2(d) $g_i = g_0^R$).

2) Sensores laterales: Para detectar obstáculos que obstruyen al robot, nuestro método

necesita medir distancias entre los puntos extremo del lado izquierdo y derecho (hacia adelante del robot) y los obstáculos. Sean estos puntos particulares del robot el punto del lado izquierdo lp y el punto del lado derecho rp. La dirección particular tangente a la frontera del robot en el punto rp es llamada rt. La dirección particular tangente a la frontera del robot en el punto lp es llamada lt (vea la Fig. 4.2(a)). Note que basado en la distancia, las discontinuidades pueden ser detectadas. Sea d_R la distancia entre rp y los obstáculos en la dirección particular rt, y d_L la distancia entre lp y los obstáculos en la dirección particular lt (vea la Fig. 4.2(b)).

Si la dirección particular, ya sea rt o lt, está apuntando a un vértice reflejo[2] (una brecha está alineada con esta dirección), entonces ocurre una discontinuidad en la lectura del sensor en esta dirección. Sea ${d_R}^t$ la distancia desde rp hasta el punto más cercano en la frontera del entorno ∂E a lo largo de la dirección de la discontinuidad. Similarmente, ${d_L}^t$ denota la distancia desde lp. Ve la Fig. 4.2(c).

Nuestra estrategia de movimiento sólo requerirá comparación de distancias para determinar cual es mayor, en lugar de necesitar medidas precisas de dicha distancia. Esto hace que el método pueda ser robusto contra errores de sensado y de control.

3) Sensores de contacto: Nuestro enfoque requiere detectar si el robot está tocando ∂E en rp o lp.

4) Sensor de alcance de la landmark: Nuestro enfoque también requiere un sensor que detecte que el landmark ha sido alcanzado.

Las medidas de distancia entre los obstáculos y rp y lp en las direcciones rt y lt (hacia a delante), y la información de si el robot está tocando ∂E en rp o lp, puede ser obtenida con diferentes configuraciones de sensores. Por ejemplo, es posible usar dos punteros láser y dos sensores de contacto, cada uno de ellos ubicado en rp y lp. Sin embargo, para usar un menor número de sensores y facilitar la instrumentación de un sistema robótico, es posible emular ambos sensores de contacto y uno de los punteros láser, usando el sensor omnidireccional. La lectura del sensor omnidireccional en la dirección de avance del robot emula la lectura del puntero láser. Las lecturas del sensor en las direcciones perpendiculares a la dirección de avance del robot son usadas en este caso. Si el robot está tocando ∂E en el punto en el cual el robot está ubicado, entonces la lectura del sensor es cero. Si el robot está tocando ∂E en el punto diametralmente opuesto a la del sensor omnidireccional, entonces la lectura del sensor corresponderá al diámetro del robot (vea la Fig. 4.2(e)). Por lo tanto, una opción es tener

[2]Definición en el Apéndice A

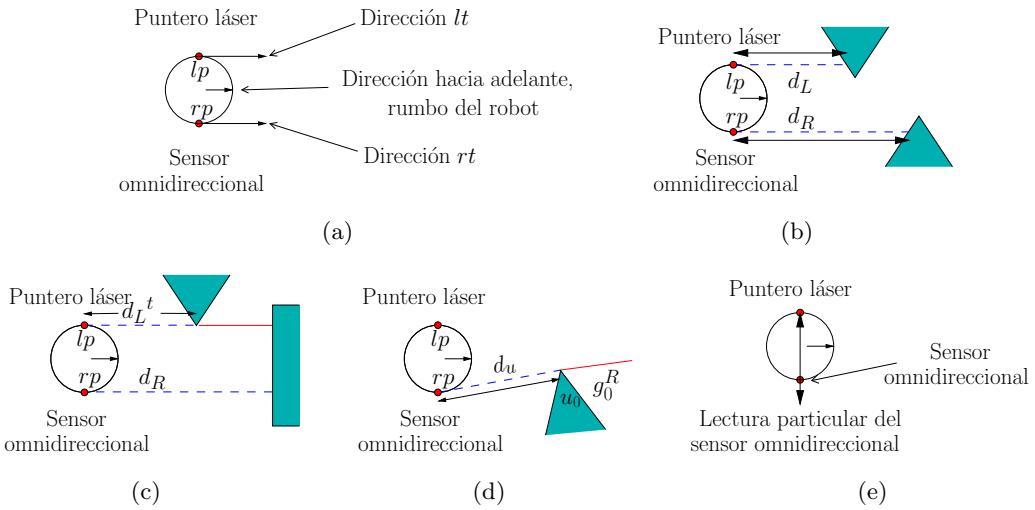


Figura 4.2: Sensores laterales: (a) Puntos rp y lp, y direcciones rt y lt, (b) d_L y d_R, (c) ${d_L}^t$, (d) d_u, (e) Lecturas del sensor omnidireccional para detectar contacto.

al robot equipado con un sensor omnidireccional y un puntero láser; éstos estarán ubicados en lp y rp. Recuerde que una torreta puede ser usada para intercambiar la ubicación del puntero láser y el sensor omnidireccional para evitar rotaciones en sitio innecesarias del robot . La torreta se mueve sobre la frontera del robot. Las brechas o landmarks siempre son cazadas con el sensor omnidireccional; el puntero láser es usado para ayudar a detectar obstáculos.

Un vector de observación es obtenido con los sensores del robot, este vector es usado para decidir la acción del robot. En la Sección 4.3 se describe en detalle este vector de observación.

4.2.2. Codificación de la landmark

En [49], se dice que el landmark es reconocido si Λ es visible, al menos parcialmente, desde la ubicación del sensor omnidireccional. Si el landmark es totalmente visible (completamente contenido en el polígono de visibilidad $V(q)$) desde la ubicación del sensor omnidireccional, entonces la landmark se codifica en el GNT como nodo hijo de la raíz. Si la landmark está totalmente o parcialmente ocluida desde la ubicación del sensor omnidireccional, entonces la landmark se codifica como nodo hijo del nodo que representa un gap en el GNT [93].

Sea rp_Λ un punto extremo sobre la landmark tal que, si rt está alineada a rp_Λ, entonces

el cuerpo del landmark está a la derecha de la dirección particular rt. Existe una definición análoga para el punto lp_Λ.

Si un vértice reflejo ocluye el punto rp_Λ, entonces la Λ es codificado con la brecha que es generada por el vértice. Del mismo modo, si un vértice reflejo ocluye el punto lp_Λ, entonces la landmark es codificada con la brecha que es generada por el vértice. Esta codificación es la codificación usada en [49], por lo tanto, la landmark Λ puede ser codificada a lo más con dos brechas. La estrategia de navegación propuesta en esta tesis, es capaz de encontrar la trayectoria que minimiza las distancia Euclidiana para alcanzar la landmark.

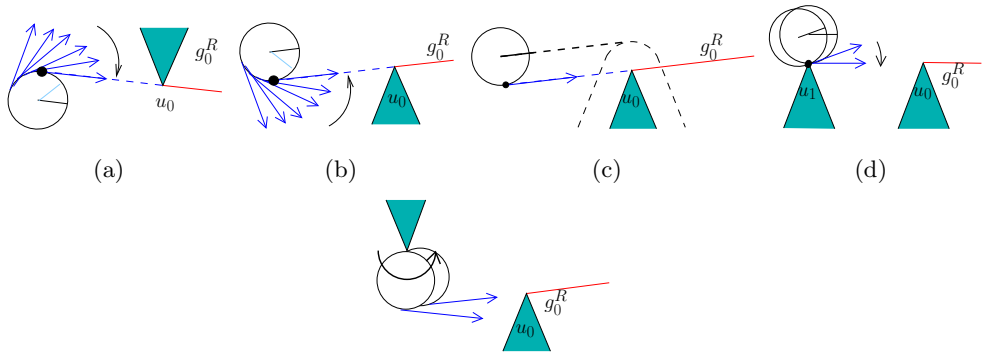
De hecho, la estrategia de exploración propuesta en [49] nos proporciona el GNT completo (cuando todo el entorno ha sido explorado), incluyendo la codificación de la landmark en él.

4.2.3. Modelo de movimiento

El robot navega usando una secuencia de primitivas de movimiento, que son generadas por un autómata para el cual las transiciones entre estados son inducidas únicamente por retroalimentación sensorial. Para navegar, se le da al robot una brecha (o equivalentemente el vértice que lo genera) o una landmark como meta. Existen cinco primitivas de movimiento (vea la Fig. 4.3). Sean w_r y w_l las velocidades angulares de las ruedas derecha e izquierda, respectivamente, con $w_r, w_l \in \{-1, 0, 1\}$. Por lo tanto, las primitivas de movimiento son generadas por los siguientes controles.

- Rotación en sitio en sentido de las manecillas del reloj: $w_r = -1, w_l = 1$.
- Rotación en sitio en sentido contrario a las manecillas del reloj: $w_r = 1, w_l = -1$.
- Rotación con respecto al punto rp en sentido de las manecillas del reloj : $w_r = 0, w_l = 1$.
- Rotación con respecto al punto lp en sentido contrario a las manecillas del reloj: $w_r = 1, w_l = 0$
- Movimiento en línea recta hacia delante: $w_r = 1, w_l = 1$.

Las primitivas de rotación son usadas para alinear rt o lt hacia una brecha (o landmark) específica. Una vez que rt o lt están alineados hacia una brecha, el robot se mueve en línea recta para cazar la brecha. Si la trayectoria de la brecha seleccionada está bloqueada,



(a) (b) (c) (d) (e)

Figura 4.3: La primitivas de movimiento: (a) Rotación en sitio en sentido de las manecillas del reloj, (b) Rotación en sitio en sentido contrario a las manecillas del reloj, (c) Movimiento en línea recta hacia delante, (d) Rotación con respecto al punto rp en sentido de las manecillas del reloj, (e) Rotación con respecto al punto lp en sentido contrario a las manecillas del reloj.

entonces el robot ejecuta un desvío seleccionando un nuevo vértice como una sub-meta. Más detalles acerca de como encontrar un vértice sub-meta son proporcionados en la Sección 4.4.

4.3. Vector de observación

El GNT codifica la trayectoria más corta hacia cualquier lugar en un entorno para un robot puntual, y también codifica la meta del robot. Sin embargo, para un robot disco esta información no es suficiente para la navegación óptima. El vector de observación proporciona las observaciones que el robot necesita para tomar decisiones acerca de las acciones que producen el movimiento óptimo. En primer lugar, se presentan algunas definiciones útiles.

Definición 4.3.1 *La brecha que codifica la trayectoria hacia la landmark en el GNT es llamada una brecha meta g_{goal}. La brecha meta es codificada como un nodo hijo de la raíz en el GNT.*

Definición 4.3.2 *Un vértice meta u_{goal} es el vértice que genera la brecha meta, un vértice meta es visible para el sensor omnidireccional.*

Definición 4.3.3 *Un vértice sub-meta es el siguiente vértice a ser visitado en la trayectoria óptima para un robot disco, cuando se requiere un desvío hacia el vértice meta.*

Definición 4.3.4 *Un vértice que genera una brecha derecha, es llamado un vértice derecho.*

Definición 4.3.5 *Un vértice que genera una brecha izquierda, es llamado un vértice izquierdo.*

También nos referiremos a un vértice candidato, el cual es un vértice que puede volverse un vértice sub-meta. Las condiciones precisas que definen a un vértice candidato están dadas en la Definición 4.4.1.

El vector de observación yn_i tiene 14 observaciones binarias. En la lista de abajo se presenta una descripción para cada observación. Algunas de ellas son más complejas y éstas son descritas con mayor detalle.

1. FI: la landmark Λ fue alcanzada (1) o no (0).
2. LV: la landmark es totalmente visible desde la ubicación del sensor omnidireccional (1) o no (0).
3. FR: el robot estuvo alineado por primera vez a un vértice sub-meta (1) de otra manera (0).
4. RP: el robot está tocando ∂E con el punto rp (1) o no (0).
5. LP: el robot está tocando ∂E con el punto lp (1) o no (0).
6. VR: los vértices que generan brechas derechas han sido ubicados sobre un marco de referencia local (1) o no (0).
7. VL: los vértices que generan brechas izquierdas han sido ubicados sobre un marco de referencia local (1) o no (0).
8. AL: el robot está alineado a un vértice dado o a una landmark (1) o no (0).
9. BL: existe un bloqueo hacia un vértice dado o hacia una landmark (1) de otra manera (0).
10. UN: la brecha meta se ha unido (1) o (0).
11. GT: el tipo de la brecha meta, derecha (1) o izquierda (0).
12. CT: el tipo del vértice candidato, derecho (1) o izquierdo (0).

13. O1: ubicación del sensor omnidireccional (dos bits son necesarios para establecer la ubicación del sensor).

14. O2: ubicación del sensor omnidireccional.

VL y VR: Para encontrar el vértice candidato, es necesario ubicar los vértices en *marcos de referencia local*. Hay cuatro tipos de marcos de referencia local. Dos de ellos son sólo el caso simétrico de los otros dos. Describimos los dos marcos de referencia local básicos en los Apéndices B.1 y B.2.

VR = 1 cuando la ubicación de los vértices derechos ha sido calculada, de otra manera VR = 0. VL = 1 cuando la ubicación de los vértices izquierdos ha sido calculada, de otra manera VL = 0.

Alineación del robot AL: El robot puede estar alineado a un vértice dado o a una landmark. El robot alinea la dirección rt hacia un vértice derecho. Simétricamente, el robot alinea la dirección lt a un vértice izquierdo. Para seguir (cazar) una landmark, el robot alinea lt con lp_Λ o rt con rp_Λ.

Bloqueo BL: Este bit es (1) si existe un bloqueo hacia un vértice dado o landmark y (0) de otra manera. Un vértice dado o landmark está bloqueado si el robot no puede viajar en línea recta hacia éste. Una definición formal para las condiciones de bloqueo es dada en la Sección 4.4.

Existen cuatro formas de detectar un bloqueo:

1. El primero ocurre cuando el robot está alineado a un vértice. Para detectar el bloqueo, las distancias d_L, d_R, ${d_L}^t$ y ${d_R}^t$ son usadas. Si la dirección rt está alineada a un vértice derecho y ${d_R}^t > d_L$ entonces la trayectoria del robot en línea recta hacia este vértice está bloqueada. Igualmente, si la dirección lt está alineada al vértice izquierdo y ${d_L}^t > d_R$ entonces la trayectoria del robot en línea recta está bloqueada.

2. La segunda forma para detectar un bloqueo es usada cuando el robot no está alineado a un vértice dado. Puede ocurrir que el robot no pueda alinear rt o lt hacia el vértice meta, debido a que este movimiento puede producir una traslación innecesaria del robot, y la optimalidad global sería perdida. En la Fig. 4.4, si el robot rota en sentido contrario a las manecillas del reloj con respecto a rp entonces el centro del robot se

moverá hacia atrás y la optimalidad se perdería. Siempre que el robot está tocando un vértice con el punto rp, para alinear rt con u_{goal} el vértice meta o landmark, el máximo ángulo medido en sentido a las manecillas del reloj desde rt hacia el vértice meta nunca puede ser mayor que π. Por lo tanto, si este ángulo es mayor que π un bloqueo es detectado.

Existe una detección de bloqueo simétrica si el robot está tocando un vértice con lp.

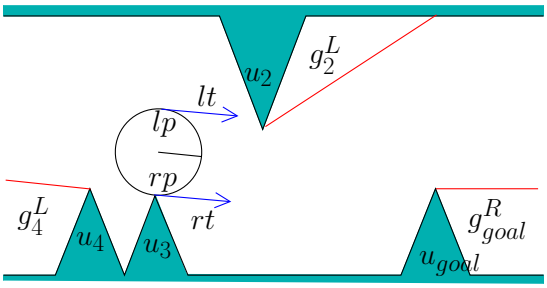


Figura 4.4: Bloqueo tipo 2.

3. La tercera forma de detectar un bloqueo, requiere varias medidas. Este tipo de detección es equivalente a cumplir las condiciones de bloqueo definidas en la Sección 4.4.

4. La cuarta forma de detectar un bloqueo es como sigue. El robot está alineado a una landmark y la landmark es totalmente visible desde la ubicación del sensor omnidireccional. La distancia desde el sensor omnidireccional hacia la landmark no puede ser sensada (dado que la landmark está pintada en el suelo y no genera discontinuidades de distancia). Consecuentemente, la trayectoria para alcanzar la landmark es declarada como bloqueada, ya que un movimiento de la torreta debe ser ejecutada para establecer que la landmark puede ser alcanzada o no sin visitar primero un vértice. Por lo tanto, un bloqueo hacia el landmark es detectado usando visibilidad. Si la landmark es totalmente visible en ambos puntos rp y lp entonces la trayectoria hacia la landmark no está bloqueada. El Caso 2-R y el Caso II-RP (vea la subsección 4.4.1) presentan los procedimientos para alineamiento con una landmark.

Ubicación del sensor omnidireccional O1 y O2: Usamos dos bits para establecer la ubicación del sensor omnidireccional.

O1=0 y O2=0 establece que el sensor omnidireccional está en lp, O1=0 y O2=1 establece que el sensor omnidireccional se está moviendo desde lp a rp, O1=1 y O2=0

establece que el sensor omnidireccional está en rp, y O1=1 y O2=1 establece que el sensor omnidireccional se está moviendo desde rp a lp.

4.4. Máquina de Estados Finitos para la Navegación Óptima

Una máquina de estados finitos (FSM) está definida como un modelo matemático de computación, es concebida como una máquina abstracta que puede estar en uno de un número finito de estados. La máquina está en sólo un estado a la vez, y puede cambiar de un estado a otro mediante la activación de eventos o condiciones llamadas transiciones. Una FSM está definida por la lista de sus estados, y las condiciones de activación para cada transición. Un tipo especial de FSM es la Máquina de Moore la cual incluye salidas asociadas con cada estado. Estas salidas dependen exclusivamente del estado y no toman en cuenta la entrada. De acuerdo a la definición presentada, es posible representar la estrategia de navegación completa como una Máquina de Moore. La FSM M representa el planificador del robot o la estrategia de navegación. M está formalmente definida como una séxtupla $(\Sigma, S, s_0, \delta, \Gamma, \omega)$, donde :

- Σ es el alfabeto de entrada (un conjunto finito no vacío de símbolos). En M, Σ está definido como ambos: el vector de observación yn_i y las consultas de entrada adicionales dadas por el GNT.
- S es un conjunto de estados finito no vacío.
- s_0 es el estado inicial, en el cual la tarea de navegación comienza.
- δ es la función de transición de estados: $\delta : S \times \Sigma \to S$. En M, dada una observación y el estado actual, δ define cual será el siguiente estado. Es importante notar que δ es una función parcial, por ejemplo, $\delta(q, x)$ no está definido para cada combinación de $q \in S$ y $x \in \Sigma$.
- Γ es el alfabeto de salida (un conjunto finito de símbolos), está definido como las señales dadas a los motores para ejecutar un control dado.
- ω es la función de salida: $\omega : S \to \Gamma$. Cada estado proporciona una salida específica definida en Γ.

La navegación consiste en visitar vértices meta. Sin embargo, el camino en línea recta hacia el vértice meta puede estar bloqueado. En tal caso un vértice sub-meta es encontrado.

El vértice que genera una brecha meta que está codificada en el GNT, siempre es visitado. Por esta razón, llamamos a la trayectoria modificada un desvío. Un desvío empieza si un vértice meta está bloqueado y termina cuando un vértice sub-meta es alcanzado.

En esta sección se proporciona una Máquina de Estados Finitos (FSM). Un objetivo principal de esta FSM es encontrar el vértice sub-meta. Una vez que se encuentra el vértice sub-meta, el robot rota para alinearse con ese vértice y después se mueve en línea recta para alcanzarlo. Para encontrar el vértice sub-meta, el robot determina vértices candidatos. La selección de un vértice sub-meta de un conjunto de vértices candidatos, es diferente para una rotación en sitio que para una rotación con respecto al punto rp o lp, esto es descrito en las siguientes subsecciones.

La definición de vértices candidatos se da a continuación. Esta definición es recursiva y hace uso del marco local $\mathcal{F}$, los espacios de búsqueda y de los ángulos θ definidos en el Apéndice B.1 (rotaciones en sitio) y B.2 (rotaciones con respecto a un punto extremo), los cuales son usados para verificar el no bloqueo hacia un vértice u. La definición cambia ligeramente dependiendo de si $\mathcal{F}$ está sobre rp o lp, y si el vértice u, el cual es usado para construir $\mathcal{F}$, es un vértice derecho o izquierdo, produciendo dos escenarios diferentes. El escenario 1 corresponde a $\mathcal{F}$ centrado en rp (o lp) y u siendo derecho (o izquierdo respectivamente), el cual aparece en ambas rotaciones en sitio y rotaciones con respecto a un punto extremo (el marco definido en el Apéndice B.1 es utilizado para ubicar vértices en el caso de rotación en sitio y el marco definido en el Apéndice B.2 es utilizado para ubicar vértices para cuando el robot rota con respecto a rp o lp). El escenario 2 corresponde a $\mathcal{F}$ centrado en rp (o lp) y u siendo izquierdo (o derecho respectivamente), el cual solamente aparece en rotaciones con respeto a un punto extremo (usando el marco definido en el Apéndice B.2). Para adecuar la correcta definición para cada escenario, solamente haga las sustituciones adecuadas en la definición de acuerdo a la Tabla 4.1.

Tabla 4.1: Etiquetas para Definición 4.4.1.

	Escenario 1	Escenario 2
$q_p = rp$ (ó $q_p = lp$)	u es derecho (ó u es izquierdo respectivamente)	u es izquierdo (ó u es derecho respectivamente)
Ξ significa	mayor	menor
Υ significa	el mayor	el menor
Use el marco local $\mathcal{F}$ construido de acuerdo a	Apéndice B.1 o Apéndice B.2	Apéndice B.2

Condiciones B (condiciones de bloqueo): Considere un marco local $\mathcal{F}$ construido correctamente (vea la Tabla 4.1) sobre q_p (rp o lp). Un vértice opuesto o (izquierdo si

u es derecho, o derecho si u es izquierdo) bloquea u en $\mathcal{F}$, si

(a) o está dentro del espacio de búsqueda de u con respecto a $\mathcal{F}$

(b) o tiene un ángulo θ_o (θ_L si o es izquierdo, o θ_R si o es derecho) que es Ξ que el ángulo θ_u relacionado a u (θ_R si o izquierdo, o θ_L si o es derecho), ambos medidos de acuerdo a $\mathcal{F}$, y

(c) o tiene una distancia ${d_o}^t$ (${d_L}^t$ si o es izquierdo, o ${d_R}^t$ si o es derecho) menor que la distancia ${d_u}^t$ (${d_R}^t$ si o es izquierdo, o ${d_L}^t$ si o es derecho) relacionado a u.

Definición 4.4.1 *Para un vértice meta dado u_{goal}, un vértice candidato c es un vértice reflejo tal que:*

(a) c es un vértice de tipo opuesto a u_{goal}, donde c satisface las condiciones de bloqueo para u_{goal}, y c tiene Υ ángulo θ generando correctamente el marco local $\mathcal{F}$ (vea la Tabla 4.1), o

(b) c es un vértice de tipo opuesto a otro vértice candidato c', donde c satisface las condiciones de bloqueo para c', y c tiene Υ ángulo θ generando correctamente el marco local $\mathcal{F}$ (vea la Tabla 4.1).

En esta sección, describimos las procedimientos para encontrar un vértice sub-meta y para alinear al robot a ese vértice. Los procedimientos descritos en las subsecciones 4.4.1 y 4.4.3 son parte de la FSM. La FSM completa es presentada en la subsección 4.4.6.

4.4.1. Rotación en sitio: Align

Align es un procedimiento que está organizado en casos. Cada caso corresponde a un conjunto de estados en la FSM.

En el procedimiento Align, una rotación en sitio es ejecutada para (1) alinear al robot hacia un vértice meta no bloqueado, (2) alinearlo a una landmark, o (3) alinearlo a un vértice sub-meta; un alineamiento a un vértice sub-meta puede requerir uno o más alineamientos a vértices candidatos.

La Fig. 4.5 muestra el diseño de la parte de la FSM que trata con los tres casos mencionados arriba, a su vez cada uno de ellos es instanciado en un caso izquierdo y un

caso derecho. Por lo tanto, esta parte de la FSM está organizada en tres casos derechos y tres casos izquierdos. Note que en la máquina los casos están relacionados, esto significa que mientras el procedimiento para encontrar el vértice sub-meta se lleva a cabo, el caso en curso puede cambiar. Abajo, tres de los seis casos son descritos. Los tres casos corresponden respectivamente a un alineamiento hacia un vértice meta derecho, hacia rp_Λ, y hacia un vértice candidato izquierdo. Los otros tres casos son sólo simétricos.

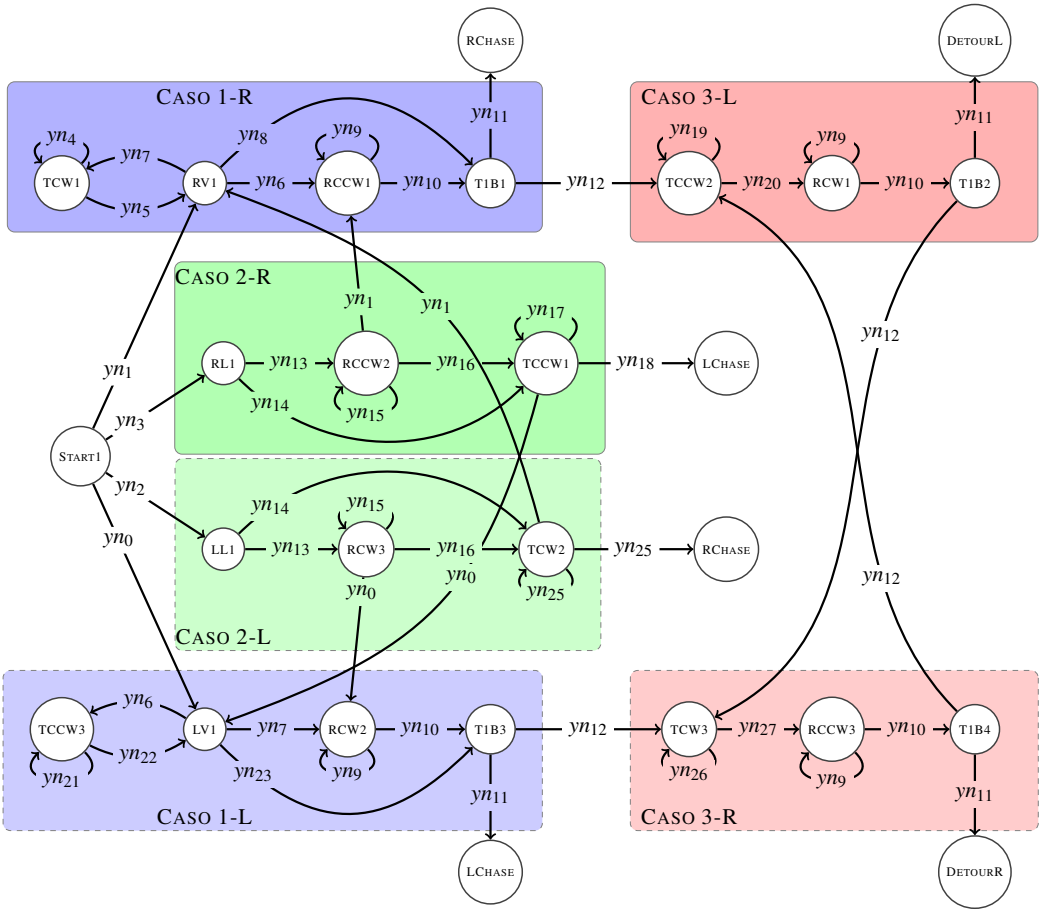


Figura 4.5: Procedimiento ALIGN.

El estado START1 es el estado inicial de la FSM completa, decide si el robot debe alinearse a un vértice meta izquierdo, a un vértice meta derecho, a lp_Λ o a rp_Λ.

Los vectores de observación que hacen que este estado transite a otro son mostrados en la Tabla 4.2. En todas las tablas siguientes, "1" denota verdadero, "0" falso y "x" cualquier valor.

Tabla 4.2: Los vectores de observación para el estado START1.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_0	x	0	x	x	x	x	x	x	x	x	0	x	x	x
yn_1	x	0	x	x	x	x	x	x	x	x	1	x	x	x
yn_2	x	1	x	x	x	x	x	x	x	x	0	x	0	x
yn_3	x	1	x	x	x	x	x	x	x	x	0	x	1	x

Caso 1-R: alineamiento a un vértice meta derecho

El robot ejecuta una rotación en sitio para quedar alineado con el vértice meta. Si la trayectoria no está bloqueada entonces el robot decide que una trayectoria óptima hacia el vértice meta es una línea recta. De otra manera, es necesario un desvío y la máquina transita al Caso 3-L.

Este caso es mostrado en azul en la Fig. 4.5. En este caso existen 4 estados:

1. RV1 indica que la tarea del robot es alinear rt hacia el vértice meta derecho.
2. TCW1 coloca al sensor omnidireccional en el punto rp usando la torreta.
3. RCCW1 hace rotar al robot para alinear rt hacia un vértice meta derecho.
4. T1B1 decide si el vértice meta derecho está bloqueado o no (detección de bloqueo tipo 1).

Los vectores de observación de entrada que desencadenan una transición entre estados en el Caso 1-R se muestran en la Tabla 4.3.

Tabla 4.3: Vectores de observación para el Caso 1-R.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_4	x	x	x	x	x	x	x	x	x	x	x	x	x	1
yn_5	x	0	x	x	x	x	x	x	x	x	x	x	1	0
yn_6	x	x	x	x	x	x	x	0	x	x	x	x	1	x
yn_7	x	x	x	x	x	x	x	0	x	x	x	x	0	x
yn_8	x	x	x	x	x	x	x	1	x	x	x	x	1	x
yn_9	x	x	x	x	x	x	x	0	x	x	x	x	x	x
yn_{10}	x	x	x	x	x	x	x	1	x	x	x	x	x	x
yn_{11}	x	x	x	x	x	x	x	x	0	x	x	x	x	x
yn_{12}	x	x	x	x	x	x	x	x	1	x	x	x	x	x

Caso 2-R: alineamiento a rp_Λ

El robot ejecuta una rotación en sitio para quedar alineado con una landmark. Si durante la rotación en sitio la landmark es por lo menos parcialmente ocluida por un vértice,

entonces la máquina transita al Caso 1-R. De otra manera, una vez que el robot está alineado con la landmark, la torreta mueve el sensor omnidireccional al lado opuesto del robot. Si durante el movimiento de la torreta la landmark es al menos parcialmente ocluida por un vértice, entonces la máquina transita al Caso 1-L. De otra manera, la landmark es alcanzable por un movimiento en línea recta.

El Caso 2-R es mostrado en verde en la Fig. 4.5. Existen tres estados en este caso:

1. RL1 indica que la tarea dada al robot es alinear rt a rp_Λ.
2. RCCW2 hace que el robot rote para alinear rt a rp_Λ.
3. TCCW1 coloca al sensor omnidireccional en el punto lp.

Los vectores de información de entrada en el Caso 2-R son mostrados en la Tabla 4.4.

Tabla 4.4: Vectores de observación para el Caso 2-R.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_0	x	0	x	x	x	x	x	x	x	x	0	x	x	x
yn_1	x	0	x	x	x	x	x	x	x	x	1	x	x	x
yn_{13}	x	x	x	x	x	x	x	0	1	x	x	x	x	x
yn_{14}	x	x	x	x	x	x	x	1	1	x	x	x	x	x
yn_{15}	x	1	x	x	x	x	x	0	x	x	x	x	x	x
yn_{16}	x	1	x	x	x	x	x	1	x	x	x	x	x	x
yn_{17}	x	1	x	x	x	x	x	x	x	x	x	x	1	1
yn_{18}	x	1	x	x	x	x	x	1	x	x	x	x	0	0

Caso 3-L: alineamiento a un vértice candidato izquierdo

El objetivo principal de este caso es encontrar un vértice candidato izquierdo y decidir si ese vértice es el vértice sub-meta o no. La ubicación de los vértices depende de un marco de referencia local. Para una rotación en sitio, los vértices son ubicados en el marco de referencia local $\mathcal{F}$ definido en el Apéndice B.1. Basado en la ubicación de los vértices sobre $\mathcal{F}$, se calcula la distancia ${d_L}^t$. ${d_L}^t$ se obtiene suponiendo que la dirección particular lt está apuntando a un vértice izquierdo. El ángulo θ_L, es el ángulo que el robot necesita para rotar (en sentido de las manecillas del reloj) para alinear lt con un vértice izquierdo. Este ángulo también es calculado. Basado en las distancias ${d_L}^t$ y ángulos θ_L para cada vértice izquierdo en el espacio de búsqueda presentado en el Apéndice B.1, se encuentra un vértice candidato usando la definición 4.4.1.

En este Caso 3-L, en primer lugar, se encuentra un vértice candidato y el robot rota en sitio para quedar alineado con este vértice candidato. Una vez que el robot está alineado

a este vértice, si el vértice está bloqueado entonces el robot busca un nuevo vértice candidato (Caso 3-R). Los dos pasos anteriores se repiten hasta que el vértice candidato no esté bloqueado, y este vértice candidato no bloqueado se vuelve el vértice sub-meta.

El Caso 3-L es mostrado en rojo en la Fig. 4.5. Existen tres estados en este caso:

1. TCCW2 coloca el sensor omnidireccional en el punto lp, ubica los vértices izquierdos y encuentra un vértice candidato.
2. RCW1 hace que el robot rote para alinear lt a un vértice candidato izquierdo.
3. T1B2 decide si el vértice candidato izquierdo está bloqueado (detección de bloqueo tipo 1) o no.

Los vectores de observación en el Caso 3-L se muestran en la Tabla 4.5.

Tabla 4.5: Vectores de observación del Caso 3-L.

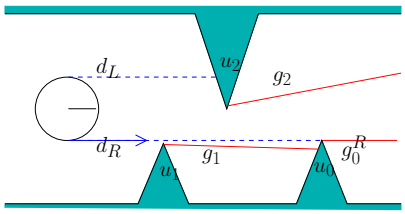
	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{19}	x	x	x	x	x	1	0	0	0	x	x	0	1	1
yn_{20}	x	x	x	x	x	x	1	x	x	x	x	0	0	0
yn_{9}	x	x	x	x	x	x	x	0	x	x	x	x	x	x
yn_{10}	x	x	x	x	x	x	x	1	x	x	x	x	x	x
yn_{11}	x	x	x	x	x	x	x	x	0	x	x	x	x	x
yn_{12}	x	x	x	x	x	x	x	x	1	x	x	x	x	x

4.4.2. Ejemplo de la ejecución de ALIGN

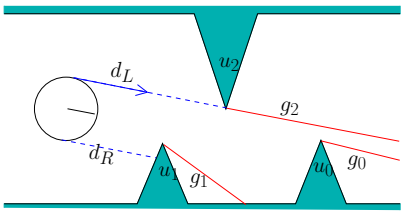
La Fig. 4.6 muestra un ejemplo de la ejecución del Caso 1-R, Caso 3-L (vértice candidato izquierdo) y Caso 3-R (vértice candidato derecho) en ALIGN para encontrar un vértice sub-meta. Estos casos ocurren secuencialmente uno después del otro.

La Fig. 4.6(a) muestra el alineamiento de rt hacia el vértice meta u_0 correspondiente a la ejecución del Caso 1-R. Primero, el vértice meta u_0 es un vértice derecho, por lo tanto la FSM transita al estado RV1. Segundo, la FSM transita al estado RCCW1 para alinear rt al vértice meta derecho u_0. Tercero, una vez que el robot está alineado, la FSM transita a T1B1. Este estado decide si el vértice meta derecho u_0 está bloqueado o no. Ya que este vértice meta está bloqueado la FSM transita al estado TCCW2, el primer estado en el Caso 3-L.

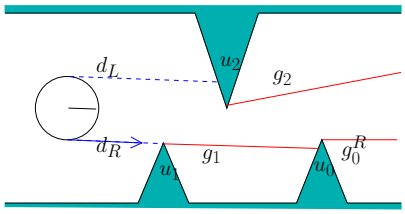
La Fig. 4.6(b) muestra el alineamiento de lt al vértice candidato izquierdo u_2 correspondiente a la ejecución del Caso 3-L (vértice candidato izquierdo). Primero, en el



(a) Caso 1-R.



(b) Caso 3-L (vértice candidato izquierdo).



(c) Caso 3-R (vértice candidato derecho).

Figura 4.6: Un ejemplo de la ejecución del robot del Caso 1-R, Caso 3-L (vértice candidato izquierdo) y Caso 3-R (vértice candidato derecho) en ALIGN.

estado TCCW2 el robot mueve el sensor omnidireccional al punto lp. Una vez que el sensor omnidireccional está en lp el robot ubica los vértices izquierdos (en el marco de referencia local $\mathcal{F}$ definido en el Apéndice B.1) y selecciona a u_2 como el vértice candidato izquierdo actual. Segundo, ya que lt no está alineado al vértice candidato izquierdo u_2, la FSM transita a RCW1. En este estado, el robot rota para alinear lt al vértice candidato u_2 (vea la Fig. 4.6(b)). Tercero, una vez que el robot está alineado, la FSM transita a T1B2. Este estado decide si el vértice candidato izquierdo u_2 está bloqueado o no. Dado que el vértice candidato izquierdo u_2 está bloqueado, la FSM transita al estado TCW3. Éste es el primer estado en el Caso 3-R (vértice candidato derecho).

La Fig. 4.6(c) muestra el alineamiento de rt al vértice candidato derecho u_1 correspondiente a la ejecución del Caso 3-R (vértice candidato derecho). En este caso, mientras el estado es TCW3, la FSM selecciona a u_1 como un vértice candidato, luego la máquina transita al estado RCCW3 y después a T1B4. El estado T1B4 decide que el vértice candidato derecho u_1 no está bloqueado. Por lo tanto, el vértice candidato derecho u_1 se vuelve el vértice sub-meta derecho.

4.4.3. Rotación con respecto al punto rp: RPAlign

En esta subsección, presentamos un procedimiento que hace que el robot rote con respecto a un punto. Por simplicidad, solamente presentamos el procedimiento RPAlign que hace que el robot rote con respecto al punto rp. Existe un procedimiento equivalente llamado LPAlign que es simétrico a RPAlign y hace que el robot rote con respecto al punto rp. De nuevo, el objetivo principal es encontrar el vértice sub-meta. RPAlign también está organizado en casos. Cada caso corresponde a un conjunto de estados en la FSM.

Una rotación con respecto a rp es ejecutada para (I) alinear al robot a un vértice meta derecho no bloqueado, (II) alinearlo a una landmark, (III) alinearlo a un vértice meta izquierdo no bloqueado, o (IV) alinearlo a un vértice sub-meta.

Durante una rotación en sito, el centro del robot no se traslada, y el robot puede alinear rt a cualquier vértice derecho, o lt a cualquier vértice izquierdo sin el riesgo de perder la optimalidad, en términos de la distancia viajada por el centro del robot. En contraste, durante una rotación con respecto al punto rp el centro del robot se traslada. Consecuentemente, el robot no puede alinear lt a cada vértice izquierdo, o rt a cada vértice derecho, exhaustivamente uno por uno para verificar si ese vértice está bloqueado o no. Esto es debido a que el centro del robot puede trasladarse innecesariamente y la optimalidad puede perderse. Para lidiar con este problema, el Algoritmo 4.4.4.1 es usado. Por lo tanto, en el Caso IV, un vértice sub-meta es encontrado con el Algoritmo 4.4.4.1 (descrito más abajo). Este algoritmo garantiza que el robot es capaz de rotar para alinearse con el vértice sub-meta sin perder la optimalidad en la trayectoria.

Puede pasar que la brecha generada por un vértice sub-meta se una con otra brecha, mientras el robot rota para quedar alineado con el vértice sub-meta. Si esto ocurre, entonces el vértice sub-meta es recalculado ejecutando de nuevo el Algoritmo 4.4.4.1.

Existe otro problema, existen vértices que no generan brecha desde la ubicación del sensor omnidireccional cuando el Algoritmo 4.4.4.1 es invocado. Tales vértices pueden bloquear el camino hacia el vértice meta o la landmark. Llamamos a este problema un vértice oculto (vea los casos III, IV y el Lema 4.6.4 para más detalles acerca de un vértice oculto). En el Lema 4.6.4 se muestra que un vértice oculto siempre eventualmente generará una brecha. Una vez que un vértice oculto es detectado, se invoca Algoritmo 4.4.4.1 y se recalcula el vértice sub-meta.

La Fig. 4.7 muestra el diseño de la parte de la FSM que trata con los cuatro casos

antes mencionados. De nuevo, los casos están relacionados, ya que mientras el procedimiento para encontrar el vértice sub-meta se lleva a cabo, el caso actual puede cambiar. Los casos simétricos corresponden a LPAlign.

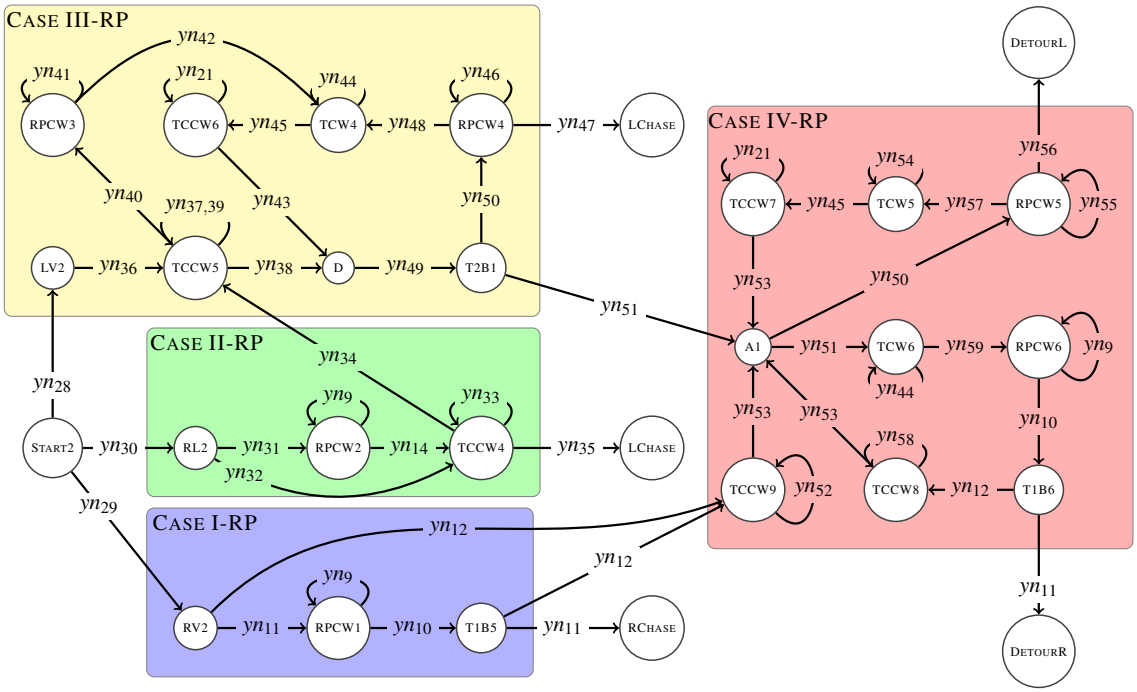


Figura 4.7: Procedimiento RPAlign.

El estado Start2 es el estado inicial para el procedimiento RPAlign. Decide si el robot debe alinearse con un vértice meta izquierdo, un vértice meta derecho o al punto extremo de la landmark rp_Λ. Los vectores de observación que hacen que la máquina transite entre estados son mostrados en la Tabla 4.6.

Tabla 4.6: Vectores de observación para el estado Start2.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{28}	x	0	x	x	x	0	0	x	0	x	0	0	x	x
yn_{29}	x	0	x	x	x	0	0	x	x	x	1	0	x	x
yn_{30}	x	1	x	x	x	0	0	x	x	x	0	0	x	x

Caso I-RP: alineamiento a un vértice meta derecho

El robot decide si puede alinearse a un vértice meta derecho sin perder optimalidad en la trayectoria o no.

Esta decisión es hecha usando el ángulo entre la dirección particular rt y el vértice

meta. Este ángulo es medido desde rt a la dirección del vértice meta en sentido de las manecillas del reloj. Si ese ángulo es menor que π, entonces el alineamiento es posible. Si el alineamiento es posible, el robot rota para quedar alineado con el vértice meta. Una vez que el robot está alineado con el vértice meta, el robot decide si la trayectoria hacia el vértice meta está bloqueada o no. La decisión es hecha usando las distancias ${d_R}^t$ y d_L. Si ${d_R}^t \leq d_L$, entonces la trayectoria no está bloqueada. Si el alineamiento no es posible (detección de bloqueo tipo 2) o ${d_R}^t > d_L$ (detección de bloqueo tipo 1) la FSM transita a el Caso IV, de otra manera el vértice meta no está bloqueado y es alcanzable por el robot con un movimiento en línea recta.

Este caso es mostrado en azul en la Fig. 4.7. En este caso existen tres estados:

1. RV2 indica que la tarea del robot es alinear rt al vértice meta derecho, y decide si existe un bloqueo (detección de tipo 2) o no.
2. RPCW1 hace que el robot rote con respecto a rp para alinear rt al vértice meta derecho.
3. T1B5 decide si el vértice meta derecho está bloqueado (detección de bloqueo tipo 1) o no.

Los vectores de observación de entrada que desencadenan una transición entre dos estados en el Caso I-RP son mostrados en la Tabla 4.7.

Tabla 4.7: Vectores de observación para el Caso I-RP.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{11}	x	x	x	x	x	x	x	x	0	x	x	x	x	x
yn_{12}	x	x	x	x	x	x	x	x	1	x	x	x	x	x
yn_9	x	x	x	x	x	x	x	0	x	x	x	x	x	x
yn_{10}	x	x	x	x	x	x	x	1	x	x	x	x	x	x

Caso II-RP: alineación del robot a un landmark

El robot decide si puede alinearse con la landmark sin perder optimalidad de la trayectoria o no. Esta decisión es hecha usando el ángulo entre la dirección particular rt y el punto rp_Λ sobre la landmark. El ángulo es medido desde rt a la dirección del punto rp_Λ en sentido de las manecillas del reloj. Si el ángulo es menor que π entonces el alineamiento es posible, de otra manera existe un bloqueo (detección de tipo 2). Si el alineamiento no es posible la FSM transita al Caso III-RP. De otra manera, una vez que el robot está alineado con la landmark, la torreta mueve el sensor omnidireccional al lado opuesto del robot. Si

durante el movimiento de la torreta la landmark queda al menos parcialmente ocluida por un vértice (detección de bloqueo tipo 4) entonces la máquina transita al Caso III-RP. De otra manera, la landmark es alcanzable realizando un movimiento en línea recta, no existe bloqueo.

El caso II-RP es mostrado en verde en la Fig. 4.7. Existen 3 estados en este caso:

1. RL2 indica que la tarea dada al robot es alinear rt a rp_Λ, y decide si existe un bloqueo (detección de tipo 2) o no.
2. RPCW2 hace que el robot rote con respecto al punto rp para alinear rt a rp_Λ.
3. TCCW4 coloca al sensor omnidireccional en el punto lp, el movimiento es realizado para detectar el bloqueo (detección de tipo 4).

Los vectores de observación en el Caso II-RP son presentados en la Tabla 4.8.

Tabla 4.8: Vectores de observación para el Caso II.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{31}	x	x	x	x	x	1	x	x	0	x	x	x	x	x
yn_{32}	x	x	x	x	x	1	x	x	1	x	x	x	x	x
yn_{9}	x	x	x	x	x	x	x	0	x	x	x	x	x	x
yn_{14}	x	x	x	x	x	x	x	1	1	x	x	x	x	x
yn_{33}	x	x	x	x	x	x	x	x	0	x	x	x	1	1
yn_{34}	x	0	x	x	x	x	x	x	0	x	0	x	x	x
yn_{35}	x	1	x	x	x	x	x	1	0	x	x	x	0	0

Caso III-RP: alineación del robot a un vértice meta izquierdo

El robot decide si el vértice meta izquierdo está bloqueado (detección de bloqueo tipo 3) o no. Para tomar esta decisión, los vértices son ubicados en un marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2. Usando la ubicación de los vértices, las distancias y ángulos de alineamiento: ${d_R}^t$, ${d_L}^t$, θ_R y θ_L son calculados. Basados en esas distancias y ángulos la decisión es tomada.

Un vértice derecho bloquea a un vértice meta izquierdo si tiene un ángulo θ_R menor que el ángulo θ_L relacionado con el vértice meta izquierdo, y una distancia ${d_R}^t$ menor que la distancia ${d_L}^t$ relacionada a ese vértice.

Si el vértice no está bloqueado, el robot rota con respecto a rp para quedar alineado con el vértice. De otra manera, la FSM transita al Caso IV-RP.

Este caso también considera la siguiente complicación. Puede pasar que durante el movimiento de la torreta (estado TCCW5), el vértice meta izquierdo se una con una brecha derecha, esta unión produce una brecha derecha. Vea la Fig. 4.8. El vértice que genera esta brecha derecha es el vértice que el robot está tocando. Para solucionar este problema el robot rota con respecto a rp hasta que la brecha derecha se divida.

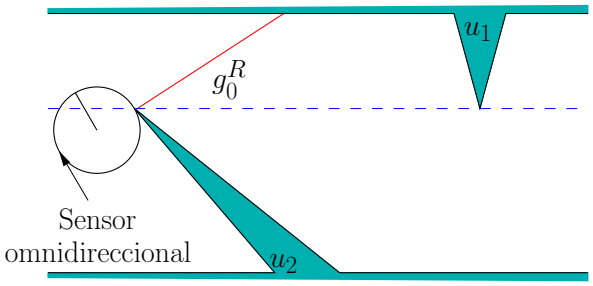


Figura 4.8: Una brecha meta izquierda (generada por u_1) se une con una brecha derecha (generada por u_2).

El Caso III-RP es mostrado en amarillo en la Fig. 4.7. Existen 8 estados en este caso:

1. LV2 indica que la tarea del robot es alinear lt al vértice meta izquierdo, y en este estado los vértices derechos son ubicados en el marco de referencia local presentado en el Apéndice B.2.
2. TCCW5 coloca al sensor omnidireccional en el punto lp.
3. RPCW3 hace que el robot rote con respecto al punto rp hasta que la brecha derecha se divida.
4. D ubica los vértice izquierdos en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2.
5. T2B1 decide si existe un bloqueo (detección de tipo 3) hacia el vértice meta o no.
6. RPCW4 hace que el robot rote con respecto a rp para alinear lt al vértice meta izquierdo.
7. TCW4 coloca al sensor omnidireccional en el punto rp, y en este estado los vértices derechos son ubicados en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2.
8. TCCW6 coloca al sensor omnidireccional en el punto lp.

Los vectores de observación de entrada en el Caso III-RP son presentados en la Tabla 4.9.

Tabla 4.9: Vectores de observación para el Caso III-RP.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{36}	x	x	x	x	x	1	x	x	x	x	x	x	x	x
yn_{37}	x	x	x	x	x	x	x	x	x	x	0	x	1	1
yn_{38}	x	x	x	x	x	x	x	x	x	x	0	x	0	0
yn_{39}	x	x	x	x	x	x	x	x	x	1	1	x	1	1
yn_{40}	x	x	x	x	x	x	x	x	x	1	1	x	0	0
yn_{41}	x	x	x	x	x	0	0	x	x	x	1	0	x	x
yn_{42}	x	x	x	x	x	0	0	x	x	x	0	0	x	x
yn_{21}	x	x	x	x	x	x	x	x	x	x	x	x	1	1
yn_{43}	x	x	x	x	x	x	x	x	x	x	x	x	0	0
yn_{44}	x	x	x	x	x	x	x	x	x	x	x	x	0	1
yn_{45}	x	x	x	x	x	1	x	x	x	x	x	x	1	0
yn_{46}	x	x	x	x	x	0	0	0	x	0	x	0	x	x
yn_{47}	x	x	x	x	x	0	0	1	x	0	x	0	x	x
yn_{48}	x	x	x	x	x	0	0	0	x	1	x	0	x	x
yn_{49}	x	x	x	x	x	x	1	x	x	x	x	x	x	x
yn_{50}	x	x	x	x	x	x	x	x	0	0	x	0	x	x
yn_{51}	x	x	x	x	x	x	x	x	0	0	x	1	x	x

Caso IV-RP: alineación del robot a un vértice sub-meta

En este caso la trayectoria hacia el vértice meta o la landmark está bloqueada. Para resolver este caso, el robot encuentra un vértice sub-meta, rota para alinearse con este vértice y después viaja en una línea recta hacia el vértice sub-meta.

Primero, se calcula la ubicación de vértices izquierdos y derechos, usando el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2. El Algoritmo 4.4.4.1 es usado para encontrar un vértice sub-meta.

Si el vértice sub-meta es izquierdo entonces el robot rota con respecto a rp para alinear lt al vértice sub-meta izquierdo. Si la brecha generada por el vértice se une con otra brecha (generada por un vértice oculto) entonces el vértice sub-meta es recalculado usando el Algoritmo 4.4.4.1.

Si el vértice sub-meta es derecho entonces el robot rota con respecto a rp para alinear rt al vértice sub-meta derecho. El robot siempre puede alinear rt con el vértice sub-meta derecho, sin perder la optimalidad global. Una vez que el robot está alineado con el vértice sub-meta derecho, el robot decide si la trayectoria hacia el vértice sub-meta está bloqueada o no. Esta decisión es tomada usando las distancias ${d_R}^t$ y d_L. Si ${d_R}^t < d_L$ entonces la trayectoria no está bloqueada. Es importante notar que la única manera de que el Algoritmo

4.4.4.1 no detecte que un vértice izquierdo bloquea la trayectoria hacia el vértice meta derecho es que el vértice izquierdo no genere brecha cuando el Algoritmo 4.4.4.1 es ejecutado (el vértice izquierdo es un vértice oculto). Sin embargo, una vez que el robot está alineado con el vértice sub-meta derecho, un vértice izquierdo que bloquea la trayectoria hacia el vértice sub-meta debe generar una brecha izquierda (vea la Subsección 4.6.2, y el Lema 4.6.3). Por lo que si la trayectoria hacia el vértice sub-meta derecho está bloqueada, entonces el vértice sub-meta es recalculado otra vez usando el Algoritmo 4.4.4.1. Si la trayectoria hacia el vértice sub-meta derecho está bloqueada, entonces el robot siempre es capaz de rotar en el sentido de las manecillas del reloj para quedar alineado con el vértice que genera el bloqueo.

El Caso IV-RP se muestra en la Fig. 4.7. Existen 9 estados en este caso:

1. TCCW9: en este estado, primero los vértices derechos son ubicados en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2. Segundo, el sensor omnidireccional es colocado en el punto lp, y finalmente los vértices izquierdos son ubicados en el mismo marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2.

2. A1: este estado ejecuta el Algoritmo 4.4.4.1 y selecciona un vértice sub-meta.

3. RPCW5 hace que el robot rote con respecto al punto rp para alinear lt al vértice sub-meta izquierdo.

4. TCW5 coloca al sensor omnidireccional en el punto rp, y en este estado los vértices derechos son ubicados en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2.

5. TCCW7 coloca al sensor omnidireccional en el punto lp, y en este estado los vértices izquierdos son ubicados en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2.

6. TCW6 coloca al sensor omnidireccional en el punto rp.

7. RPCW6 hace que el robot rote con respecto al punto rp para alinear rt al vértice sub-meta derecho.

8. T1B6: este estado decide si existe un bloqueo (detección de tipo 1) hacia el vértice sub-meta derecho o no.

9. TCCW8: en este estado, primero, los vértices derechos son ubicados en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2. Segundo, el sensor omnidireccional se coloca en el punto lp, y finalmente los vértices izquierdos se ubican en el mismo marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2.

Los vectores de observación de entrada en el Caso IV-RP son presentados en la Tabla 4.10.

Tabla 4.10: Vectores de observación del Caso IV-RP.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{50}	x	x	x	x	x	x	x	x	0	0	x	0	x	x
yn_{51}	x	x	x	x	x	x	x	x	0	0	x	1	x	x
yn_{52}	x	x	x	x	x	1	x	0	x	x	x	x	1	1
yn_{53}	x	x	x	x	x	x	1	x	x	x	x	x	0	0
yn_{21}	x	x	x	x	x	x	x	x	x	x	x	x	1	1
yn_{54}	x	x	x	x	x	0	0	x	x	x	x	0	0	1
yn_{45}	x	x	x	x	x	1	x	x	x	x	x	x	1	0
yn_{55}	x	x	x	x	x	x	x	0	x	0	x	x	x	x
yn_{56}	x	x	x	x	x	x	x	1	x	0	x	x	x	x
yn_{57}	x	x	x	x	x	x	x	0	x	1	x	x	x	x
yn_{58}	x	x	x	x	x	1	0	0	x	x	x	0	1	1
yn_{44}	x	x	x	x	x	x	x	x	x	x	x	x	0	1
yn_{59}	x	x	x	x	x	x	x	x	x	x	x	x	1	0
yn_{9}	x	x	x	x	x	x	x	0	x	x	x	x	x	x
yn_{10}	x	x	x	x	x	x	x	1	x	x	x	x	x	x
yn_{11}	x	x	x	x	x	x	x	x	0	x	x	x	x	x
yn_{12}	x	x	x	x	x	x	x	x	1	x	x	x	x	x

4.4.4. Algoritmo para encontrar un vértice sub-meta

Este algoritmo sólo es usado en el Caso IV-RP en RPAlign para encontrar un vértice sub-meta (o en el Caso IV-LP en LPAlign). Llamamos u_p a un vértice sub-meta izquierdo y u_n a un vértice sub-meta derecho. Para encontrar los vértices u_p ó u_n todos los vértices son ubicados en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2. Usando la ubicación de los vértices, las distancias y ángulos de alineamiento: ${d_R}^t$, ${d_L}^t$, θ_R y θ_L son calculados.

El Algoritmo 4.4.4.1 usa dos ordenes en los vértices. El primer orden es establecido con respecto a distancias. Este orden incluye distancias ${d_R}^t$ para vértices derechos y distancias ${d_L}^t$ para vértices izquierdos. El orden está definido de menor a mayor distancia. El segundo orden es un orden angular, también de menor a mayor; los vértices son ordenados por

ángulo incluyendo ambos θ_R y θ_L, el ángulo θ_R es usado para considerar vértices derechos y θ_L es usado para considerar vértices izquierdos. Los vértices que el Algoritmo 4.4.4.1 debe considerar para encontrar un vértice sub-meta están ubicados en el espacio de búsqueda presentado en el Apéndice B.2.

Algorithm 4.4.4.1 Encontrar un vértice sub-meta en RPAlign o LPAling.

1) El algoritmo comienza con el vértice meta. El vértice meta es declarado un vértice candidato.

2) Si el vértice candidato es un vértice izquierdo, entonces ir al Paso 6.

3) Detectar los vértices izquierdos que bloquean la trayectoria hacia un vértice candidato derecho u_c^R. Para bloquear la trayectoria hacia u_c^R, los vértices izquierdos deben tener un ángulo θ_L mayor que el ángulo θ_R relacionado a u_c^R, y una distancia ${d_L}^t$ menor que la distancia ${d_R}^t$ relacionada a u_c^R.

4) Si ningún vértice bloquea la trayectoria hacia el vértice candidato u_c^R, entonces ir al Paso 9.

5) Selección de un vértice candidato izquierdo u_c^L.
El vértice izquierdo con el mayor θ_L, el último en el orden angular es seleccionado como u_c^L.

6) Detectar los vértices derechos que bloquean la trayectoria hacia un vértice candidato izquierdo u_c^L. Para bloquear la trayectoria hacia u_c^L, los vértices derechos deben tener un ángulo θ_R menor que el ángulo θ_L relacionado a u_c^L, una distancia ${d_R}^t$ menor que la distancia ${d_L}^t$ relacionada a u_c^L.

7) Si ningún vértice bloquea la trayectoria hacia el vértice candidato u_c^L, entonces ir al Paso 9.

8) Selección de un vértice candidato derecho u_c^R. El vértice derecho con el menor ángulo θ_R, el primero en el orden angular es seleccionado como nuevo vértice candidato u_c^R. Ir al Paso 3.

9) El vértice derecho u_c^R es seleccionado como un vértice sub-meta u_n o el vértice izquierdo u_c^L es seleccionado como un vértice sub-meta u_p.

La Tabla 4.11 muestra un ejemplo de la ejecución del Algoritmo 4.4.4.1 y la selección de un vértice u_p. Este ejemplo se muestra en la Fig. 4.9. En la Tabla 4.11, $\uparrow$ indica el vértice candidato actual sobre el cual las condiciones de bloqueo son verificadas, $\times$ indica los vértices cuya distancia es menor que la distancia hacia el vértice candidato, $\otimes$ indica el vértice seleccionado como el siguiente candidato en cada iteración, $-$ indica que la distancia hacia este vértice es menor que la distancia hacia el vértice candidato, $+$ indica que la

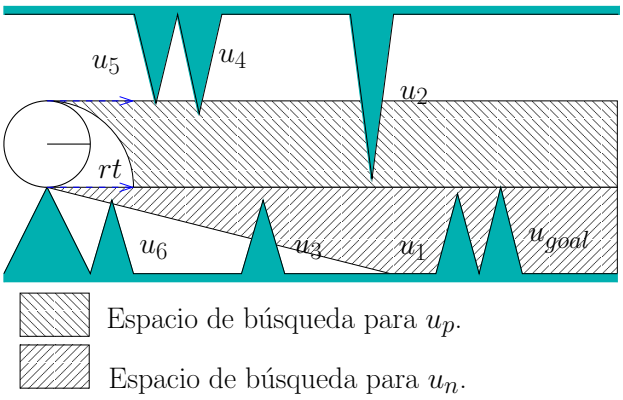


Figura 4.9: Búsqueda de un vértice u_p, u_4 en este ejemplo.

distancia a este vértice es mayor que la distancia a el vértice candidato, $\rightarrow$ indica que para un vértice izquierdo, el vértice que debe ser seleccionado como el siguiente candidato es el último en el orden angular, y $\leftarrow$ indica que para un vértice derecho, el vértice que debe ser seleccionado como el siguiente candidato es el primero en el orden angular. El algoritmo determina que u_4 es un vértice u_p.

Tabla 4.11: Ejemplo de ordenes para seleccionar un vértice u_p (refiérase a la Fig. 4.9).

Orden angular								Orden de distancias							
Índice	1	2	3	4	5	6	7	Índice	1	2	3	4	5	6	7
Dirección	rt	rt	lt	rt	lt	rt	lt	Dirección	rt	lt	lt	rt	lt	rt	rt
Tipo	R	R	L	R	L	R	L	Tipo	R	L	L	R	L	R	R
Vértice	u_{goal}	u_1	u_5	u_3	u_4	u_6	u_2	Vértice	u_6	u_5	u_4	u_3	u_2	u_1	u_{goal}
$\rightarrow$	$\uparrow$		$\times$		$\times$		$\otimes$			-	-		-		$\uparrow$
$\leftarrow$				$\otimes$		$\times$	$\uparrow$		-			-	$\uparrow$	$+$	$+$
$\rightarrow$			$\times$	$\uparrow$	$\otimes$					-	-	$\uparrow$	$+$		
$\leftarrow$					$\uparrow$	$\times$			-		$\uparrow$	$+$		$+$	$+$

La Tabla 4.12 muestra otro ejemplo de la ejecución del Algoritmo 4.4.4.1 y la selección de un vértice u_n, u_2 es un u_n. El ejemplo correspondiente se muestra en la Fig. 4.10.

4.4.5. Ejemplo de la ejecución de RPAlign

La Fig. 4.11 muestra un ejemplo de la ejecución del robot del Caso I-RP (vértice meta derecho) y el Caso IV-RP (vértice sub-meta izquierdo) en RPAlign. Estos casos ocurren

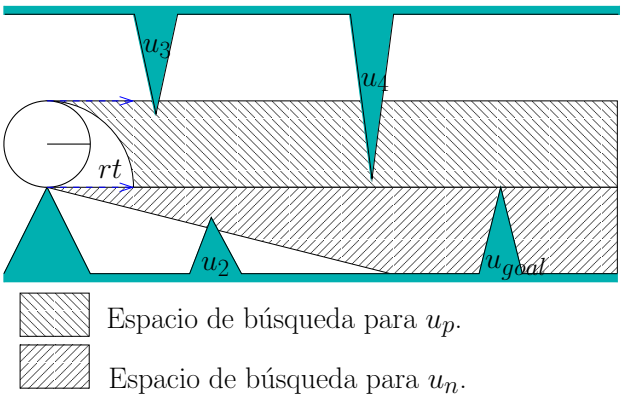


Figura 4.10: Búsqueda de un vértice u_n, u_2 en este ejemplo.

Tabla 4.12: Ordenes para seleccionar un vértice u_n (refiérase a la Fig. 4.10).

Orden angular					Orden de distancias				
Índice	1	2	3	4	Índice	1	2	3	4
Dirección	rt	lt	rt	lt	Dirección	lt	rt	lt	rt
Tipo	R	L	R	L	Tipo	L	R	L	R
Vértice	u_{goal}	u_3	u_2	u_4	Vértice	u_3	u_2	u_4	u_{goal}
$\rightarrow$	$\uparrow$	$\times$		$\otimes$		-		-	$\uparrow$
$\leftarrow$			$\otimes$	$\uparrow$			-	$\uparrow$	$+$
$\rightarrow$		$\times$	$\uparrow$			-	$\uparrow$	$+$	

secuencialmente uno después del otro.

La Fig. 4.11(a) muestra la ejecución del Caso I-RP. Primero, el vértice meta u_0 es un vértice derecho, por consiguiente la FSM transita al estado RV2. Segundo, la FSM transita al estado RPCW1 para alinear rt al vértice meta derecho u_0. Tercero, una vez que el robot está alineado (vea la Fig. 4.11(b)), la FSM transita a T1B5. Este estado decide si el vértice meta derecho está bloqueado (detección de bloqueo tipo 1). Ya que este vértice meta está bloqueado la FSM transita al estado TCCW9, el primer estado en el Caso IV-RP.

En el estado TCCW9, primero, el robot ubica los vértices derechos en el marco de referencia local $\mathcal{F}$ presentado en el Apéndice B.2. Segundo, el robot mueve al sensor omnidireccional al punto lp. Tercero, una vez que el sensor omnidireccional está en lp, los vértices izquierdos son ubicados en el marco de referencia presentado en el Apéndice B.2. La máquina transita al estado A1. En este estado el vértice sub-meta izquierdo u_2 es seleccionado usando el Algoritmo 4.4.4.1. Una vez que el vértice sub-meta izquierdo u_2 es

seleccionado, la FSM transita al estado RPCW5. En este estado el robot rota con respecto al punto rp para tratar de alinear lt hacia u_2 (Vea la Fig. 4.11(b)). Durante la rotación, la brecha generada por el vértice u_2 se une con la brecha generada por el vértice u_3. Refiérase a la Fig. 4.11(c), el sensor omnidireccional cruza la línea bitangente entre los vértices u_2 y u_3. Cuando la unión entre las brechas ocurre, la FSM transita al estado TCW5. En el estado TCW5, el robot se detiene y la torreta coloca al sensor omnidireccional en rp. Los vértices derechos son ubicados en el marco de referencia $\mathcal{F}$ definido en el Apéndice B.2. Una vez que los vértices derechos son ubicados en el marco de referencia local, la FSM transita al estado TCCW7. En el estado TCCW7, el sensor omnidireccional es colocado en lp, y los vértices izquierdos son ubicados en el marco de referencia. La FSM transita al estado A1. En este estado el vértice u_3 es seleccionado como vertice sub-meta usando el Algoritmo 4.4.4.1. Una vez que el vértice sub-meta izquierdo es seleccionado, la FSM transita al estado RPCW5. En este estado el robot rota con respecto al punto rp para alinear lt con u_3 (Vea la Fig. 4.11(d)). Ya que u_3 no está bloqueado, no es necesaria una nueva invocación del Algoritmo 4.4.4.1 y u_3 se mantiene como vértice sub-meta.

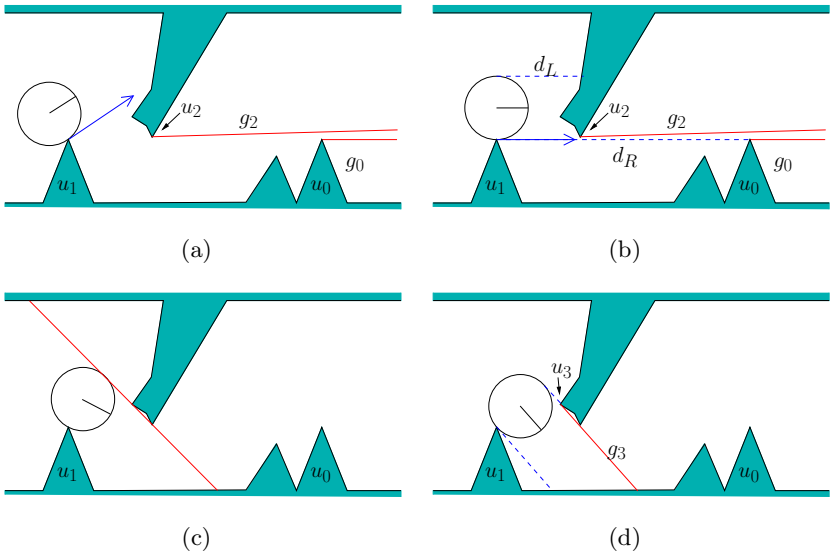


Figura 4.11: El Caso I-RP (vértice meta derecho u_0) y el Caso IV-RP (vértices sub-meta izquierdos u_2 y u_3) en RPAlign.

4.4.6. La FSM completa para la navegación óptima

La Fig. 4.12 muestra la FSM M para navegación. Para modularidad y simplicidad M está organizada en 3 procedimientos y 5 estados. Note que los procedimiento están

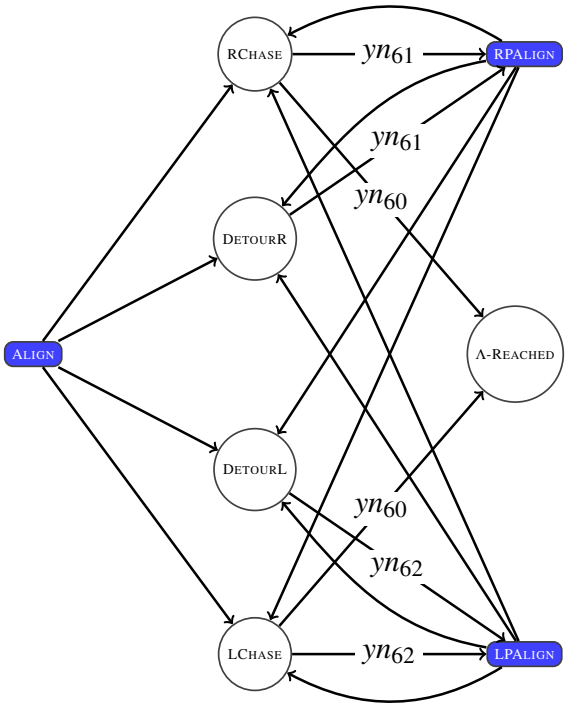
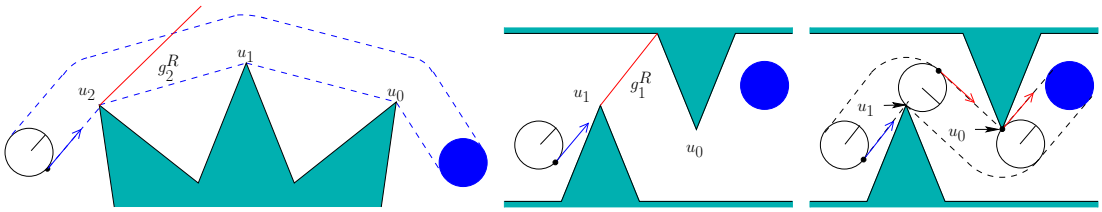


Figura 4.12: La FSM completa para la navegación óptima.

compuestos por un conjunto de estados. Los procedimientos son Align, RPAlign y LPAlign. Estos procedimientos tienen como objetivo principal encontrar un vértice sub-meta. Los procedimientos Align y RPAlign fueron descritos en detalle arriba, el procedimiento LPAlign es sólo el caso simétrico de RPAlign.

El robot rota para quedar alineado con el vértice sub-meta. En Align, el robot rota en sitio ya sea en sentido de las manecillas del reloj o en sentido contrario a las manecillas del reloj. En RPAlign el robot rota en sentido de las manecillas del reloj con respecto al punto rp y en LPAlign el robot rota en sentido contrario a las manecillas con respecto al punto lp.

Los estados en la máquina M son RChase, LChase, DetourL, DetourR y Λ-Reached. RChase hace que el robot se mueva en línea recta hacia el vértice meta derecho o la landmark (cuando el robot se mueve hacia la landmark la dirección rt está alineada con el punto rp_Λ), el sensor omnidireccional está colocado en rp. LChase hace que el robot se mueva en línea recta hacia el vértice meta izquierdo o la landmark (cuando el robot se mueve hacia la landmark la dirección lt está alineada con el punto lp_Λ), el sensor omnidireccional está colocado en lp. DetourR hace que el robot se mueva en línea recta hacia el vértice



(a) (b) (c)

Figura 4.13: a) Una trayectoria codificada en el GNT no bloqueada que involucra sólo brechas derechas, b) Una trayectoria codificada en el GNT que involucra ambos tipos de brechas (izquierdas y derechas), c) Trayectoria del robot.

sub-meta derecho, con el sensor omnidireccional colocado en rp. DetourL hace que el robot se mueva en línea recta hacia el vértice sub-meta izquierdo, con el sensor omnidireccional colocado en lp. El estado Λ-Reached indica que el robot ha alcanzado la landmark y la tarea de navegación termina.

Los vectores de observación para los estados en la máquina M son mostrados en la Tabla 4.13 (refiérase a la Fig. 4.12). Las transiciones entre procedimientos y estados no están etiquetadas con observaciones, debido a que pueden existir más de una observación que genere la transición. Sin embargo, dentro de los procedimientos al nivel de estados todas las transiciones son etiquetadas con la observación correspondiente.

Tabla 4.13: Vectores de observación de los estados en la máquina M, Fig. 4.12.

	FI	LV	FR	RP	LP	VR	VL	AL	BL	UN	GT	CT	O1	O2
yn_{60}	1	1	1	0	0	x	x	x	x	x	x	x	x	x
yn_{61}	x	x	1	1	0	x	x	0	x	x	x	x	x	x
yn_{62}	x	x	1	0	1	x	x	0	x	x	x	x	x	x

4.4.7. Ejemplos de la ejecución de la FSM para trayectorias codificadas en el GNT no bloqueadas

La Fig. 4.13(a) muestra un ejemplo de como M genera una trayectoria óptima para el caso de no bloqueo. En la figura, el GNT codifica la secuencia $H = (g_2^R, g_1^R, g_0^R)$. En este ejemplo, la máquina M atraviesa la siguiente secuencia de estados y conjunto de procedimientos mientras se ejecutan las primitivas de movimiento convenientes: Align, RChase, RPAlign, RChase, RPAlign, RChase, RPAlign, RChase y Λ-Reached.

Ahora describimos la asociación de los estados con cada brecha y la landmark. Primero,

g_2^R es seguido, ejecutando ALIGN y el estado RCHASE. Después, g_1^R es seguido, ejecutando RPALIGN y el estado RCHASE. Después, g_0^R es seguido, ejecutando otra vez RPALIGN y el estado RCHASE. Después, Λ es seguida, ejecutando RPALIGN y el estado RCHASE. Finalmente, cuando el robot alcanza la landmark, M transita al estado Λ-REACHED.

En el ejemplo previo todas las brechas en H fueron del mismo tipo. Usando el ejemplo ilustrado en la Fig. 4.13(b), explicamos la operación de M cuando existen diferentes tipos de brechas. Para alcanzar Λ, el robot sigue la secuencia $H = (g_1^R, g_0^L)$. La secuencia resultante es ALIGN y RCHASE (para seguir (cazar) g_1^R), después RPALIGN y LCHASE, (para seguir (cazar) g_0^L), después LPALIGN y LCHASE (para seguir (cazar) Λ), y finalmente Λ-REACHED.

4.4.8. Considerando todas las posibilidades para la FSM

Los grafos mostrados en las Figs. 4.14 y 4.15 tienen como objetivo mostrar que el diseño de la FSM es exhaustivo, todas las posibilidades son consideradas.

Para el caso de rotación en sitio (vea la Fig. 4.14), las posibilidades son como sigue. (G) el objetivo del robot es alinearse a una brecha, o (Λ) el objetivo del robot es alinearse con la landmark. Primero, considere las posibilidades para el nodo (G). De hecho, existe sólo una posibilidad ya que el alineamiento con una brecha siempre es posible ejecutando una rotación en sitio, por lo tanto, (R) el robot rota en sitio, y (A) se alinea con la brecha. Una vez que el robot está alineado con la brecha, existen solamente dos posibilidades. (B1) existe un bloqueo (detección de tipo 1) o no (NB1). Si (B1) ocurre existe sólo una posibilidad, esta es, (C) el robot encuentra un vértice candidato. Ya que el vértice candidato genera una brecha, la única posibilidad es regresar al nodo (G) para que el robot pueda alinearse con esta brecha. Si (NB1) no existe bloqueo, entonces (SL) el robot se mueve en línea recta. Un movimiento en línea recta sólo puede producir dos posibilidades: (S) existe una trayectoria óptima hacia la brecha (el robot toca el vértice con el punto rp o lp), o, (NS) el robot colisiona (toca ∂E con un punto diferente a rp o lp y por lo tanto, no existe una trayectoria solución).

Ahora considere que (Λ) el objetivo del robot es alinearse con la landmark. Existe sólo una posibilidad, (R) el robot rota en sitio, y (A) se alinea con la landmark. Una vez que el robot está alineado con la landmark, (B4) la trayectoria hacia la landmark siempre se supone bloqueada (detección de bloqueo tipo 4). Desde (B4) hay dos posibilidades: (S) existe una trayectoria óptima hacia la landmark, o la trayectoria está bloqueada por un vértice, ya que este vértice genera una brecha, (G) el robot debe alinearse con una brecha.

Este análisis lista todas las posibilidades para la rotación en sitio.

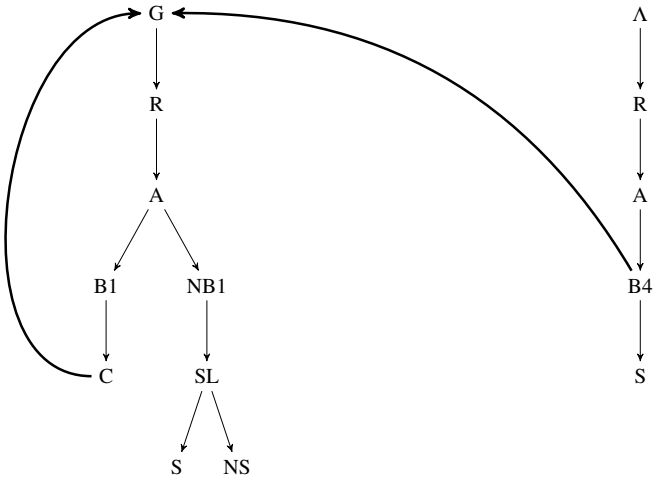


Figura 4.14: Grafo que representa todas las posibilidades en la FSM para una rotación en sitio.

Ahora, consideremos una rotación con respecto al punto rp o lp. Refiérase a la Figura 4.15. (G) el objetivo del robot es alinearse con una brecha, o (Λ) el objetivo del robot es alinearse con la landmark. Primero, considere las posibilidades para el nodo (G). Hay dos posibilidades: (NB2,NB3) la trayectoria hacia el vértice que genera la brecha no está bloqueada, o (B2,B3) está bloqueada (detección de bloqueo tipo 2 o tipo 3). Si (B2,B3) ocurre sólo existe una posibilidad: (SG) un vértice sub-meta es encontrado usando el Algoritmo 4.4.4.1. Una vez que el vértice sub-meta ha sido encontrado, (R) el robot rota. Si (NB2,NB3) ocurre la única posibilidad es (R). Si (R) pasa hay cuatro posibilidades: (NS) el robot colisiona (el robot toca ∂E con un punto diferente a rp o lp), (UG) la brecha generada por el vértice meta se une con otra brecha, (US) la brecha generada por el vértice sub-meta se une con otra brecha, o (A) el robot es capaz de lograr el alineamiento con el vértice que genera la brecha. Si (UG) entonces existe un nuevo vértice meta y es necesario verificar un bloqueo para este vértice; el proceso empieza de nuevo en (G). Si (U) ocurre entonces el vértice sub-meta debe ser recalculado usando el Algoritmo 4.4.4.1 en (SG). Si (A), una vez que el alineamiento ha sido realizado, hay dos posibilidades: (NB1) la trayectoria hacia el vértice no está bloqueada, o (B1) está bloqueada (detección de bloqueo tipo 1). Si (B1) entonces un vértice sub-meta es calculado usando el Algoritmo 4.4.4.1 en (SG). Si (NB1), no hay bloqueo, entonces (SL) el robot se mueve en línea recta. Un movimiento en línea recta sólo puede producir dos posibilidades: (S) existe una trayectoria óptima hacia la brecha (el robot toca el vértice con el punto rp o lp), o (NS) el robot colisiona (toca ∂E con un punto

diferente a rp o lp).

Finalmente, considere que (Λ) el objetivo del robot es alinearse a la landmark. Hay dos posibilidades, (B2) la trayectoria hacia la landmark está bloqueada (detección de bloqueo tipo 2), o (NB2) la trayectoria no está bloqueada. Si (B2) entonces la trayectoria hacia la landmark está bloqueada por un vértice, ya que este vértice genera una brecha, (G) el robot debe alinearse con tal brecha. Si (NB2) entonces, (R) el robot rota. Una vez que el robot rota hay dos posibilidades: (NS) el robot colisiona, no hay solución, o, (A) el robot es capaz de lograr el alineamiento hacia la landmark. Si (A) entonces, una vez que el robot está alineado a la landmark, (B4) la trayectoria hacia la landmark siempre se supone bloqueada (detección de bloqueo tipo 4). Desde (B4) hay dos posibilidades: (S) existe una trayectoria óptima hacia la landmark, o la trayectoria hacia la landmark está bloqueada por un vértice, ya que este vértice genera una brecha, (G) el robot debe alinearse con una brecha. Este análisis lista todas las posibilidades para la rotación con respecto al punto rp o lp.

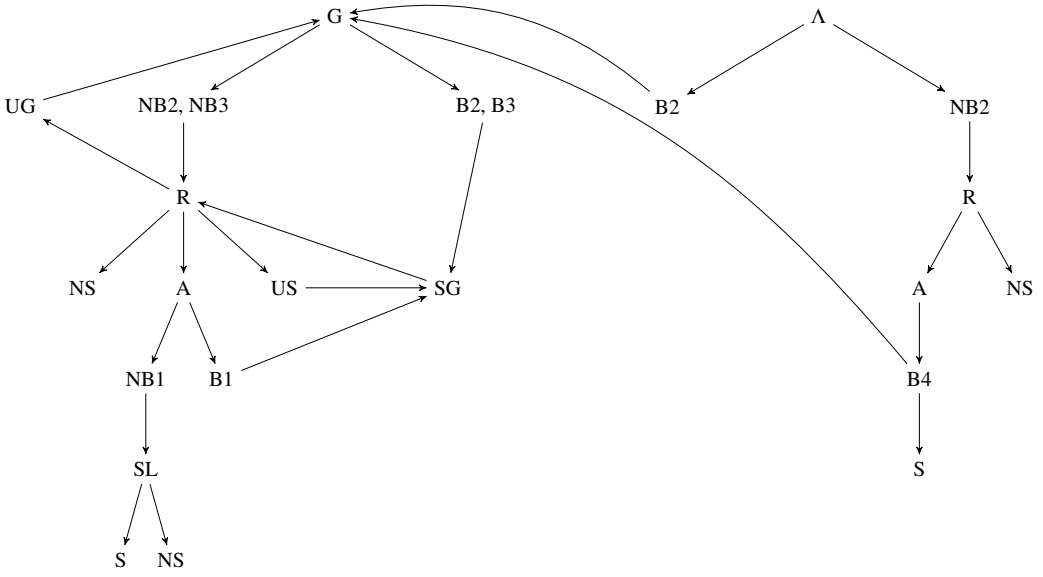


Figura 4.15: Grafo que presenta todas las posibilidades en la FSM para una rotación con respecto a rp o lp.

4.5. La política de movimiento para navegación

En esta sección se sintetizan dos políticas de movimiento a partir la FSM, la cual mapea observaciones con acciones. En la primera, se incluye el movimiento de una torreta que cambia la posición del sensor omnidireccional y en la segunda es implícito.

La política de movimiento está basada en el paradigma de evitar la estimación del estado para llevar a cabo dos mapeos consecutivos: $y \rightarrow x \rightarrow u$, eso es desde la observación y al estado x y después al control u, pero en lugar de eso existe un mapeo directo $y \rightarrow u$.

El vector de observación yn para navegación, que incluye el movimiento de la torreta, tiene 14 observaciones sensoriales binarias (vea la Sección 4.3 para la descripción de cada observación binaria). Es interesante notar que el vector de observación ym para la política de movimiento basada en retroalimentación, sin el movimiento de la torreta es un subconjunto de yn, esto es $ym \subset yn$. Esto significa que algunos de los elementos de observación de yn no son relevantes para la política de movimiento, donde se considera implícito el movimiento de la torreta, y éstos pueden tomar cualquier valor, consecuentemente no están incluidos.

4.5.1. Política de movimiento basada en retroalimentación incluyendo el movimiento de la torreta

En esta política de movimiento basada en retroalimentación, se incluye el movimiento de la torreta. Cada vez que el robot se traslada, la torreta está estática y cada vez que la torreta está en movimiento el robot está inmóvil.

Dependiendo de la observación, el robot ejecutará una de las cinco diferentes primitivas de movimiento o una de las dos primitivas de movimiento de la torreta: (1) movimiento en línea recta; (2) rotación en sitio en sentido de las manecillas del reloj; (3) rotación en sitio en sentido contrario a las manecillas del reloj; (4) rotación con respecto al punto rp en sentido a las manecillas del reloj y (5) rotación con respecto al punto lp en sentido contrario a las manecillas del reloj. En las siguientes primitivas de movimiento de la torreta, el robot está inmóvil y la torreta se mueve: (6) el robot mueve la torreta sobre su frontera en sentido de las manecillas del reloj y (7) el robot mueve la torreta sobre su frontera en sentido contrario a las manecillas del reloj.

La estrategia de movimiento basada en retroalimentación puede ser establecida como: $\gamma : \{0,1\}^{14} \rightarrow \{-1,0,1\}^3$. La estrategia de movimiento basada en retroalimentación está dada por $\gamma(yn_i) = (w_r, w_l, v_t)$, donde w_r y w_l son las velocidades angulares de las

ruedas derecha e izquierda, y v_t es la velocidad de desplazamiento de la torreta sobre la frontera del robot. El conjunto de todos los 16384 posibles vectores de observación puede ser agrupado dejando que x denote "cualquier valor" para obtener:

$$yn_{63} = (0, x, x, x, x, x, 1, 1, 0, 0, x, x, x, 0), \quad yn_{64} = (0, x, x, x, x, x, 0, 1, 0, 0, x, x, x, 0),$$
$$yn_{65} = (0, x, 0, x, x, 0, 0, 0, x, 0, 0, 0, 0, 0), \quad yn_{66} = (0, x, 0, x, x, 1, 1, 0, 0, 0, x, 0, 0, 0),$$
$$yn_{67} = (0, 0, 0, x, x, 0, 0, 0, 0, 0, 1, 0, 1, 0), \quad yn_{68} = (0, x, 0, x, x, 0, 0, 0, 1, 0, x, 0, 1, 0),$$
$$yn_{69} = (0, x, 0, x, x, 1, 1, 0, 0, 0, x, 1, 1, 0), \quad yn_{70} = (0, x, 1, 1, 0, 1, 1, 0, 0, 0, x, 0, 0, 0),$$
$$yn_{71} = (0, x, 1, 1, 0, 1, 1, 0, 0, 0, x, 1, 1, 0), \quad yn_{72} = (0, 0, 1, 1, 0, x, 0, 0, 0, 0, 1, 0, 1, 0),$$
$$yn_{73} = (0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0), \quad yn_{74} = (0, 0, 1, 1, 0, x, 0, 0, 0, 1, 1, 0, 0, 0),$$
$$yn_{75} = (0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), \quad yn_{76} = (0, x, 1, 0, 1, 1, 1, 0, 0, 0, x, 1, 1, 0),$$
$$yn_{77} = (0, x, 1, 0, 1, 1, 1, 0, 0, 0, x, 0, 0, 0), \quad yn_{78} = (0, x, 1, 0, 1, 0, x, 0, 0, 0, 0, 0, 0, 0),$$
$$yn_{79} = (0, 0, 1, 0, 1, 0, x, 0, 0, 1, 0, 0, 1, 0), \quad yn_{80} = (0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0),$$
$$yn_{81} = (x, x, x, x, x, x, x, x, x, x, x, x, 0, 1), \quad yn_{82} = (0, x, x, x, x, x, x, 1, 1, 0, x, x, 0, 0),$$
$$yn_{83} = (0, 0, 0, x, x, 0, 0, 0, 0, 0, 1, 0, 0, 0), \quad yn_{84} = (0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0),$$
$$yn_{85} = (0, x, 1, 1, 0, 1, 1, 0, 0, 0, x, 1, 0, 0), \quad yn_{86} = (0, x, 1, 1, 0, 1, 1, 0, 0, 1, x, 0, 0, 0),$$
$$yn_{87} = (0, 0, 1, 0, 1, 0, 1, 0, 0, x, 1, 0, 0, 0), \quad yn_{88} = (0, x, 1, 0, 1, 0, 1, 0, 0, 1, x, 0, 0, 0),$$
$$yn_{89} = (0, x, 1, 0, 1, 0, x, 0, 1, 0, 0, 0, 0, 0), \quad yn_{90} = (x, x, x, x, x, x, x, x, x, x, x, x, 1, 1),$$
$$yn_{91} = (0, x, x, x, x, x, x, 1, 1, 0, x, x, 1, 0), \quad yn_{92} = (0, 0, x, x, x, x, 0, 0, 0, 0, 0, 0, 1, 0),$$
$$yn_{93} = (0, x, 1, 1, 0, 1, 0, 0, 0, 1, x, 0, 1, 0), \quad yn_{94} = (0, x, 1, 1, 0, x, 0, 0, 1, 0, x, 0, 1, 0),$$
$$yn_{95} = (0, 0, 1, 0, 1, x, x, 0, 0, 1, 1, x, 1, 0), \quad yn_{96} = (0, x, 1, 0, 1, 1, 1, 0, 0, 0, x, 0, 1, 0),$$
$$yn_{97} = (0, x, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0).$$

La estrategia γ puede ser finalmente codificada como

(1) $\gamma(yn_{63} \vee yn_{64}) = (1, 1, 0)$;
(2) $\gamma(yn_{65} \vee yn_{66}) = (-1, 1, 0)$;
(3) $\gamma(yn_{67} \vee yn_{68} \vee yn_{69}) = (1, -1, 0)$;
(4) $\gamma(yn_{70} \vee yn_{71} \vee yn_{72} \vee yn_{73} \vee yn_{74} \vee yn_{75}) = (0, 1, 0)$;
(5) $\gamma(yn_{76} \vee yn_{77} \vee yn_{78} \vee yn_{79} \vee yn_{80}) = (1, 0, 0)$;
(6) $\gamma(yn_{81} \vee yn_{82} \vee yn_{83} \vee yn_{84} \vee yn_{85} \vee yn_{86} \vee yn_{87} \vee yn_{88} \vee yn_{89}) = (0, 0, 1)$;
(7) $\gamma(yn_{90} \vee yn_{91} \vee yn_{92} \vee yn_{93} \vee yn_{94} \vee yn_{95} \vee yn_{96} \vee yn_{97}) = (0, 0, -1)$.

en la que $\vee$ significa "o".

Un programa computacional fue creado para asegurar que si un estado se puede alcanzar por más de una transición (observación) el conjunto de observaciones validas, unión de las

observaciones que llevan al estado sean correctas. Por ejemplo: $(1, x, x, 1, x) \vee (1, x, x, 0, x) = (1, x, x, x, x)$.

La obtención de la política de movimiento (reducción de expresiones lógicas) fue realizada manualmente usando lógica booleana. Sin embargo, se diseñó un segundo programa computacional para verificar que el conjunto de observaciones que corresponden a la ejecución de una primitiva de movimiento no se traslape con otro conjunto asociado a una primitiva diferente.

4.5.2. Política de movimiento basada en retroalimentación considerando implícito el movimiento de la torreta

Solamente seis observaciones afectan el control de los motores de las ruedas. El vector de observación usado es ym_i = (FR, RP, LP, AL, BL, O1). Refiérase a la Sección 4.3 para el significado de cada elemento del vector de observación. Dependiendo de la observación, una de las cinco diferentes primitivas de movimiento será ejecutada: (1) movimiento en línea recta; (2) rotación en sitio en sentido de las manecillas del reloj; (3) rotación en sitio en sentido contrario a las manecillas del reloj; (4) rotación con respecto al punto rp en sentido a las manecillas del reloj; (5) rotación con respecto al punto lp en sentido contrario a las manecillas del reloj. Recuerde que las velocidades angulares de las ruedas llevan a una de estás primitivas de movimiento. Por lo tanto, la estrategia de movimiento puede ser establecida por: $\gamma : \{0, 1\}^6 \rightarrow \{-1, 0, 1\}^2$ para obtener $\gamma(ym_i) = (w_r, w_l)$. El conjunto de los 64 vectores de observación posibles puede ser agrupado dejando que x denote "cualquier valor" para obtener: $ym_1 = (x, x, x, 1, 0, x)$, $ym_2 = (0, x, x, 0, x, 0)$, $ym_3 = (0, x, x, 0, x, 1)$, $ym_4 = (1, 0, 1, 0, x, x)$, $ym_5 = (1, 1, 0, 0, x, x)$.

La estrategia γ puede ser codificada como

$$\begin{aligned} (1)\gamma(ym_1) &= (1, 1); \quad (4)\gamma(ym_2) = (-1, 1); \\ (2)\gamma(ym_3) &= (1, -1); \quad (5)\gamma(ym_4) = (0, 1); \\ (3)\gamma(ym_5) &= (1, 0). \end{aligned}$$

4.6. Prueba de la navegación óptima

4.6.1. Caminos no bloqueados codificados en el GNT

En esta sección establecemos que el robot sigue un camino óptimo en el sentido de la distancia Euclidiana en ausencia de bloqueos. La trayectoria más corta hacia Λ es codificada como una secuencia de brechas en el GNT. Sea $U = (u_n, u_{n-1}, ..., u_0)$ la secuencia de intervalos conectados $u_i \subset \partial E$ que el robot toca cuando el sensor de brechas (fijo a la frontera del robot) se mueve desde su posición inicial hasta su posición final en Λ. En esta sección establecemos que el robot ejecuta una trayectoria óptima en distancia Euclidiana en ausencia de bloqueos, es decir, no se realizan desvíos entre los intervalos u_{i+1} y u_i.

Sea $H = (g_n, g_{n-1}, ..., g_0)$ la secuencia correspondiente de brechas que son seguidas, en la cual $g_i \in H$ es la brecha que está siendo seguida sobre el camino hacia u_i o mientras viaja en u_i.

Ahora consideremos el problema en términos del espacio de configuraciones del robot. La región obstáculo en el espacio de configuraciones es obtenida al hacer crecer los obstáculos del entorno con el radio del robot. Sea C la proyección de la región obstáculo dentro del plano, de modo que la rotación es ignorada dado que el robot es un disco. Sea $V = (v_n, v_{n-1}, ..., v_0)$ la secuencia de intervalos $v_i \subset \partial C$ obtenidos por la transformación de la secuencia de intervalos U desde ∂E hacia ∂C, elemento por elemento. El siguiente lema usa la definición de una bitangente generalizada presentada en [93].

Lema 4.6.1 *Siguiendo la secuencia de brechas H se produce el camino más corto si y soló si: 1) existe un camino libre de colisiones desde el centro del robot hasta v_n, 2) existe una línea bitangente (generalizada) entre v_{i+1} y v_i, 3) existe un camino libre de colisiones desde v_0 hasta el centro de la landmark, y 4) C está conectado.*

Demostración. Primero demostramos la dirección de izquierda a derecha del enunciado si y sólo si. El sensor de brechas está ubicado sobre el punto rp o lp. Seguir la secuencia H de brechas hace que el robot toque los intervalos $u_i \subset \partial E$ con los puntos rp y lp. Como consecuencia, el centro del robot visita los intervalos $v_i \subset \partial C$. Si seguir las brechas en H produce la trayectoria más corta y el centro del robot visita cada intervalo $v_i \subset V$, entonces una trayectoria solución global debe existir (C está conectado) y deben existir trayectorias libres de colisión en línea recta locales conectando la ubicación inicial del centro del robot con la secuencia V, conectando cada par v_{i+1} y $v_i \in V$, y la secuencia V con Λ, que en

conjunción son las cuatro condiciones en el enunciado del lema.

Para la demostración de la otra dirección del enunciado, procedemos por contradicción. Supongamos que las tres primeras condiciones se satisfacen y que existe una trayectoria solución, por lo tanto, la cuarta condición también se satisface. Ahora supongamos que la trayectoria más corta no es la representada por las tres primeras condiciones. Si tal trayectoria existe, entonces debe existir un bloqueo en las trayectorias en línea recta de ya sea la posición inicial del centro del robot y v_n, entre intervalos v_{i+1} y $v_i \in V$, o entre v_0 y Λ, por lo tanto, al menos una de las tres condiciones no se cumple, lo cual es una contradicción. Además, como la trayectoria representada por las tres condiciones visita los intervalos $v_i \in V$, entonces la trayectoria es transitable siguiendo las brechas en H. El resultado se mantiene. ■

4.6.2. Caminos bloqueados codificados en el GNT

Ahora consideraremos los casos para los que cualquiera de las tres primeras condiciones del Lema 4.6.1 es violada, lo que significa que el robot quedará bloqueado cuando aplique el GNT para un robot puntual. Para estos casos, se requieren varias formas de "desvíos". El camino codificado en el GNT está basado en líneas bitangente entre intervalos en E. Sin embargo, en el espacio de configuraciones, desaparecen algunas líneas bitangentes. Las líneas bitangentes en el espacio de trabajo que permanecen en el espacio de configuraciones son desplazadas por una distancia r o están rotadas por un ángulo fijo.

El camino codificado en el GNT no puede ser ejecutado por el robot cuando existe un bloqueo para seguir (cazar) $g_i \in H$ (o Λ). Cuando esto ocurre, significa que: 1) el robot está en una zona en la cual no puede detectar el cruce de una línea bitangente en C, esto significa que debido a la anchura del robot disco, existe una línea bitangente entre la ubicación del sensor y el vértice u_n, pero no existe una linea bitangente entre el centro del robot y v_n, 2) no existe una línea bitangente entre v_{i+1} y v_i en C, 3) no existe un camino libre en línea recta para seguir Λ cuando el robot percibe la Λ, o 4) C está desconectado. Éstas son las condiciones del Lema 4.6.1. En el Teorema 4.6.5, afirmamos que nuestra estrategia de navegación es capaz de tratar los primeros tres casos presentados arriba. Por lo tanto, siempre es posible detectar una trayectoria óptima libre de colisión si existe alguna. La desconexión de C, el caso en el cual no existe camino hacia la landmark, producirá una colisión del robot cuando es comandado por la FSM M. En el Teorema 4.6.6, probamos que si nuestra estrategia de movimiento no encuentra una trayectoria libre de colisión para

alcanzar la landmark, entonces no existe una trayectoria para alcanzarla.

Si el robot detecta un camino bloqueado, entonces realiza un desvío para evitar los obstáculos que bloquean la trayectoria codificada en el GNT. No podemos re-planificar la trayectoria completa hacia la Λ debido a que la trayectoria depende de la brecha $g_i \in H$ (o la Λ), esto es en el campo de vista del sensor de brechas. Por esta razón el desvío para evadir obstáculos es realizado cuando el robot detecta una trayectoria bloqueada mientras sigue (caza) g_i o la Λ.

En lo que resta de esta sección, sin pérdida de generalidad, cuando se refiere a una rotación con respecto un punto, es una rotación con respecto al punto rp. Para probar algunos lemas y teoremas abajo, introducimos los siguientes conceptos.

Consideremos un vértice candidato dado u_c. Si u_c está bloqueado por cualquier otro vértice entonces el vértice candidato u_j que deforma más la trayectoria en línea recta hacia u_c se dice que *está en la frontera de las restricciones*. Para un $u_c = u_c^R$, el u_j que está en la frontera de la restricción corresponde al u_j^L que bloquea la trayectoria a u_c^R, y tiene el ángulo θ_L más grande, el último en el orden angular, vea la Fig. 4.16.

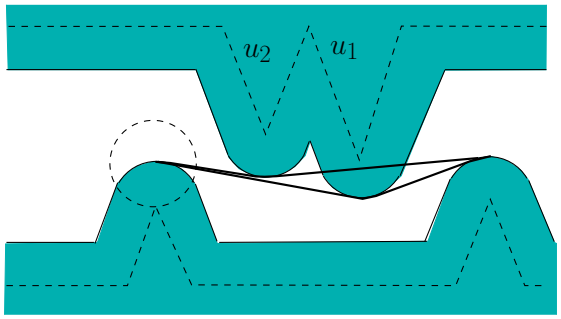


Figura 4.16: El vértice u_1 se encuentra en la frontera de la restricción.

Simétricamente, para un $u_c = u_c^L$ el u_j que está en la frontera de la restricción corresponde a u_j^R que bloquea la trayectoria hacia u_c^L, y tiene el ángulo θ_R menor, el primero en el orden angular.

Lema 4.6.2 *El Algoritmo 4.4.4.1 encuentra un vértice sub-meta entre los vértices que generan brechas en la configuración del robot cuando el algoritmo es invocado. Los vértices candidato, incluyendo al vértice sub-meta, se encuentran en la frontera de las restricciones del movimiento semi-libre de colisión impuesto por los vértices que bloquean la trayectoria hacia el vértice meta.*

Demostración. Por construcción el Algoritmo 4.4.4.1 detecta los vértices izquierdos que bloquean la trayectoria hacia el vértice candidato derecho o los vértices derechos que bloquean la trayectoria hacia el vértice candidato izquierdo. Para un vértice candidato derecho bloqueado, el Algoritmo 4.4.4.1 selecciona como nuevo vértice candidato al vértice izquierdo con el mayor θ_L, el último en el orden angular, que se encuentra en la frontera de la restricción. Para un vértice candidato izquierdo bloqueado, el Algoritmo 4.4.4.1 selecciona como nuevo vértice candidato al vértice derecho con el menor θ_R, el primero en el orden angular, que se encuentra en la frontera de la restricción. Este procedimiento se repite hasta que el vértice candidato no está bloqueado, seleccionado este último como el vértice sub-meta. Por lo tanto, el resultado se mantiene. ■

Un camino hacia una landmark puede estar bloqueado por un vértice oculto. Si un vértice es oculto o no, depende de las posiciones relativas entre el sensor omnidireccional y el vértice. Un vértice oculto es importante porque si bloquea el camino del robot debe ser considerado para encontrar el camino óptimo, lo que se hace usando el Algoritmo 4.4.4.1 y por lo tanto si hubo un vértice oculto cuando el Algoritmo 4.4.4.1 fue invocado, entonces este algoritmo debe ser invocado nuevamente cuando el vértice genere una brecha.

Exhaustivamente, existen cuatro casos posibles analizando si (EG) el vértice oculto eventualmente generará una brecha o (NG) nunca generará una brecha y si (S) existe una trayectoria libre de colisión hacia la landmark o no (NS), mientras el robot se mueve comandado por la FSM. Más adelante mostramos, con los cuatro casos, que si existe una solución (una trayectoria óptima), entonces la FSM la encuentra, o si no existe una solución, entonces el robot colisiona.

El caso 1 es el siguiente: 1) en la trayectoria que el robot sigue, el vértice oculto eventualmente genera una brecha (EG) y 2) no existe una trayectoria óptima (NS). Vea la Fig. 4.17(a) y el elemento (EG,NS) en la Tabla 4.14. En este caso la FSM regresa un vértice sub-meta, pero el robot colisiona ya que no existe una trayectoria solución.

El caso 2 es: 1) el vértice oculto eventualmente genera una brecha (EG) mientras el robot se mueve comandado por la FSM y 2) existe una trayectoria óptima (S). Vea las Figs. 4.17(b) y 4.17(c), y el elemento (EG, S) en la Tabla 4.14. El Lema 4.6.3 demuestra que si existe una trayectoria solución, entonces cada vértice oculto siempre generará eventualmente una brecha mientras el robot es comandado por la FSM.

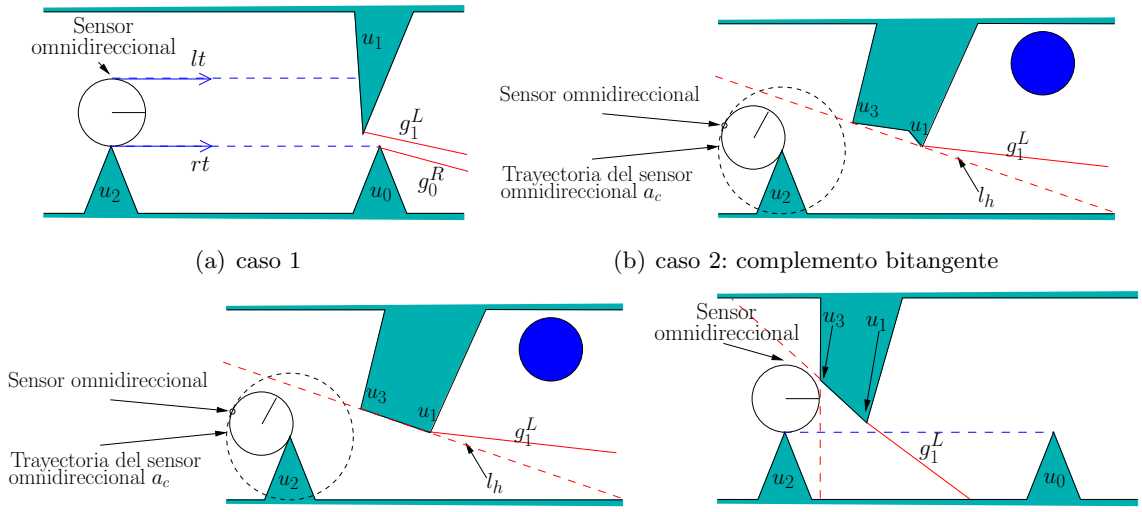
El caso 3 es: 1) el vértice oculto nunca genera una brecha mientras el robot se mueve comandado por la FSM (NG) y 2) no existe una trayectoria óptima (NS). Vea la Fig. 4.17(d)

Tabla 4.14: 4 casos relacionados con la existencia de la trayectoria óptima y un vértice oculto.

	NS	S
EG	Yes	Yes
NG	Yes	No

y el elemento (NG, NS) en la Tabla 4.14. En este caso la FSM nunca tomará en cuenta al vértice oculto, pero la no solución será detectada debido a que el robot colisionará.

El caso 4 es: 1) el vértice oculto nunca genera una brecha mientras el robot se mueve comandado por la FSM (NG) y 2) existe una trayectoria óptima (S). El elemento (NG, S) en la Tabla 4.14. Note que dado que existe una solución, este caso es el complemento del caso 2 y por el Lema 4.6.3, el caso 2 siempre ocurre, por lo tanto el caso 4 *no existe*.



(a) caso 1

(b) caso 2: complemento bitangente

(c) case 2: el mismo intervalo conectado sobre ∂E

(d) case 3

Figura 4.17: Casos relacionados a la existencia de una trayectoria óptima y un vértice oculto.

Lema 4.6.3 *Supongamos que un vértice sub-meta u_s es obtenido por el Algoritmo 4.4.4.1. Si existe una trayectoria libre de colisión hacia la landmark Λ, entonces un vértice oculto u_h que bloquea a u_s debe generar eventualmente una brecha, mientras el robot rota con respecto a rp o lp para alinear lt a un vértice sub-meta izquierdo u_s.*

Demostración. Refiérase a las Figs. 4.17(b) y 4.17(c). El robot rota con respecto al punto *rp* para alinear *lt* a un vértice sub-meta izquierdo u_s. Mientras el robot rota, el sensor omnidireccional se mueve en una trayectoria de arco de circulo a_c. Considere una línea l_h

que contiene a u_s y u_h. Para bloquear la trayectoria hacia u_s por u_h, la línea l_h debe cruzar a_c, de otra manera, el robot siempre es capaz de alinear lt a u_s. Además la línea l_h debe ser cruzada por el sensor omnidireccional para alinear lt a el vértice u_s, ya que lt es tangente a la frontera del robot, l_h contiene a u_s y l_h es una linea secante al circulo a_c. Para esta rotación del robot, existen dos casos de acuerdo a los eventos críticos originados por cruzar la línea l_h. El primero ocurre cuando el vértice u_h genera una brecha y esta brecha se une con la brecha generada por u_s, vea la Fig. 4.17(b). Ya que dos brechas se unen, debe existir un complemento bitangente entre u_h y u_s (como se definió en [93]) sobre l_h. Originalmente el vértice u_h no generó una brecha, pero debido al movimiento del sensor omnidireccional se cruza el complemento bitangente (se unen las brechas generadas por u_s y u_h), por lo tanto, un gap originado por u_s debe aparecer. El segundo caso ocurre cuando el gap generado por u_s cambia de vértice, y la brecha es generada por el vértice u_h, vea la Fig. 4.17(c). En este segundo caso el segmento que una los vértices u_s y u_h están sobre la línea l_h. Ya que el segmento de línea que une u_s y u_h es el mismo intervalo conectado, entonces el gap que es generado por u_s debe cambiar el vértice que lo genera. La brecha debe ser generada por el vértice u_h en el momento en que el sensor de brechas cruza la línea l_h mientras el robot está rotando. Por lo tanto, el vértice u_h debe generar eventualmente una brecha. El resultado se mantiene. ■

Existe un lema equivalente para el caso en el que el robot rota con respecto al punto lp para alinear rt a un vértice sub-meta derecho u_s.

Lema 4.6.4 *Considere un procedimiento ζ, el cual invoca al Algoritmo 4.4.4.1 cada vez que un vértice oculto es detectado. El procedimiento ζ termina y encuentra un vértice sub-meta. El camino hacia el vértice sub-meta es óptimo en el sentido de la distancia Euclidiana viajada por el centro del robot.*

Demostración. Por el Lema 4.6.2 el Algoritmo 4.4.4.1 entrega un vértice sub-meta. Por el Lema 4.6.3 un vértice oculto siempre es detectado. Si el vértice sub-meta no está bloqueado por un vértice oculto entonces el vértice sub-meta no es recalculado. De otra forma, el Algoritmo 4.4.4.1 es invocado otra vez. Por lo tanto, el procedimiento ζ encuentra un vértice sub-meta u_s y como existe un número finito de vértices el procedimiento ζ termina. Además, por el Lema 4.6.2 un vértice sub-meta u_s está en la frontera de la restricción y no está bloqueado, por lo tanto la trayectoria hacia el vértice sub-meta u_s es óptima. ■

Ahora, presentamos el teorema que garantiza la navegación óptima al utilizar M.

Teorema 4.6.5 *La trayectoria que sigue el centro del robot cuando es comandado por el autómata M, usando la información codificada en el GNT y realizando desvíos cuando la trayectoria en línea recta para seguir $g_i \in H$ está bloqueada o cuando la trayectoria en línea recta para seguir la Λ está bloqueada, es globalmente óptima en el sentido de la distancia Euclidiana.*

Demostración. Supongamos que existe una trayectoria libre de colisión hacia la landmark, esto es, C está conectado. La trayectoria codificada en el GNT es la trayectoria más corta para un punto en el espacio de trabajo y está en la misma clase homotópica que la trayectoria más corta en C debido a que E y C son simplemente conectados. La secuencia de intervalos conectados V en ∂C que el robot viaja solamente cambia cuando cualquiera de las primeras tres condiciones del Lema 4.6.1 no son satisfechas. Sin embargo, incluso si cualquiera de las primeras tres condiciones del Lema 4.6.1 no son satisfechas entonces la secuencia de intervalos en V no cambia de orden, ya que la clase homotópica de la trayectoria más corta en C sigue siendo la misma, pero nuevos vértices sub-meta son añadidos correspondiendo a nuevos intervalos de ∂C.. Estos vértices sub-meta producen desvíos entre intervalos originales consecutivos de v_i y v_{i-1}, y son localmente óptimos (como se ha demostrado en el Lema 4.6.4) ya que todos ellos pertenecen a la frontera de la restricción (como se ha demostrado en el Lema 4.6.2). Por consiguiente, la trayectoria global resultante es óptima. ■

El siguiente Teorema 4.6.6 es uno de los resultados principales de esta tesis. En resumen, este teorema indica que si nuestra estrategia de movimiento no encuentra una solución, entonces no hay solución. La lógica detrás de este resultado es la siguiente: Nuestra estrategia de movimiento mueve el centro del robot lo menos posible, en otras palabras el volumen del espacio que el robot barre mientras se mueve, es tan pequeño como es posible, por consiguiente la estrategia de movimiento propuesta requiere la región más pequeña del espacio libre para mover al robot. Este resultado también aumenta el interés del criterio de optimización propuesto, es decir, mover el centro de un robot circular lo menos posible, ya que desde un punto de vista de la existencia geométrica de una solución, se establece una solución para el peor escenario -el entorno poligonal más restringido, esto es, el entorno con el pasaje estrecho más pequeño, tal que, este pasaje es más ancho que el diámetro del robot-.

Teorema 4.6.6 *Si el robot es comandado por el autómata M que produce la trayectoria óptima en el sentido de la distancia Euclidiana y el robot toca ∂E con un punto diferente a*

lp o rp (el robot colisiona) entonces no existe una trayectoria para alcanzar la landmark.

Demostración. Según el Lema 4.6.4 los vértices sub-meta son encontrados. La restricción de una trayectoria libre de colisión es impuesta por todos los vértices que bloquean la trayectoria en línea recta hacia el vértice meta. Los vértices sub-meta están ubicados en la frontera de la restricción del movimiento semi-libre de colisión y deben ser tocados por el punto rp o lp. Por lo tanto, si el robot colisiona (esto es el robot toca ∂E con un punto diferente a lp o rp), entonces no existe solución. ■

4.7. Implementación

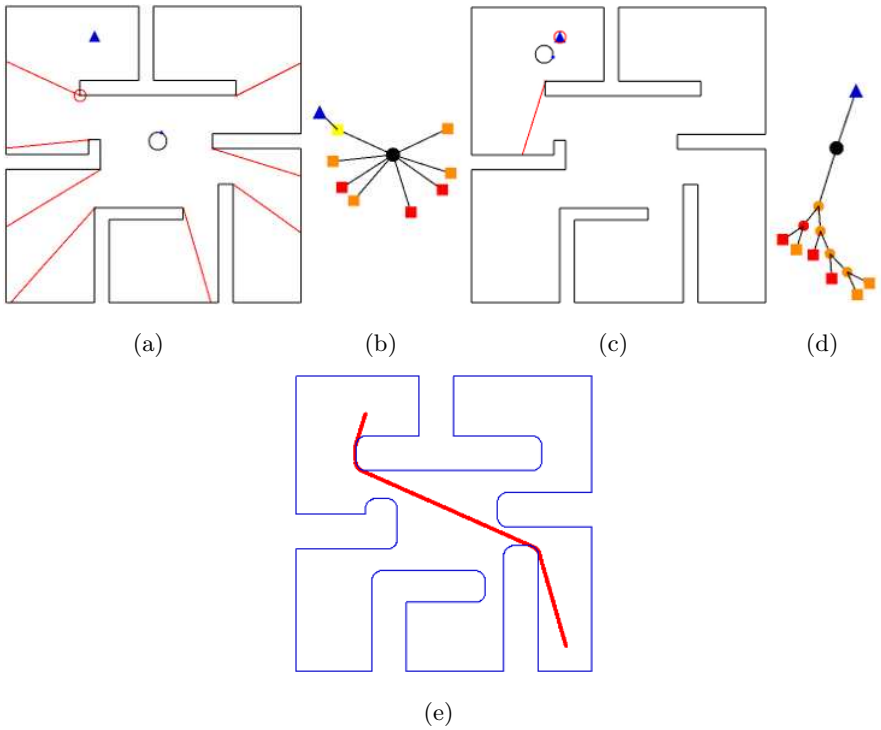


Figura 4.18: Una simulación de la navegación óptima para un robot disco para el caso donde no hay bloqueos. (e) Muestra la trayectoria en el espacio de configuración que el robot viaja para ir hacia la landmark desde la parte inferior derecha del ambiente.

El método completo ha sido implementado y se incluyen los resultados de las simulaciones. Todos nuestros experimentos en simulación se ejecutaron sobre una PC equipada con un procesador de 4 núcleos con 4 GB de RAM, con sistema operativo Linux y

fue programado en C++ usando la librería computacional LEDA. Nuestra implementación emula exactamente la FSM.

El tiempo de planificación para obtener cada uno de los cinco experimentos en simulación presentados en esta sección, es siempre menor a un segundo. Estos tiempos de planificación no incluyen el tiempo de ejecución de la navegación misma.

Hemos implementado el método en un lenguaje de programación con varios objetivos: (1) para ilustrar la ejecución de la FSM, (2) para mostrar gráficamente la trayectoria en el espacio de configuraciones C, y (3) para presentar la evolución del GNT mientras la tarea de navegación es ejecutada.

El código desarrollado para generar las simulaciones que mostramos en esta sección consta de aproximadamente de 16000 lineas.

La Fig. 4.18 muestra una simulación de la navegación óptima basada en brechas para un robot disco para el caso en que ninguno de los vértices que el robot toca estuvieron bloqueados. La figura muestra resultados del programa de simulación. Las Figs. 4.18(a) y 4.18(c) muestran al robot en diferentes momentos mientras sigue la secuencia de brechas para alcanzar la landmark. A la derecha de cada figura se muestra el GNT completo con la representación usada en [93]. La landmark a seguir (cazar) es marcada como un triangulo azul en el espacio de trabajo y en el GNT como un nodo hoja con forma de triangulo. La Fig. 4.18(a) muestra al robot siguiendo (cazando) la brecha en la trayectoria hacia la landmark. Finalmente, la Fig. 4.18(c) muestra al robot siguiendo (cazando) la landmark. La Fig. 4.18(e) muestra la trayectoria más corta en el espacio de configuraciones que el robot recorre para navegar hacia la landmark. Este camino es calculado en base a la información obtenida por los sensores del robot y usando el autómata M de la Sección 4.4.

La Fig. 4.19 muestra un segundo ejemplo de navegación para alcanzar la landmark para el caso donde existen bloqueos en la trayectoria codificada en el GNT. La Fig. 4.19(a) muestra al robot alineado a un vértice derecho. La Fig. 4.19(b) muestra el caso en el cual el primer vértice sub-meta es un vértice u_p cuando el robot está tocando ∂E con rp (el vértice u_4 en la figura). El Algoritmo 4.4.4.1 es usado para encontrar este vértice sub-meta u_p. Las Figs. 4.19(c) y 4.19(d) muestran algunas escenas de la tarea de navegación. La Fig. 4.19(e) muestra la trayectoria más corta en distancia en el espacio de configuraciones proyectado C que el centro del robot viaja para alcanzar la landmark. Note que los vértices u_5 y u_6 no son tocados por el robot. La Fig. 4.19(f) muestra el GNT cuando el robot sigue al landmark.

La Fig. 4.20 muestra otro ejemplo de navegación para el caso donde existen bloqueos

en la trayectoria codificada en el GNT. La Fig. 4.20(a) muestra el caso en el cual el vértice sub-meta es un vértice u_n (el vértice u_2 en la figura). El Algoritmo 4.4.4.1 también es usado para encontrar este vértice sub-meta. Las Figs. 4.20(b), 4.20(c) y 4.20(d) muestran algunas escenas de la tarea de navegación. La Fig. 4.20(e) muestra la trayectoria más corta en distancia en el espacio de configuraciones proyectado C que el centro del robot viaja para alcanzar la landmark. Note que el vértice u_3 no es tocado por el robot. La Fig. 4.20(f) muestra el GNT cuando el robot alcanza la landmark.

La Fig. 4.21 muestra un ejemplo en el cual en la trayectoria más corta para alcanzar la landmark, el robot se mueve en contacto con los segmentos del entorno poligonal (en contacto con ∂E). Las Figs. 4.21(b) y 4.21(c) muestran este caso. La Fig. 4.21(e) muestra la trayectoria más corta en distancia en el espacio de configuraciones proyectado C que el centro del robot viaja para alcanzar la landmark. La Fig. 4.21(f) muestra el GNT cuando el robot sigue la landmark.

La Fig. 4.22 muestra un ejemplo de un vértice oculto (el vértice u_1). La Fig. 4.22(a) muestra el comienzo de la trayectoria del robot. La Fig. 4.22(b) muestra que en el momento cuando el robot está tocando el vértice u_3 con rp teniendo el sensor omnidireccional colocado en rp, el vértice u_1 no genera una brecha (es un vértice oculto). Para encontrar un vértice sub-meta, el Algoritmo 4.4.4.1 es ejecutado sin considerar el vértice u_1. El Algoritmo 4.4.4.1 determina que el vértice meta u_{goal} no está bloqueado. Cuando el robot rota para alinear lt hacia la brecha meta izquierda, esta brecha se une con la brecha generada por u_1, vea la Fig. 4.22(c). Cuando las brechas generadas por los vértices u_{goal} y u_1 se unen, el Algoritmo 4.4.4.1 es invocado otra vez. El algoritmo determina que u_2 es el vértice sub-meta. Ya que u_2 no está bloqueado por un vértice oculto, el Algoritmo 4.4.4.1 no es invocado otra vez.

La Fig. 4.22(d) muestra rt alineado con el vértice u_2. La Fig. 4.22(e) muestra la trayectoria más corta en distancia en el espacio de configuraciones C. La Fig. 4.22(f) muestra el GNT cuando el robot sigue la landmark.

Vídeos que muestran todos los experimentos están disponibles en:

http://www.cimat.mx/ %7Erigolpz/Videos/Prueba1.avi
http://www.cimat.mx/ %7Erigolpz/Videos/Prueba2.ogv
http://www.cimat.mx/ %7Erigolpz/Videos/Prueba3.ogv
http://www.cimat.mx/ %7Erigolpz/Videos/Prueba4.ogv
http://www.cimat.mx/ %7Erigolpz/Videos/Prueba5.ogv

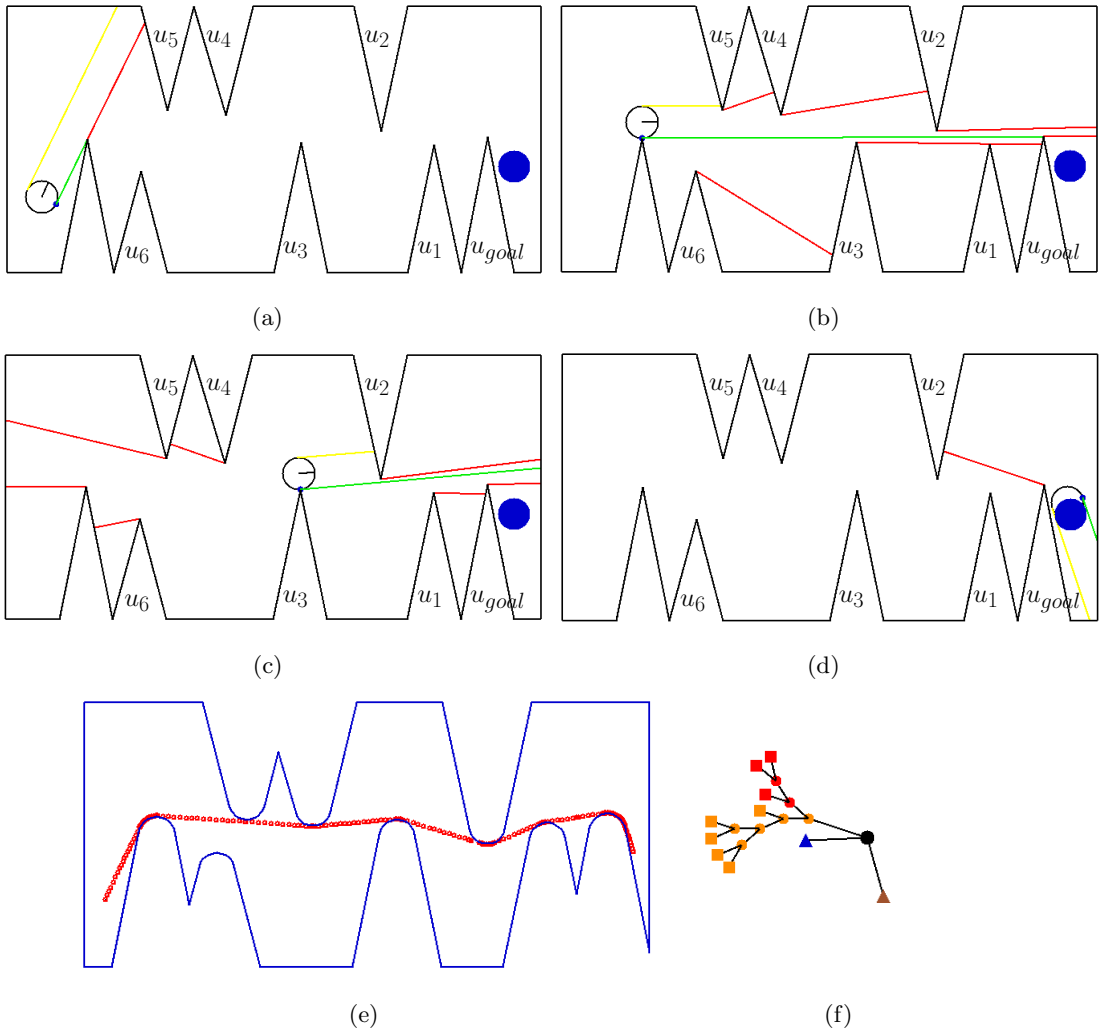


Figura 4.19: Ejemplo en el cual el primer vértice sub-meta es un vértice u_p cuando el robot está tocando ∂E con rp. Dicho vértice es u_4.

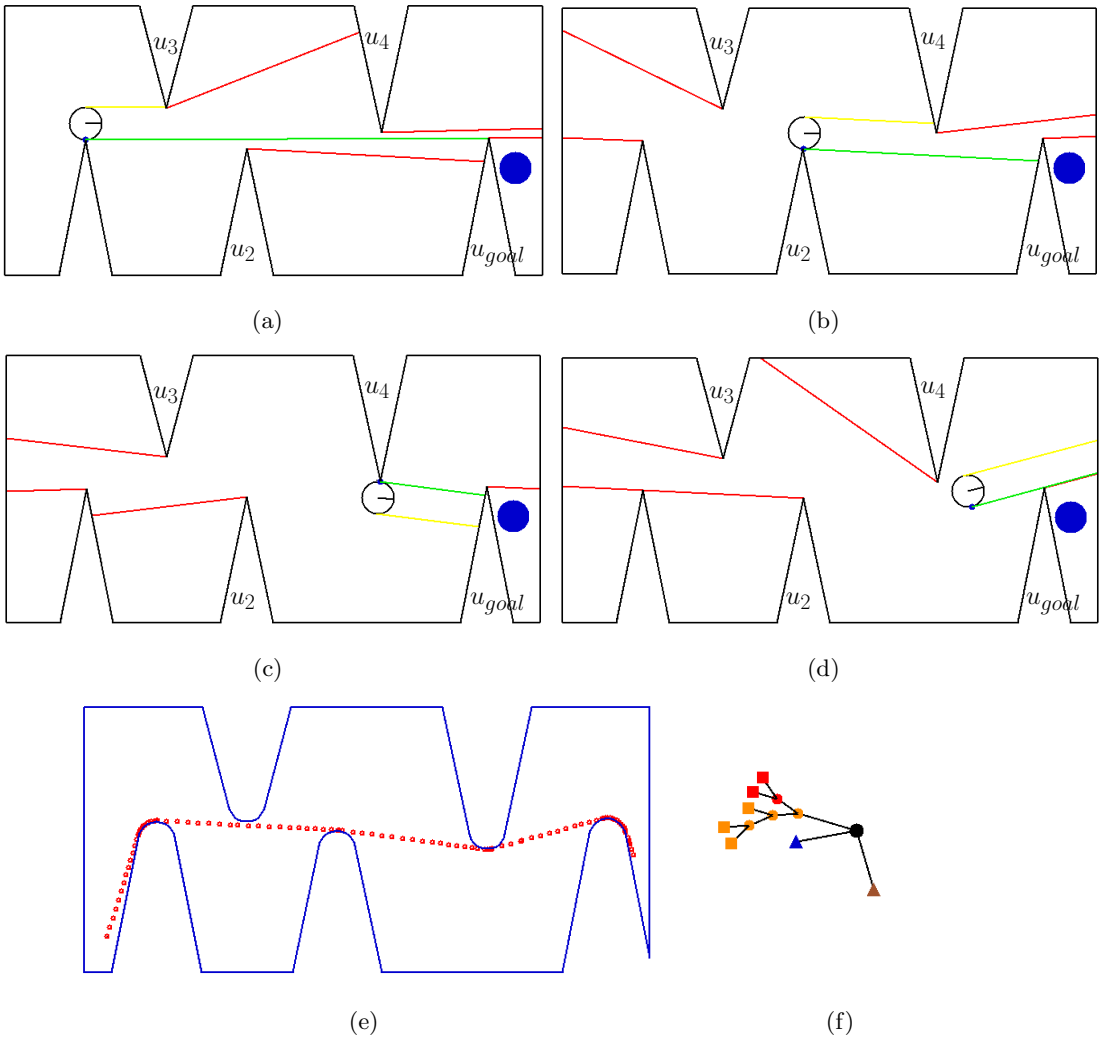


Figura 4.20: Ejemplo en el cual el primer vértice sub-meta es un vértice u_n cuando el robot está tocando ∂E con lp. Dicho vértice es u_2

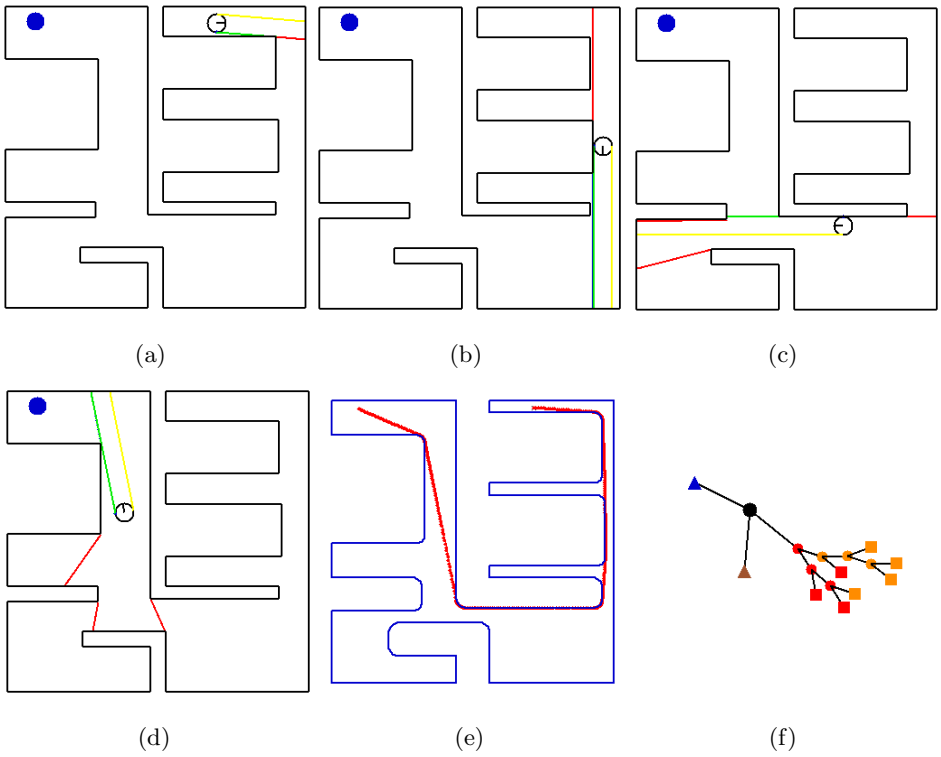


Figura 4.21: El robot se mueve en contacto con segmentos del entorno poligonal.

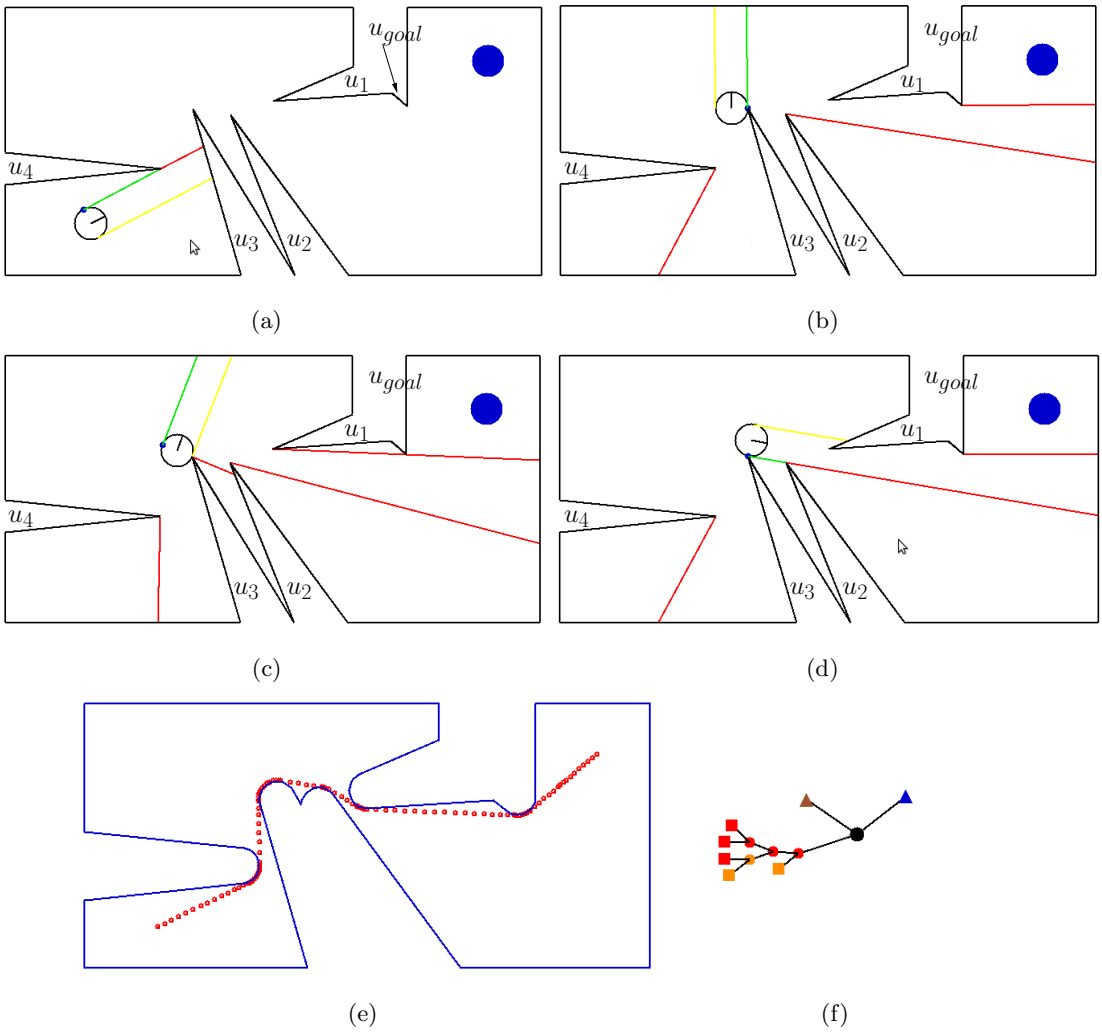


Figura 4.22: Un ejemplo de un vértice oculto, u_1 en la figura.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo extendimos el enfoque del GNT en [93] a un robot de manejo diferencial con forma de disco. El robot está equipado con sensores simples y es incapaz de construir mapas geométricos o ubicarse por sí mismo en un marco de referencia Euclidiano global. Este problema es considerablemente más retador que el caso para un robot punto debido a que la información de la visibilidad no corresponde exactamente a las trayectorias libres de colisión en el espacio de configuraciones. Consecuentemente, el robot debe ejecutar desvíos de la bitangente en el espacio de trabajo. De hecho, información critica del espacio de trabajo es obtenida de los sensores del robot, para inferir las trayectorias óptimas del robot en el espacio de configuraciones. Para resolver este problema desarrollamos una estrategia de movimiento basada en retroalimentación sensorial y después demostramos que la estrategia de movimiento produce movimientos óptimos globalmente en el sentido de la distancia Euclidiana caracterizando todas las posibles trayectorias en términos de la secuencia de estados visitados en una máquina de estados finitos. Dicha estrategia incluye una política de movimiento basada en retroalimentación sensorial. La política de movimiento está basada en el paradigma de evitar estimación del estado, dado que hay un mapeo directo entre observaciones y controles. Incluso si comparaciones precisas de distancia no son posibles, creemos que la estrategia de movimiento es efectiva en un escenario más amplio. Líneas importantes para trabajo futuro incluyen entornos múltiplemente conectados (entornos con agujeros) y cotas con respecto a optimalidad para los casos en que existe incertidumbre en el sensado y el control. También es interesante generar una ley de control retroalimentado en información sensorial para ejecutar la política de movimiento propuesta en esta tesis.

Apéndice A

A.1. Espacio de configuraciones $\mathcal{C}$

Una configuración de un robot es la especificación de las posiciones y orientaciones de todos los puntos del robot relativos a un sistema de coordenadas fijo.

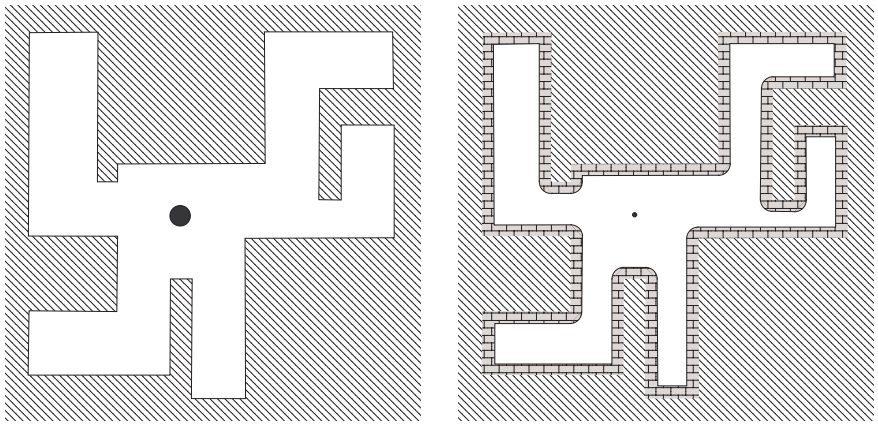
El espacio de configuraciones es el espacio de todas las posibles configuraciones del robot. En el espacio de configuraciones, cualquier robot sin importar su forma geométrica puede ser representado por un punto. $\mathcal{C}$ puede tener una topología no Euclidiana. El espacio de configuraciones se divide en el espacio de configuraciones libre $\mathcal{C}_{free}$ y el espacio de configuraciones obstáculo $\mathcal{C}_{obs}$.

Para un robot tipo disco, el espacio de configuraciones corresponde simplemente al polígono original (también llamado el espacio de trabajo), expandido por el radio del robot disco. En la Figura A.1(b), se muestra el espacio de configuraciones para un robot tipo disco. El espacio de trabajo es mostrado en las Figura A.1(a). La expansión de un entorno poligonal se calcula mediante una suma de Minkowski entre el disco que representa al robot y el polígono que representa el espacio de trabajo.

La suma de Minkowski corresponde a una convolución del robot disco con el polígono. Es decir, el robot disco se mantiene en contacto con la frontera del polígono y la curva que dibuja el centro del disco es la frontera entre el espacio de configuraciones obstáculo $\mathcal{C}_{obs}$ y el espacio de configuraciones libre $\mathcal{C}_{free}$.

A.2. Clase homotópica

Dos caminos con los mismos puntos terminales son homotópicos (pertenecen a la misma clase homotópica) si uno puede ser continuamente deformado en el otro. La Figura A.2(b)



(a) Espacio de trabajo. (b) Espacio de configuraciones.

Figura A.1: Espacio de configuraciones para un robot tipo disco.

muestra dos caminos que pertenecen a dos clases homotópicas diferentes.

A.3. Entorno simplemente conectado o múltiplemente conectado

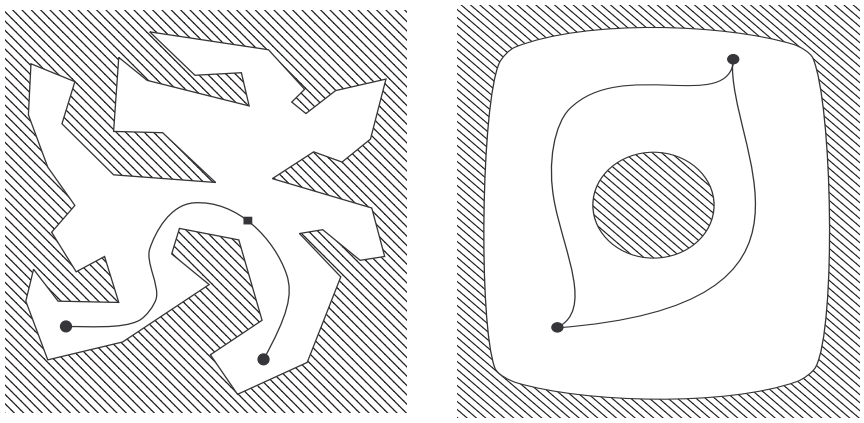
El espacio de configuraciones $\mathcal{C}$ es simplemente conectado si cualquier dos caminos conectando los mismos puntos terminales son homotópicos. Sino $\mathcal{C}$ es múltiplemente conectado. La Figura A.2(a) muestra un ambiente simplemente conectado y la Figura A.2(b) muestra un entorno múltiplemente conectado.

A.4. Vértice reflejo

Un vértice reflejo es un vértice de un polígono con un ángulo interno mayor que π. Vea la Fig A.3.

A.5. Grafo de visibilidad reducido

El grafo de visibilidad reducido (RVG por sus siglas en inglés) de un entorno poligonal, también conocido como el mapa de los caminos más cortos, considerando un robot puntual, es construido de la siguiente manera: en primer lugar, los vértices en el RVG son los vértices



(a) Ambiente simplemente conectado. (b) Entorno múltiplemente conectado.

Figura A.2: Ambiente simplemente conectado y múltiplemente conectado. La figura (b) muestra dos caminos que pertenecen a dos clases homotópicas diferentes.

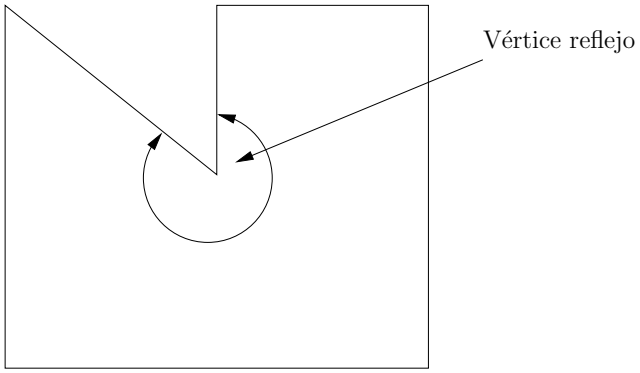
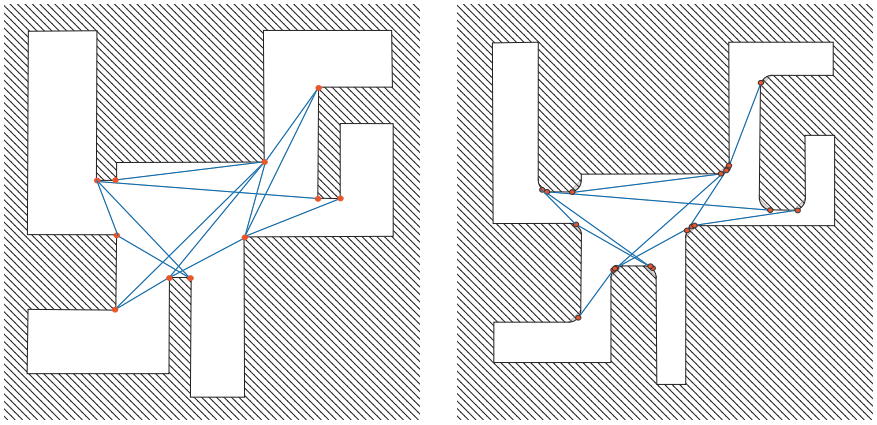


Figura A.3: Vértice reflejo.

reflejos del entorno. Por otro lado, la arista entre dos vértices en el RVG es generada si los dos vértices son puntos terminales de la misma frontera de un obstáculo, o si se puede dibujar una línea bitangente entre tales vértices. Para poder dibujar una línea bitangente entre dos vértices reflejos, los vértices necesitan ser mutuamente visibles el uno del otro, y la línea debe pasar tangencialmente por el entorno poligonal en ambos vértices.

El grafo de visibilidad reducido es usado para calcular las trayectorias más cortas entre cualesquiera dos puntos dentro del entorno poligonal, únicamente conectando los puntos al grafo y haciendo la búsqueda apropiada sobre el grafo. Algo importante para notar es que las trayectorias más cortas generadas usando el RVG son en contacto con los obstáculos. Para mayores detalles consulte [54]. La Figura A.4(a) muestra un ejemplo de un RVG

considerando a un robot puntual.



(a) Espacio de trabajo. (b) Espacio de configuraciones.

Figura A.4: Grafo de visibilidad reducido en: (a) el espacio de trabajo y (b) el espacio de configuraciones.

Apéndice B

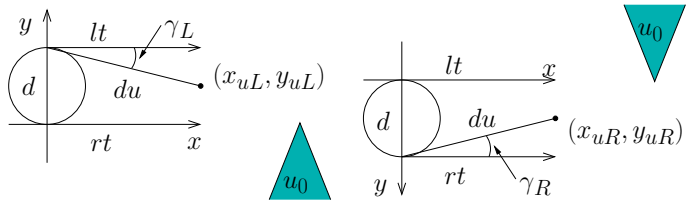
B.1. Marcos de referencia local para rotación en sitio

Para una rotación en sitio, son definidos dos marcos de referencia locales. Un marco de referencia está definido por el punto rp y un vértice derecho. El otro marco de referencia está definido por el punto lp y un vértice izquierdo.

El tipo de la brecha es relativo a la ubicación del sensor omnidireccional. Por lo tanto, el sensor omnidireccional debe estar ubicado en rp para observar las brechas derechas y en lp para ubicar las brechas izquierdas.

Consecuentemente, las ubicaciones de los vértices derechos con respecto al marco de referencia son calculadas con el sensor omnidireccional colocado en rp y las ubicaciones de los vértices izquierdos son calculadas con el sensor omnidireccional colocado en lp.

B.1.1. Ubicación de vértices



(a) Ubicación de un vértice izquierdo (x_{uL}, y_{uL}) sobre un marco de referencia local definido por rp y u_0.

(b) Ubicación de un vértice derecho (x_{uR}, y_{uR}) sobre un marco de referencia definido por lp y u_0.

Figura B.1: Ubicación de vértices sobre marcos de referencia local.

La Fig. B.1(a) muestra la ubicación de un vértice izquierdo (x_{uL}, y_{uL}) sobre un marco de referencia local $\mathcal{F}$ definido por el punto rp y un vértice derecho u_0. La Fig. B.1(b) muestra la ubicación de un vértice derecho (x_{uR}, y_{uR}) sobre un marco de referencia local $\mathcal{F}$ definido por el punto lp y un vértice izquierdo u_0.

La Ecuación B.1 indica las coordenadas de vértices izquierdos en un marco de referencia $\mathcal{F}$ definido por el punto rp y el vértice derecho u_0 basado en la distancia d_u y un ángulo γ_L, vea la Fig. B.1(a). La Ecuación B.2 indica las coordenadas en el mismo marco de referencia $\mathcal{F}$ de un vértice derecho hacia el cual el robot está alineado.

$$\begin{aligned} x_{uL} &= d_u \cos\gamma_L \\ y_{uL} &= d - d_u \sin\gamma_L \end{aligned} \tag{B.1}$$

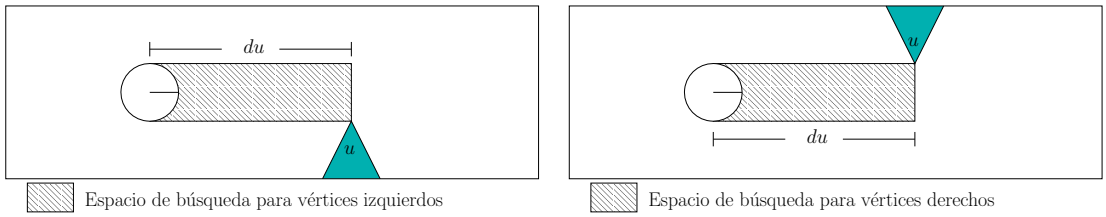
$$\begin{aligned} x_{uR} &= {d_R}^t \\ y_{uR} &= 0 \end{aligned} \tag{B.2}$$

donde d denota el diámetro y ${d_R}^t$ es la distancia desde rp hacia el vértice u_0 (rt está alineado a u_0). Cuando el marco de referencia $\mathcal{F}$ está definido por lp y un vértice izquierdo (vea la Fig. B.1(b)), hay otras ecuaciones similares para calcular las coordenadas de vértices derechos en estos marcos de referencia. En este caso el robot está alineado a un vértice izquierdo.

B.1.2. Espacios de búsqueda

La Fig. B.2 muestra los espacios de búsqueda para una rotación en sitio. El objetivo de un espacio de búsqueda es definir una región en el marco de referencia local $\mathcal{F}$ donde un vértice candidato debe estar.

La Fig. B.2(a) muestra el espacio de búsqueda en un marco de referencia $\mathcal{F}$ definido por el punto rp y un vértice derecho. Es en este espacio de búsqueda donde pueden estar ubicados los vértices izquierdos que pueden bloquear a un vértice derecho. Este espacio de búsqueda está delimitado por las direcciones rt y lt suponiendo que el robot está alineado con el vértice derecho. La Fig. B.2(b) muestra el espacio de búsqueda en un marco de referencia local $\mathcal{F}$ definido por el punto lp y un vértice izquierdo. Es en este espacio de búsqueda donde pueden estar ubicados los vértices derechos que pueden bloquear a un vértice izquierdo.



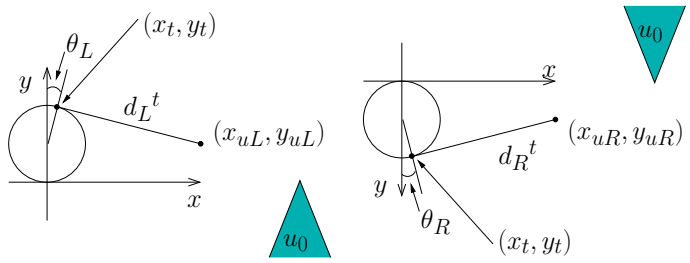
(a) Espacio de búsqueda en un marco de referencia local $\mathcal{F}$ definido por el punto rp y un vértice derecho.

(b) Espacio de búsqueda en un marco de referencia local $\mathcal{F}$ definido por el punto lp y un vértice izquierdo.

Figura B.2: Regiones en los marcos de referencia local (llamadas espacios de búsqueda) para una rotación en sitio.

B.1.3. Distancias y ángulos de alineamiento

Aquí, proporcionamos ecuaciones para calcular la distancia ${d_L}^t$ y el ángulo θ_L en un marco de referencia local $\mathcal{F}$ definido por rp y un vértice derecho.



(a) ${d_L}^t$ y θ_L son representados en la figura.

(b) ${d_R}^t$ y θ_R son representados en la figura.

Figura B.3: Distancias y ángulos de alineamiento para rotación en sitio.

La Fig. B.3(a) ilustra el ángulo θ_L que el robot debe rotar en sentido de las manecillas del reloj para alinear lt hacia el vértice izquierdo con coordenadas (x_{uL}, y_{uL}). θ_L es medido en sentido de las manecillas del reloj con respecto al eje y en el marco de referencia local $\mathcal{F}$ definido por rp y un vértice derecho. La Ecuación B.3 calcula θ_L.

$$\theta_L = \arctan\left(\frac{x_t}{y_t - r}\right) \tag{B.3}$$

donde r denota el radio del robot. El punto $(x_t,\ y_t)$ se obtiene con el procedimiento presentado en el Apéndice B.3. Este procedimiento está basado en un resultado clásico, el Teorema de Tales [85].

La Fig. B.3(a) muestra la distancia ${d_L}^t$, la cual es calculada suponiendo que la dirección particular lt está apuntando a un vértice izquierdo con coordenadas (x_{uL}, y_{uL}). La Ecuación B.4 calcula la distancia ${d_L}^t$.

$$ {d_L}^t = \sqrt{(x_{uL} - x_t)^2 + (y_{uL} - y_t)^2} \tag{B.4} $$

Cuando el marco de referencia $\mathcal{F}$ está definido por lp y un vértice izquierdo (vea la Fig. B.3(b)), existen ecuaciones similares para calcular el ángulo θ_R y la distancia ${d_R}^t$. El ángulo θ_R está calculado suponiendo que el ángulo se incrementa desde y hacia x (en sentido de las manecillas del reloj). La distancia ${d_R}^t$ está calculada suponiendo que la dirección particular rt está apuntando a un vértice derecho.

B.2. Marcos de referencia locales para cuando el robot rota con respecto al punto rp o lp

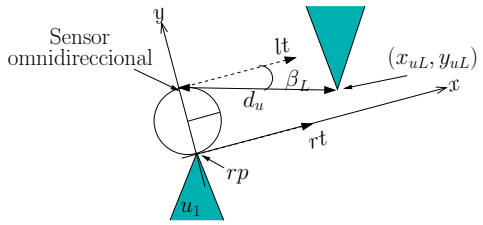
Para una rotación con respecto al punto rp o lp, también se definen dos marcos de referencia local. Un marco de referencia está definido por el punto rp y la dirección rt. El otro marco de referencia está definido por el punto lp y la dirección lt.

B.2.1. Ubicación de vértices

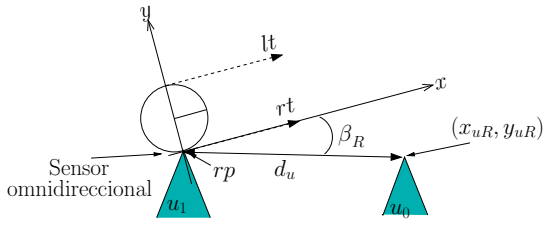
La Fig. B.4(a) muestra la ubicación de las coordenadas del vértice izquierdo (x_{uL}, y_{uL}), sobre un marco de referencia $\mathcal{F}$ definido por el punto rp y la dirección rt. La ubicación del vértice es calculada en base a la distancia d_u, y al ángulo β_L medido en sentido de las manecillas del reloj empezando desde la dirección particular lt. La Ecuación B.5 indica las coordenadas de vértices izquierdos en un marco de referencia $\mathcal{F}$ definido por el punto rp y la dirección rt. Vea la Fig. B.4(a).

$$ \begin{aligned} x_{uL} &= d_u \cos(\beta_L) \\ y_{uL} &= 2r - d_u \sin(\beta_L) \end{aligned} \tag{B.5} $$

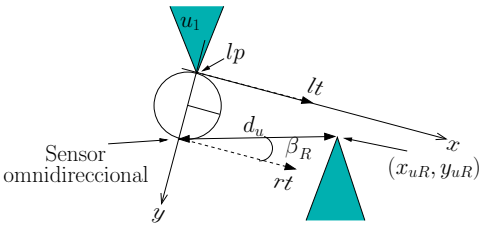
Las coordenadas de vértices derechos están dadas por la Ecuación B.6 (vea la Fig. B.4(b)).



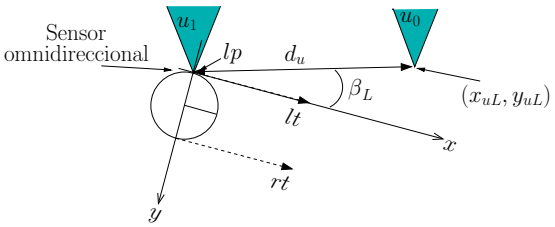
(a) Ubicación de un vértice izquierdo sobre un marco de referencia local $\mathcal{F}$ definido por el punto rp y la dirección rt.



(b) Ubicación de un vértice derecho sobre un marco de referencia local $\mathcal{F}$ definido por el punto rp y la dirección rt.



(c) Ubicación de un vértice derecho sobre un marco de de referencia local $\mathcal{F}$ definido por el punto lp y la dirección lt.



(d) Ubicación de un vértice izquierdo sobre un marco de referencia local $\mathcal{F}$ definido por el punto lp la dirección lt.

Figura B.4: Ubicación de vértice derechos e izquierdos sobre los marcos de referencia local $\mathcal{F}$ para cuando el robot rota con respecto al punto rp o lp.

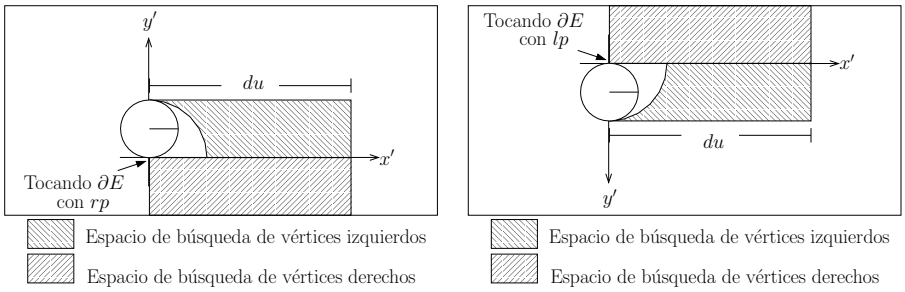
$$\begin{aligned} x_{uR} &= d_u \cos(\beta_R) \\ y_{uR} &= -d_u \sin(\beta_R) \end{aligned} \tag{B.6}$$

El ángulo β_L está medido en sentido de las manecillas de reloj empezando desde la dirección particular lt.

El sensor omnidireccional está ubicado en el punto lp para calcular las ubicaciones de vértices izquierdos y está ubicado en el punto rp para calcular las ubicaciones de vértices derechos. Esto se hace para correctamente *medir* los ángulos β_R y β_L. Existen ecuaciones equivalentes para indicar las coordenadas de vértices derechos e izquierdos sobre un marco de referencia definido por el punto lp y la dirección lt, vea las Figs. B.4(c) y B.4(d).

B.2.2. Espacios de búsqueda

La Fig. B.5 muestra los espacios de búsqueda para una rotación con respecto al punto rp o lp. El objetivo de un espacio de búsqueda es definir una región en el marco de referencia



(a) Espacio de búsqueda. (b) Espacio de búsqueda.

Figura B.5: Regiones en los marcos de referencia local (llamados espacios de búsqueda).

local donde el vértice candidato debe estar. La Fig. B.5(a) muestra el espacio de búsqueda para el caso donde el robot rotará con respecto al punto rp. La Fig. B.5(b) muestra el espació de búsqueda para el caso donde el robot rotará con respecto al punto lp. Note que en la Fig. B.4(a) la dirección rt que define el marco de referencia local $\mathcal{F}$ (vea la Sub-sección B.2.1) no está alineada con el vértice meta. Por lo tanto, una rotación del marco de referencia $\mathcal{F}'$ definido por el punto rp y la dirección rt es necesaria. Note que el robot está orientado de acuerdo al marco de referencia $\mathcal{F}$ como se muestra en Fig. B.4, pero se realiza una rotación virtual sobre $\mathcal{F}$ para obtener $\mathcal{F}'$. Esto se hace para establecer el espacio de búsqueda como si el robot estuviera alineado al vértice meta. De hecho, se dibuja el espacio de búsqueda sobre el marco de referencia $\mathcal{F}'$ definido por x' y y'. Todo esto se realiza para determinar los vértices que pueden bloquear al vértice meta.

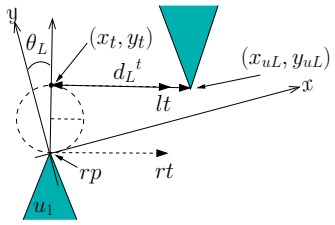
B.2.3. Distancias y ángulos de alineamiento

La Ecuación B.7 calcula el ángulo de rotación del robot θ_L con respecto al eje y del marco de referencia local $\mathcal{F}$; equivalente al ángulo necesario para alinear la dirección particular lt a un vértice izquierdo ejecutando una rotación del robot en sentido de las manecillas del reloj con respecto al punto rp (Vea la Fig. B.6(a)).

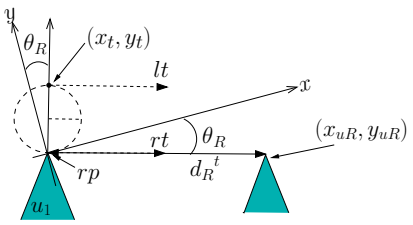
$$\theta_L = \arctan\left(\frac{x_t}{y_t}\right) \tag{B.7}$$

De nuevo, el punto $(x_t,\, y_t)$ se obtiene basado en el Teorema de Tales [85].

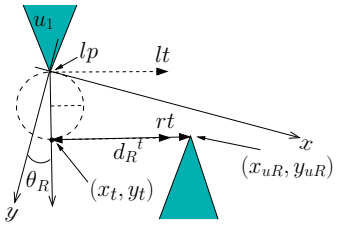
La Ecuación B.8 calcula la distancia ${d_L}^t$. La Fig. B.6(a) muestra la distancia ${d_L}^t$. Para calcular la distancia, se asume que la dirección lt apunta al vértice izquierdo asociado con



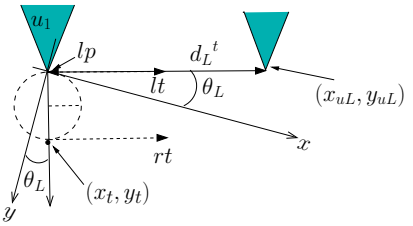
(a) El ángulo de rotación del robot θ_L (éste es el ángulo de rotación para alinear lt con un vértice izquierdo) y la distancia $d_L{}^t$.



(b) El ángulo de rotación del robot θ_R (éste es el ángulo de rotación para alinear rt con un vértice derecho) y la distancia $d_R{}^t$.



(c) El ángulo de rotación del robot θ_R (éste es el ángulos de rotación para alinear rt con un vértice derecho) y la distancia $d_R{}^t$.



(d) El ángulo de rotación del robot θ_L (éste es el ángulo de rotación para alinear lt con un vértice izquierdo) y la distancia $d_L{}^t$.

Figura B.6: El ángulo de rotación del robot tocando ∂E en el punto rp o lp: distancias y ángulos de alineamiento usado para encontrar un vértice candidato.

$d_L{}^t$.

$$d_L{}^t = \sqrt{(x_{uL} - x_t))^2 + (y_{uL} - y_t)^2} \tag{B.8}$$

$d_R{}^t$ es medido directamente por el sensor omnidireccional (Vea la Fig. B.6(b)). θ_R también es medido directamente por el sensor omnidireccional (Vea la Fig. B.6(b)). Existen ecuaciones totalmente análogas para calcular θ_R, θ_L, $d_R{}^t$ y $d_L{}^t$ sobre un marco de referencia local definido por el punto lp y la dirección lt. Refiérase a las Figs. B.6(c) y B.6(d).

B.3. Obteniendo el punto $T(x_t, y_t)$

Refiérase a la Fig. B.7. Dada una circunferencia k y un punto conocido $P(x_p, y_p)$, queremos encontrar un punto $T(x_t, y_t)$, tal que, exista una línea tangente a k en el punto T y que pasa por el punto P.

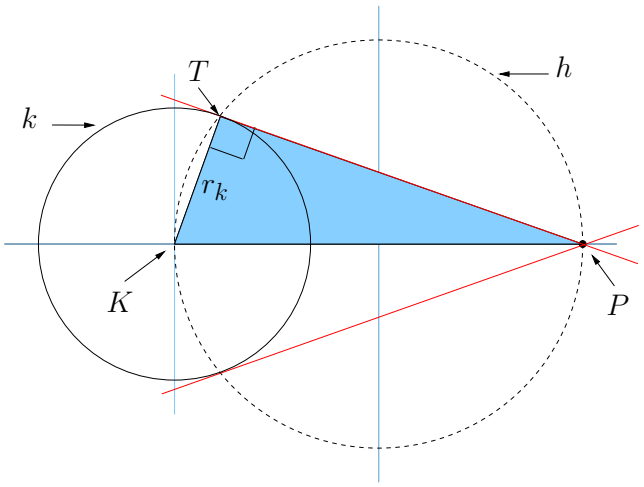


Figura B.7: Punto $T(x_t, y_t)$.

La Ecuación B.9 o la Ecuación B.10 define la circunferencia k.

El punto $K(x_k, y_k)$ es el centro de la circunferencia k y r_k es el radio.

$$(x - x_k)^2 + (y - y_k)^2 = {r_k}^2 \tag{B.9}$$

$$x^2 + y^2 + Ax + By + C = 0 \tag{B.10}$$

donde $A = -2x_k$, $B = -y_k$ y $C = {x_k}^2 + {y_k}^2 - {r_k}^2$. El triangulo rectángulo KTP puede ser inscrito en una circunferencia h que tiene un radio igual a $1/2$ la hipotenusa KP del triangulo -Teorema de Tales [85]-. Análogamente, la Ecuación B.11 o la Ecuación B.12 define la circunferencia h.

El punto $H(x_h, y_h)$ es el centro de la circunferencia h y r_h es el radio.

$$(x - x_h)^2 + (y - y_h)^2 = {r_h}^2 \tag{B.11}$$

donde $x_h = \frac{x_k + x_p}{2}$, $y_h = \frac{y_k + y_p}{2}$ y $r_h = \frac{\sqrt{(x_k - x_p)^2 + (y_k - y_p)^2}}{2}$.

$$x^2 + y^2 + Dx + Ey + F = 0 \tag{B.12}$$

y donde $D = -2x_h$, $E = -y_h$ y $F = {x_h}^2 + {y_h}^2 - {r_h}^2$.

Resolviendo el sistema de las Ecuaciones B.10 y B.12, es posible encontrar el punto de intersección $T(x_t, y_t)$ de las circunferencias k y h.

Así obtenemos:

$$x_t(A - D) + y_t(B - E) + (C - F) = 0 \tag{B.13}$$

$$x_t = -Hy_t - I \tag{B.14}$$

donde $H = \frac{B-E}{A-D}$ y $I = \frac{C-F}{A-D}$.

Substituyendo B.14 en B.10, obtenemos:

$$y_t^2(H^2 + 1) + y_t(2HI - AH + B) + I^2 - AI + C = 0 \tag{B.15}$$

y

$$y_t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{B.16}$$

donde $a = H^2 + 1$, $b = 2HI - AH + B$ y $c = I^2 - AI + C$.

Bibliografía

[1] F. Amigoni. Experimental evaluation of some exploration strategies for mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2008*, pages 2818–2823, 2008.

[2] F. Amigoni and V. Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.

[3] F. Amigoni, S. Gasparini, and M. Gini. Building segment-based maps without pose information. *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 94(7):1340–1359, 2006.

[4] K. O. Arras, J. Persson, N. Tomatis, and R. Siegwart. Real-time obstacle avoidance for polygonal robots with a reduced dynamic window. In *ICRA*, pages 3050–3055. IEEE, 2002.

[5] N. Ayache and O. Faugeras. Building, registrating, and fusing noisy visual maps. *Int. J. Rob. Res.*, 7(6):45–65, 1988.

[6] K. Azarm and G. Schmidt. Integrated mobile robot motion planning and execution in changing indoor environments. In *IROS*, pages 298–305. IEEE, 1994.

[7] D. J. Balkcom and M. T. Mason. Time optimal trajectories for bounded velocity differential drive vehicles. *I. J. Robotic Res*, 21(3):199–218, 2002.

[8] H. M. Becerra, G. López-Nicolás, and C. Sagüés. Omnidirectional visual control of mobile robots based on the 1D trifocal tensor. *Robotics and Autonomous Systems*, 58(6):796–808, 2010.

[9] H. M. Becerra and C. Sagüés. Exploiting the trifocal tensor in dynamic pose estimation for visual control. *IEEE Trans. Contr. Sys. Techn*, 21(5):1931–1939, 2013.

[10] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson. Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints. *IEEE Transactions on Robotics*, 23(1):47–59, 2007.

[11] A. Bicchi, G. Casalino, and C. Santilli. Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles. *Journal of Intelligent and Robotic Systems*, 16(4):387–405, 1996.

[12] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1179–1187, 1989.

[13] J. Borenstein and Y. Koren. The vector field histogram: Fast obstacle avoidance for mobile robots. *IEEE Trans. Robotics and Automation*, 7:278–288, 1991.

[14] A. J. Briggs, C. Detweiler, D. Scharstein, and A. Vandenberg-Rodes. Expected shortest paths for landmark-based robot navigation. *I. J. Robotic Res*, 23(7-8):717–728, 2004.

[15] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *In IEEE Int. Conf. on Robotics and Automation*, pages 341–346, 1999.

[16] O. Brock and O. Khatib. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *ICRA*, pages 550–555. IEEE, 2000.

[17] H. Bulata and M. Devy. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *Proc. Conf. IEEE Int Robotics and Automation*, volume 2, pages 1054–1060, 1996.

[18] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21:376–386, 2005.

[19] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 49–60, 1987.

[20] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 2, pages 138–145, 1985.

[21] D. Z. Chen and H. Wang. Paths among curved obstacles in the plane. In *Proceedings of Computing Research Repository*, 2011.

[22] L. P. Chew. Planning the shortest path for a disc in $O(n^2 logn)$ time. In *Proceedings ACM Symposium on Computational Geometry*, 1985.

[23] H. Choset and J. Burdick. Sensor based motion planning: The hierarchical generalized voronoi graph. In *Workshop on Algorithmic Foundations of Robotics*, 1996.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press, June 2005.

[25] W. Cohen. Adaptive mapping and navigation by teams of simple robots. *Journal of Robotics & Autonomous Systems*, 18:411–434, 1996.

[26] J. L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proc. Conf. IEEE Int Robotics and Automation*, pages 674–680, 1989.

[27] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Ed.* Springer-Verlag, Berlin, 2000.

[28] A. W. Divelbiss and J. T. Wen. A path space approach to nonholonomic motion planning in the presence of obstacles. *IEEE T. Robotics and Automation*, 13(3):443–451, 1997.

[29] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[30] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Transactions on Robotics and Automation*, 3(3):249–265, 1987.

[31] W. Feiten, R. Bauer, and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. In E. Straub and R. S. Sipple, editors, *Proceedings of the International Conference on Robotics and Automation. Volume 3*, pages 2412–2417, Los Alamitos, CA, USA, May 1994. IEEE Computer Society Press.

[32] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[33] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robot. Automat. Mag*, 4(1):23–33, 1997.

[34] S. K. Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, Cambridge, U.K., 2007.

[35] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 11–19, 1987.

[36] J. Gonzalez, A. Ollero, and A. Reina. Map building for a mobile robot equipped with a 2d laser rangefinder. In *Proc. Conf. IEEE Int Robotics and Automation*, pages 1904–1909, 1994.

[37] H. Gonzalez-Banos, E. Mao, J. Latombe, T. M. Murali, and A. Efrat. Planning robot motion strategies for efficient model construction. In *Robotics Research - The 9th International Symposium*, pages 345–352, 1999.

[38] H. H. Gonzalez-Baños and J.-C. Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[39] J.-B. Hayet, C. Esteves, and R. Murrieta-Cid. A motion planner for maintaining landmark visibility with a differential drive robot. In H. Choset, M. Morales, and T. D. Murphey, editors, *WAFR*, volume 57 of *Springer Tracts in Advanced Robotics*, pages 333–347. Springer, 2008.

[40] J.-B. Hayet, C. Esteves, and R. Murrieta-Cid. Motion planning for maintaining landmarks visibility for a differential drive robot. Accepted to Journal Robotics and Autonomous Systems, 2013.

[41] J. B. Hayet, F. Lerasle, and M. Devy. A visual landmark framework for mobile robot navigation. *Image and Vision Computing*, 25(8):1341–1351, Aug. 2007.

[42] M. Hebert, C. E. Thorpe, and A. Stentz. *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*. Kluwer, Dec. 1996.

[43] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *Proc. Conf. IEEE Int Robotics and Automation*, pages 1722–1727, 1991.

[44] V. Isler, D. Sun, and S. Sastry. Roadmap based pursuit-evasion and collision avoidance. In S. Thrun, G. S. Sukhatme, and S. Schaal, editors, *Robotics: Science and Systems*, pages 257–264. The MIT Press, 2005.

[45] M. Khatib. Sensor-based motion control for mobile robots, 1996.

[46] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[47] B. Krogh and C. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation (ICRA '86)*, pages 1664 – 1669, April 1986.

[48] B. Kuipers, R. Froom, W.-Y. Lee, and D. Pierce. The semantic hierarchy in robot learning. In *Robot Learning*, pages 141–170. Kluwer Academic Publishers, 1993.

[49] G. Laguna, R. Murrieta-Cid, H. Becerra, R. Lopez-Padilla, and S. M. LaValle. Exploration of an unknown environment with a differential drive disc robot. In *To appear in IEEE Int. Conf. on Robotics & Automation*, 2014.

[50] Y. Landa and R. Tsai. Visibility of point clouds and exploratory path planning in unknown environments. *Communications in Mathematical Sciences*, 6(4):881–913, Dec. 2008.

[51] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[52] J. Laumond. *Robot motion planning and control*. Lectures Notes in Control and Information Sciences 229. Springer, N.ISBN 3-540-76219-1, 1998.

[53] J.-P. Laumond, P. E. Jacobs, M. Taïx, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE T. Robotics and Automation*, 10(5):577–593, 1994.

[54] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.

[55] S. M. LaValle. Sensing and filtering: A tutorial based on preimages and information spaces. *Foundations and Trends in Robotics*, 2011. To appear.

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

[57] Y.-H. Liu and S. Arimoto. Finding the shortest path of a disc among polygonal obstacles using a radius-independent graph. *Robotics and Automation, IEEE Transactions on*, 11(5):682 –691, oct 1995.

[58] G. López-Nicolás, N. R. Gans, S. Bhattacharya, C. Sagüés, J. J. Guerrero, and S. Hutchinson. Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 40(4):1115–1127, 2010.

[59] G. López-Nicolás, J. J. Guerrero, and C. Sagüés. Visual control through the trifocal tensor for nonholonomic robots. *Robotics and Autonomous Systems*, 58(2):216–226, 2010.

[60] R. Madhavan and H. F. Durrant-Whyte. Natural landmark-based autonomous vehicle navigation. *Robotics and Autonomous Systems*, 46(2):79–95, 2004.

[61] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte. An experiment in integrated exploration. In *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems IROS*, pages 534–539, 2002.

[62] A. A. Masoud, S. A. Masoud, and M. M. Bayoumi. Robot navigation using a pressure generated mechanical stress field, the biharmonic potential approach. In E. Straub and R. S. Sipple, editors, *Proceedings of the International Conference on Robotics and Automation. Volume 1*, pages 124–130, Los Alamitos, CA, USA, May 1994. IEEE Computer Society Press.

[63] P. Michel, C. Scheurer, J. J. Kuffner, N. Vahrenkamp, and R. Dillmann. Planning for robust execution of humanoid motions using future perceptive capability. In *IROS*, pages 3223–3228. IEEE, 2007.

[64] J. Minguez and L. Montano. Nearness diagram davigation (nd): Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics & Automation*, 20(1):45–59, 2004.

[65] J. S. B. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 607–641. Chapman and Hall/CRC Press, New York, 2004.

[66] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem. *Journal of the ACM*, 38:18–73, 1991.

[67] L. Montano and J. R. Asensio. Real-time robot navigation in unstructured environments using a 3D laser rangefinder. In *IROS*, pages 526–532. IEEE, 1997.

[68] L. Murphy and P. Newman. Using incomplete online metric maps for topological exploration with the gap navigation tree. In *Proceedings IEEE International Conference on Robotics & Automation*, 2008.

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

[70] P. M. Newman, M. Bosse, and J. J. Leonard. Autonomous feature-based exploration. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA '03*, volume 1, pages 1234–1240, 2003.

[71] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In L. Werner, Robert; O'Conner, editor, *Proceedings of the 1993 IEEE International Conference on Robotics and Automation: Volume 2*, pages 802–807, Atlanta, GE, May 1993. IEEE Computer Society Press.

[72] P. Ranganathan, J.-B. Hayet, M. Devy, S. Hutchinson, and F. Lerasle. Topological navigation and qualitative localization for indoor environment using multi-sensory perception. *Robotics and Autonomous Systems*, 41(2-3):137–144, 2002.

[73] J. H. Reif and Z. Sun. An efficient approximation algorithm for weighted region shortest path problem. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 191–203. A.K. Peters, Wellesley, MA, 2001.

[74] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *In Advances in Neural Processing Systems 12*, pages 1043–1049, 1999.

[75] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[76] S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, 2005.

[77] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *IROS*, pages 1060–1065. IEEE, 1998.

[78] R. Sim and G. Dudek. Effective exploration strategies for the construction of visual maps. In *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 69–76. Press, 2003.

[79] R. Sim and G. Dudek. Online control policy optimization for minimizing map uncertainty during exploration. In *In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1758–1763, 2004.

[80] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *ICRA*, pages 3375–3382. IEEE, 1996.

[81] R. Simmons, D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Austin, TX, 2000. AAAI.

[82] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proc. Conf. IEEE Int Robotics and Automation*, pages 254–259, 1993.

[83] P. Souères and J. Laumond. Shortest paths synthesis for a car-like robot. *IEEE Transaction on Automatic Control*, 41(05):672–688, May 1996.

[84] C. Stachniss, O. M. Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA 2006*, pages 1692–1697, 2006.

[85] J. Stillwell. *The Four Pillars of Geometry*. Springer, 2005.

[86] S. Teller. Automated urban model acquisition: Project rationale and status. In *DARPA98*, pages 455–462, 1998.

[87] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[88] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[89] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proc. IEEE Int Robotics and Automation Conf*, volume 2, pages 1546–1551, 1998.

[90] R. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 566–571 vol.1, May 1990.

[91] B. Tovar, R. Murrieta-Cid, and C. Esteves. Robot motion planning for model building under perception constraints. In *International Symposium on Intelligent Robotic Systems*, pages 447–456, 2001.

[92] B. Tovar, R. Murrieta-Cid, and C. Esteves. Robot motion planning for map building. In *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, volume 1, pages 673–680, 2002.

[93] B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, June 2007.

[94] B. Tovar, L. Muñoz, R. Murrieta-Cid, M. Alencastre, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Journal on Robotics and Autonomous Systems*, 54(4):314–331, April 2006.

[95] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-98)*, pages 1572–1577, Piscataway, May 16–20 1998. IEEE Computer Society.

[96] H. Wang, Y. Chen, and P. Souères. A geometric algorithm to compute time-optimal trajectories for a bidirectional steered robot. *IEEE Transactions on Robotics*, 25(2):399–413, 2009.

[97] K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2008*, pages 1160–1165, 2008.

[98] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. Symp. IEEE Int Computational Intelligence in Robotics and Automation CIRA'97.*, pages 146–151, 1997.