CIMAT

Centro de Investigación en Matemáticas, A.C.

Una Estrategia de Movimiento para Exploración Dirigida por un Autómata Activando Controladores Basados en Retroalimentación

T E S I S

Que para obtener el grado de

Doctor en Ciencias

con Orientación en

Ciencias de la Computación

Presenta

M. I. Edgar Daniel Martínez Rodríguez

Director de Tesis:

Dr. Rafael Murrieta Cid

Dr. Héctor M. Becerra

Guanajuato, Gto., 12 de Noviembre de 2018

CIMAT

Centro de Investigación en Matemáticas, A.C.

Una Estrategia de Movimiento para Exploración Dirigida por un Autómata Activando Controladores Basados en Retroalimentación

T E S I S

Que para obtener el grado de

Doctor en Ciencias

con Orientación en

Ciencias de la Computación

Presenta

M. I. Edgar Daniel Martínez Rodríguez

Directores de Tesis:

Dr. Rafael Murrieta Cid

Dr. Héctor M. Becerra

Autorización de la versión final

Guanajuato, Gto., 12 de Noviembre de 2018

Dedicatoria

A Dios: Por estar presente en cada etapa de mi vida y ayudarme a lograr mis metas.

A mis padres: Ramiro Martínez y Ma. Dolores Rodríguez, por brindarme su apoyo incondicional, por sus sabios consejos y enseñanzas que hoy son parte de mi formación personal y profesional.

A mis hermanos: Oscar Martínez y Diana Martínez, por ser mis amigos de toda la vida y brindarme su apoyo y confianza.

Agradecimientos

A mis asesores: Dr. Rafael Murrieta Cid y Dr. Héctor M. Becerra, por su ayuda, guía y enseñanzas durante el transcurso de mi doctorado.

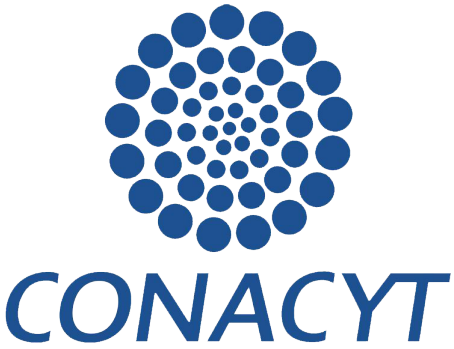
A mis tíos: Jesús, Carlos, Emma, Silvia, Rosalva, Miguel y Verónica por su apoyo incondicional.

A mis abuelos: Esperanza González, Pascual Martínez †, Belén Rodríguez †, que con su experiencia en la vida me han enseñado a no rendirme y a nunca perder la esperanza.

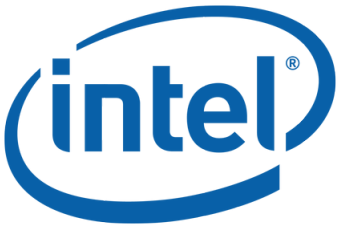
A mis amigos: Araceli, Omar, Noé y Orlando por su valiosa amistad y apoyo a lo largo del doctorado.

Agradecimientos Institucionales

Al **Consejo Nacional de Ciencia y Tecnología** (**CONACyT**), por el apoyo que me brindó al otorgarme la beca para posgrado.



A **Intel Corporation** por apoyarme con una beca de proyecto.



Resumen

Esta tesis aborda el problema de exploración de un ambiente poligonal desconocido, el cual es simplemente conexo y cuyo espacio de configuraciones puede tener varios componentes conexos. Para explorar el entorno, el robot sigue la frontera del ambiente.

Primero, se propone una estrategia de movimiento basada en retroalimentación sensorial, la cual se modela como una máquina de Moore. Dicha estrategia propone el paradigma de evitar la estimación del estado del robot. Hay una correspondencia directa entre observaciones y controles, en esta instancia se consideran mediciones y acciones de control perfectas. Se presentan las condiciones teóricas, para que el robot descubra con su sensor, la región más grande posible del ambiente.

Después se diseña un enfoque que integra planificación y control, el cual es capaz de lidiar con observaciones imperfectas, actuadores ruidosos y toma en cuenta las variaciones de velocidad del robot. Se propone un autómata que filtra observaciones espurias, las observaciones válidas restantes activan controladores basados en retroalimentación. El método de control tiene como consigna mantener una pequeña distancia entre el robot y la frontera del ambiente, dicho método conmuta controladores de acuerdo a las observaciones obtenidas por los sensores y es capaz de mantener la continuidad de las velocidades lineal y angular del robot a pesar de la conmutación entre controladores.

Finalmente, se proponer un controlador de modos deslizantes que tiene como propiedad alcanzar la consigna de control en tiempo finito, lo que permite mover al robot sin detenerse al encontrar un obstáculo. Además, en este enfoque sólo las referencias de control cambian con los estados en el autómata, el controlador es el mismo en todos los estados. Todos los métodos propuestos fueron implementados y se presentan simulaciones y experimentos en robots físicos.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

En la actualidad, la robótica se ha integrando paulatinamente en nuestra sociedad creando sistemas autónomos cuya finalidad es la de efectuar desde simples tareas cotidianas hasta la ejecución de complejos procesos que pueden implicar un cierto grado de riesgo y dificultad para algún ser humano. Estos sistemas pueden realizar diversas actividades con mayor precisión y efectuándolas durante largas jornadas de trabajo. Las capacidades de los sistemas robóticos dependen de la complejidad de la tarea y las condiciones ambientales en las que se deben desempeñar. Para ello, los robots son dotados de una variedad de sensores que proporcionan información necesaria para efectuar correctamente la tarea a ejecutar. Un siguiente paso es el desarrollo de algoritmos que puedan aprovechar la información obtenida de la etapa de sensado y para que el robot pueda desempeñarse de manera adecuada en su ambiente de trabajo.

Algunos de los problemas más estudiados en el área de la robótica son la navegación autónoma evitando colisiones con los obstáculos en el ambiente y la exploración de entornos desconocidos, con la finalidad de descubrir la mayor parte de un ambiente o ubicar objetos o *landmarks* en el ambiente. Ambos problemas de navegación y exploración están relacionados y en complemento pueden tener grandes alcances.

1.1. Problemas de navegación y exploración robótica

El trabajo que se presenta en esta tesis está relacionado al problema de planificación de trayectorias de robots que evitan colisionar con obstáculos [2, 3, 4], y particularmente con robots noholonómicos [5, 6, 7]. Además, este trabajo también se relaciona con el problema de exploración de entornos desconocidos [8, 9, 10, 11, 12] para obtener una representación

útil para otras tareas, por ejemplo la búsqueda de objetos [13, 14].

La tarea de exploración de entornos desconocidos planos, ha sido tratada en algunos trabajos previos [15, 16, 17, 13, 18, 19]; algunos de estos usan un modelo simple donde el robot móvil es considerado como un punto. Desde un punto de vista teórico, este enfoque ha permitido resolver algunos problemas de navegación con robots; sin embargo, para tareas más realistas, este enfoque no es suficiente. Modelar al robot como un punto ignora las dimensiones físicas y esta suposición puede impactar negativamente su desempeño. Un paso natural a dar, y más realista, es considerar una entidad de tamaño diferente de cero. Una forma de disco es la más simple. El tamaño del robot representa restricciones adicionales en el espacio de configuraciones. Esto plantea la principal diferencia conceptual entre un robot punto y un robot con forma de disco, lo cual hace necesario el diseño de una estrategia de exploración específica para el robot con forma de disco. En efecto, el concepto de visibilidad es equivalente al concepto de alcanzabilidad para un robot punto. Esto significa que si el robot puede ver un cierto lugar dentro del entorno, este lugar es alcanzable por el robot. Sin embargo, esta propiedad no es necesariamente cierta para un robot con forma de disco. Si el mapa original es totalmente conocido, construir el espacio de configuraciones para un robot con forma de disco es fácil, dicho mapa es expandido por el diámetro del robot. Sin embargo, el espacio de configuraciones resultante no es observable, el robot no puede medirlo con sus sensores. Por otra parte, en el problema de exploración el entorno es desconocido, para resolver dicho problema, el robot disco debe inferir información del espacio de configuraciones desde el espacio de trabajo usando el sensado. En este documento, se propone una nueva estrategia de exploración para ambientes desconocidos, planos y poligonales usando un robot en forma de disco.

1.2. Objetivo general y planteamiento del problema

Este trabajo tiene como objetivo generar estrategias de movimiento para explorar un ambiente desconocido, con un sistema noholonomico, en particular un robot de manejo diferencial.

Explorar el ambiente significa que mientras que el robot se mueve, la región de visibilidad del sensor debe cubrir al entorno al menos una vez, o en el peor de los casos cubrir la mayor región posible de ambiente. Consecuentemente, el robot encontrará un objeto estático o "*landmark*" o podrá declarar que no existe una estrategia de exploración para encontrar dicha *landmark*.

Se busca que dichas estrategias de movimiento sean capaces de lidiar con incertidumbre tanto en el sensado como en el control, además que sean capaces de generar controles adecuados al sistema, donde las velocidades del robot no sobrepasen las capacidades del robot y que no cambien discontinuamente de valor.

A continuación, en el Capítulo 2, se presenta la diversidad de los esquemas de navegación y exploración que conforman el trabajo previo de esta tesis.

Capítulo 2

Trabajo Previo

En este capítulo se presentan las tareas de navegación y exploración, las cuales han sido estudiadas previamente bajo diversos esquemas. Ambas tareas se relacionan fuertemente debido a que la navegación libre de colisiones es un componente esencial en las tareas de exploración con robots. A continuación, se describe brevemente y por separado los diversos esquemas propuestos en el trabajo previo que están relacionados con las tareas de navegación sin colisión y exploración.

2.1. Navegación sin colisión

El trabajo propuesto en el presente documento está relacionado con el problema de planeación de trayectorias del robot para evitar la colisión con obstáculos [2], [3], [4] y particularmente con robots noholonómicos, robots con restricciones de movimiento, un ejemplo clásico de un sistema noholonónico es un automóvil, el cual no puede moverse instantáneamente en la dirección perpendicular a donde apuntan sus ruedas [5], [6], [7].

En [4], se describe un método de navegación reactiva que usa una estrategia de "divide y vencerás" para simplificar el nivel de dificultad de la navegación. El diseño del método se realiza a nivel simbólico, donde se define un conjunto de situaciones y acciones asociadas a cada situación. Durante la etapa de ejecución, se utiliza la percepción para identificar cuál es la situación actual y así realizar la acción a la cual está asociada. También, 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 determinar los comandos de movimiento, donde se aplican ecuaciones tomadas de la física a la información sensorial y la

solución es transformada en comandos de movimiento. Por ejemplo: métodos de campos potenciales [2], [20], [21], [22], [3], [23], la analogía del perfume (difusión inestable) [24], y la analogía del fluido [25], entre otras.

En algunos otros métodos se determina 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 [26], [27], [28], [29], y otros calculan un conjunto de comandos de velocidad [30], [31], [32], [33].

Otros métodos calculan una forma de describir la información de alto nivel desde la información sensorial para obtener un comando de movimiento [22], [34], [35]. 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 ejecutar una acción que calcule el movimiento.

Existen esquemas relacionados con planificación de movimientos con restricciones de visibilidad [36], [37], planificación de movimientos con restricciones noholonómicas [5], [6], [38], [39], [40], y navegación basada en *landmarks* [41], [42], [43], [44], [45], [46].

La planificación de movimientos con restricciones noholonómicas es un campo de investigación muy activo (un buen panorama es presentado en [39]). Los resultados más relevantes en este campo han sido obtenidos abordando el problema mediante el uso de herramientas de geometría diferencial y la teoría de control. Laumond, pionero de esta investigación, mostró que una trayectoria libre para un robot holonómico moviéndose entre obstáculos en un espacio de trabajo 2D, siempre puede transformarse en una trayectoria factible para un robot noholonómico realizando diversas maniobras [5].

El trabajo presentado en [6] describe una técnica para la planificación de trayectorias para robots móviles noholonómicos en entornos con obstáculos. Este trabajo usa primitivas de movimiento para construir un diagrama de trayectoria básico, similar al grafo de visibilidad [47], para vehículos noholonómicos.

En [48], se presenta un algoritmo para encontrar trayectorias factibles para un sistema noholonómico en presencia de obstáculos. En primer lugar, el problema de planificación de la trayectoria sin presencia de obstáculo se transforma en un problema de mínimos cuadrados no lineal en un espacio aumentado. La evasión de obstáculos se incluye como restricciones de desigualdad y se presentan los resultados en simulación para el caso de un tractor con remolque.

Lazanas [41] presenta uno de los primeros trabajos de la navegación robótica basada en

landmarks. La visibilidad geométrica no está integrada de manera explícita; en vez de ello, cada *landmark* define una "zona de seguridad" 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 presencia de incertidumbre. En [49], los *landmarks* 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 *landmarks*. En el trabajo [42], se propone un enfoque de navegación basado en *landmarks* que generan trayectorias llamado "navegación costera". En una tarea de localización robótica probabilística, una ventaja importante de un enfoque basado en *landmarks* es que si los *landmarks* son detectados, estas detienen el crecimiento incremental de incertidumbre en la posición del robot [42], [43], [44], [46], [50].

Como ya se mencionó, el uso de *landmarks* para la navegación y localización ha sido ampliamente usada en la robótica [37], [41], [42], [43], [46], [51], [52], [53]. En este contexto, el primer requerimiento básico para el uso de *landmarks* es tener la capacidad de percibirlas, especialmente durante la ejecución del movimiento del robot, a pesar de las limitaciones del campo de vista del sensor. En [54], [7], se estudia la integración de restricciones noholonómicas y de visibilidad usando un robot DDR, el cual debe mantener en vista los *landmarks* estáticos en un entorno con obstáculos. Primero, se determinan las condiciones necesarias y suficientes para la existencia de una trayectoria de tal manera que el sistema tenga la capacidad de mantener la visibilidad hacia un *landmark* dado. Después, este resultado es extendido al problema de planificación de trayectorias garantizando la visibilidad a través de un conjunto de *landmarks*.

En [13] se presenta un método de navegación con capacidades de sensado limitadas, pero con sensado de discontinuidades en distancia. En este trabajo, se propone el *Gap Navigation Tree* (GNT), el cual es una estructura combinatoria que codifica información de discontinuidades en distancia (*gaps*) y la relación entre ellos. El enfoque original del GNT fue diseñado para la exploración y navegación de un robot modelado como un punto. En el trabajo de [17], se propone un modelo probabilístico para la detección de los (gaps) en el GNT. Este mejora la robustez dado que el modelo trata con el ruido en las mediciones sensadas. En [16], se extiende el esquema del GNT a nubes de puntos. Una amplia familia de sensores de discontinuidades es descrita en [55]. En [15], se presenta un enfoque de seguimiento de pared para la exploración de entornos simplemente conectados usando un robot modelado como un punto. En este trabajo se propone una estructura de datos llamada *cut ordering*. Una vez que se construye la representación del *cut ordering*, esta es utilizada para el problema de persecución/evasión.

Las propuestas de [40], [38], [52], [56], pertenecen a un grupo de métodos de control óptimo ejecutados en lazo abierto. Estos métodos pueden beneficiarse de los resultados de la investigación sobre control visual en robots móviles con ruedas [57], [58], [59], de tal manera que los planes puedan ejecutarse en lazo cerrado.

En el trabajo [60] se presenta un enfoque de navegación óptima respecto a la distancia euclidiana, el enfoque del GNT presentado en [13] se extiende a un robot de manejo diferencial con forma de disco posicionado dentro de una región poligonal y simplemente conectada. El principal resultado es una estrategia de movimiento que dirige al robot navegando de manera óptima hacia un *landmark* en la región. Sin embargo, en [60] no se desarrolla una estrategia de exploración para aprender el GNT y codificar los *landmarks* dentro de este. Las capacidades de sensado y los controles del robot son similares a las presentadas en [13].

2.2. Exploración

La mayoría del trabajo previo para la exploración y construcción de mapas se ha enfocado en 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.

Para una mayor eficiencia en la exploración y construcción de mapas, es posible utilizar la planificación de movimientos para determinar configuraciones convenientes de una futura observación del robot en un 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.

Algunos trabajos han abarcado el problema de exploración de ambientes desconocidos para obtener una representación del entorno [61, 62, 63, 64]. Es posible clasificar estas estrategias de exploración en dos tipos: (i) exploración sistemática y (ii) estrategias en las cuales la información de sensado se toma en cuenta para definir la siguiente ubicación de sensado. En la exploración sistemática (tipo (i)), el robot sigue un patrón de movimiento predefinido, por ejemplo, el seguimiento de pared, moviéndose en círculos concéntricos [65], etc. Algunas estrategias de exploración del tipo (ii) utilizan la exploración basada en la frontera, propuesta por [9]. En el esquema de exploración basada en la frontera, el robot se dirige a la línea imaginaria que divide las partes conocidas y las desconocidas del entorno.

En [61, 62, 63], se proponen estrategias de exploración que guían al robot a las ubicaciones en las cuales se espera la máxima ganancia de información; una función de utilidad es definida para maximizar la nueva información que será obtenida en la siguiente ubicación de sensado. Muchos trabajos han propuesto generar ubicaciones de sensado aleatorias para la exploración (ej. [10, 66]). El trabajo [66], presenta técnicas de exploración basadas en sensado. Dadas una buena capacidad de sensado y odometría, el enfoque estándar de SLAM ([67, 68, 69]) proporciona un mapa geométrico del entorno. En [70], se propone un método para la construcción de un mapa geométrico global sin precisar la ubicación del robot mediante la recolección de escaneos registrados por un telémetro láser. Un enfoque diferente para la construcción de mapas es la rejilla de ocupación [19], la cual representa el ambiente como un arreglo 2D, en lugar de utilizar primitivas geométricas (ej., segmentos de líneas). Otro tipo de representación ambientes son los mapas topológicos en forma de grafos [18, 13, 71]. El problema de exploración de ambientes desconocidos para la búsqueda de uno o más objetivos reconocibles es tratado en [18]. Este método supone capacidades de sensado limitadas en el robot y el entorno es representado en el llamado grafo de lugar de frontera, el cual registra un conjunto de *landmarks*.

2.2.1. Enfoque SLAM

Existen modelos de exploración de entorno capaces de localizar al robot respecto a un marco de referencia global y también pueden estimar la pose del robot. Estos trabajos definen el problema de localización y mapeo simultaneo *SLAM* [72], [73], [74], [75].

Los trabajos en SLAM se enfocan en cuestiones relacionadas con la incertidumbre en el sensado. En [75] usan un enfoque de filtrado de Kalman para manejar incertidumbre acumulada durante el movimiento del robot, proporcionando simultáneamente una estimación de la posición del robot y la localización de los *landmarks*. En trabajos más recientes se han propuesto enfoques Bayesianos. [74] propone flexibilizar a las condiciones restrictivas impuestas en el filtrado de Kalman mediante el enfoque Bayesiano para el problema de SLAM. El enfoque estándar de SLAM se basa en el movimiento en lazo abierto del robot, sin embargo, típicamente no se desarrolla una estrategia de planificación de movimiento en el robot.

2.2.2. Enfoque sin autolocalización

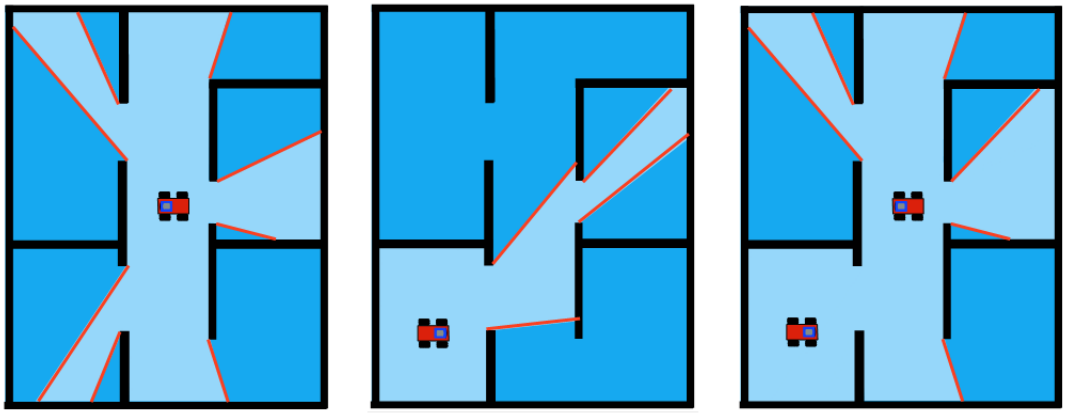
Algunas estrategias de exploración no requieren localizarse así mismos y sus capacidades de sensado son limitadas. [76] propone un método para construir un mapa global geométrico

mediante el registro de mediciones láser de rango sin estimación de la pose. En [77] se direcciona el problema de exploración en un entorno desconocido hacia la búsqueda de uno o más objetivos reconocibles en el entorno. Este método supone que el robot tiene capacidades limitadas de sensado, el entorno se representa en el llamado grafo del lugar límite, el cual registra un conjunto de *landmarks*.

En este enfoque se incluye el trabajo en [13], donde se propone el GNT para representar al entorno para un robot modelado como un punto. Este enfoque se ha extendido en diversos trabajos tales como [78], donde se presenta un modelo probabilístico para los *gaps* en el GNT, los objetivos y *gaps* son cazados probalilísticamente, esto es más robusto ya que el modelo trata con la presencia de ruido en las mediciones del sensor. En [79] este modelo es extendido a modelos de nube de puntos.

Un enfoque de seguimiento de pared para la exploración de entornos simplemente conectados mediante un robot modelado como un punto ha sido presentado en [15]. Este trabajo propone una estructura de datos llamada *cut ordering*. En esta propuesta el robot tiene la capacidad de identificar si en su posición actual está tocando una pared, una esquina convexa, una esquina cóncava o si yace en el interior del entorno. Una vez que se construye la estructura de datos llamada *cut ordering*, es utilizado para abordar problemas de persecución/evasión.

Estrategias de exploración más complejas tratan de determinar la mejor posición de observación para mejorar la eficiencia durante el proceso de exploración, utilizan planificación de movimientos para decidir la siguiente posición de observación. En general, todas las estrategias que tratan de determinar una ubicación 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 2.1). En [80], [81], se describen y comparan algunas estrategias de exploración recientes.



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

Figura 2.1: (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.

2.2.3. Enfoque Glotón

Este enfoque evalúa la posición de observación candidata p en base a su ganancia inmediata, donde la ganancia es 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.

2.2.4. Enfoque de González-Baños

[82], [83], presentan una estrategia de planificación de movimiento para construcción de mapas. [83] presenta una estrategia basada en el cálculo de la siguiente mejor vista (NBV, por sus siglas en inglés) y el uso de planificación de movimientos aleatoria. Este trabajo muestra que es posible encontrar una función que refleja intuitivamente como el robot debe explorar el espacio. 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. En [83] se toma en cuenta el costo asociado al movimiento hacia la siguiente localidad de sensado y 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.

2.2.5. Enfoque de Makarenko

El enfoque presentado en [63] propone una estrategia de exploración para construcción de mapas y localización. El método de la frontera [9] 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 (U_i^I). 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.

La utilidad de hacer una observación desde la localidad destino x_i es definida como la entropía promedio de la región sensada W_i alrededor de la localidad destino. Mientras mayor sea la entropía, menor información sobre la región está disponible y es más atractiva para su exploración.

Utilidad de navegación (U_i^N). 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.

Utilidad de la habilidad para localizarse (U_i^L). 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 **utilidad total (U^{TOT})** de las localidades destino es simplemente la suma pesada de todas las utilidades individuales. La localidad con la utilidad mayor es seleccionada como la siguiente localidad a visitar.

2.2.6. Enfoque de Amigoni

En [11], 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

$$f(p) = \frac{1}{N+A} \sum_{i \in \mathcal{A} \cup \mathcal{N}} \ln(\frac{\sigma_{unc,i}}{\sigma}) + N \ln(\frac{\sigma}{p}) + \sum_{i \in \mathcal{A}} N \ln(\frac{\sigma}{\sigma_{p,i}}) + N \ln(\frac{2\pi \times c}{\sigma}) \qquad (2.2)$$

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 2.2 es la contribución de la entropía de los puntos sensados desde p, el segundo término de la Ecuación 2.2 es la contribución de la entropía de los puntos sensados por primera vez desde p, el tercer término de la Ecuación 2.2 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.

2.2.7. Enfoque de Tovar

En [81], [84], [85], 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 [81] tiene una forma multiplicativa para descartar configuraciones que tengan un valor muy pequeño en alguna de las medidas. Formas similares han sido presentadas en [85], [84]. 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.

2.2.8. Enfoque de Newman

En [86], 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 [86] 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.

2.2.9. Enfoque de Feder

[62] presenta otra estrategia de movimiento para la exploración con el robot. En este trabajo, se propone una métrica de sensado definida en términos de información de Fisher. Esta métrica elige entre acciones discretas del robot dado el estado actual, puede localmente maximizar la información obtenida en la siguiente lectura del sensor. Al aplicar este algoritmo el robot tiende a explorar selectivamente diferentes objetos en el entorno, pero este enfoque no considera ni planeación de trayectoria ni evasión de obstáculos.

2.2.10. Enfoque de Calisi

En [87] se enfoca al problema de exploración y búsqueda simultáneas. La acción de cazar un objetivo puede ser interrumpida si un mejor objetivo puede ser detectado durante la navegación. Este enfoque es flexible, es capaz de evaluar las posibles acciones de movimiento y elige entre diferentes objetivos. La base de las estrategias de exploración y búsqueda está inspirada en la estructura del algoritmo de la siguiente mejor observación (NVB), este se divide en 2 pasos i) detección, evaluación, y selección de objetivos, y ii) navegación al objetivo elegido.

2.2.11. Enfoque de Laguna

En [1] se desarrolló una estrategia de exploración para un robot de manejo diferencial con forma de disco que es representada como una Máquina 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 [13] es usada para representar el entorno y codificar una *landmark* que está ubicada en el entorno.

2.3. Control para seguimiento de pared

Existen diferentes esquemas de control que han sido propuestos para lograr un comportamiento de seguimiento de pared, mediante el uso de retroalimentación basada en información de un sensor láser de rango [88], [89], [90]. El trabajo presentado en [88] se basa en la detección y representación robusta de paredes. En el trabajo [89] se presenta un esquema de control basado en la conmutación de controladores para el seguimiento de pared basado en información de odometría y distancia. El esquema de control está diseñado para evitar saturación en la velocidad angular del robot cuando lidia con contornos discontinuos. Este esquema de control ha sido usado para evasión reactiva de obstáculos en el seguimiento del contorno de obstáculos en [90]. El uso de un sensor de distancia ha sido extendido a un sensado activo en [91], donde la percepción y sistemas de acción del robot están dinámicamente acoplados por un seguimiento de pared reactivo. Hay otros tipos de sensores que han sido utilizados para la tarea de seguimiento de pared, por ejemplo, una antena bioinspirada [92], la cual es un sensor táctil pasivo. En [93], los autores proponen un método de navegación de seguimiento de pared implementado en un robot LEGO Minds-

torms NXT, el cual cuenta con dos sensores de contacto conocidos como *bumpers*. En otros esquemas más recientes como [94] y [95] utilizan controladores basados en lógica difusa, dichos controladores se retroalimentan con la información proporcionada por un arreglo de sensores ultrasónicos para medir distancia a la pared.

2.4. Principales contribuciones de la tesis

El problema de encontrar trayectorias libres de colisiones y el problema de exploración de entornos desconocidos con un robot móvil, son dos temas muy activos en la comunidad de robótica. Pero de la literatura antes mencionada, los trabajos que están más estrechamente relacionada con nuestro enfoque son los presentados en [13, 15, 60] y [96]. Una diferencia significativa con respecto al trabajo de [15, 13], es que el robot es un punto, nosotros modelamos un robot de manejo diferencial (noholonómico) como un disco. Además, la estrategia de exploración propuesta en este documento, está basada en el seguimiento de pared y no en seguir los *gaps* en contraste con [13]. En [60, 96] se considera un robot disco y se presenta una estrategia de navegación para alcanzar un *landmark*, pero no se incluye una estrategia de exploración para codificar un *landmark* en el GNT, en esta tesis presentamos dicha estrategia de exploración.

La principales contribuciones del presente documento son una política de movimiento basado en la simple retroalimentación con un sensor y una estrategia completa de exploración. Se presenta las condiciones teóricas que garantizan que el robot descubrirá la mayor región posible del ambiente. La estrategia propuesta es compacta, de tal manera que esta se representa como una máquina de estados finitos de Moore. Además la estrategia propuesta no requiere la localización del robot.

Para explorar el ambiente, el robot sigue la frontera del entorno. Se propone un esquema de control híbrido, cuyo objetivo es mantener una distancia deseada entre el robot y la frontera del entorno. Este esquema permite al robot lidiar con imperfecciones en los controles para tratar con la dinámica del robot (ej. variaciones de velocidades). Además, este esquema de control tiene por objetivo mantener la continuidad de las velocidades angular y lineal del robot a pesar de la conmutación entre los controladores. La principal originalidad del enfoque propuesto con respecto al trabajo previo es que el sensor proporciona observaciones relacionadas directamente a los controles, y en este enfoque, un autómata restringe los posibles estados de transición filtrando falsas observaciones debido al ruido en las lecturas del sensor.

En [1] se propuso una máquina estados finitos para explorar un ambiente desconocido por medio de seguimiento de la frontera de los obstaculos. Las principales diferencias entre este trabajo y el presentado en [1] son: 1) Se propone un autómata que filtra las observaciones espurias para activar controladores basados en retroalimentación. 2) Se propone un esquema de control híbrido, cuyo objetivo es mantener una distancia deseada entre el robot y la frontera del ambiente. 3) El trabajo presentado en [1] no incluyó ningún experimento en el robot real. En este nuevo trabajo, se presentan experimentos sobre un robot real mostrando la vialidad y practicidad de nuestro esquema.

Se propone un control de modos deslizantes que converge en tiempo finito a la consigna de control, lo que permite al robot navegar de manera continua sin detenerse cuando se presentan obstáculos en su camino. Además en esta nueva estrategia de control solo las referencias de control cambian con el estado del autómata y el controlador no varia.

2.5. Organización de la tesis

En esta sección se muestra la organización del resto de este documento de tesis:

El Capítulo 3 se presenta los preliminares, esto es, los conceptos correspondientes al GNT [13]. En el Capítulo 4 se presenta la estrategia de movimiento modelada como una máquina de Moore y una política de movimiento basada en retroalimentación que mapea las observaciones a los controles del robot. En el Capítulo 5 se determinan diversas mediciones en el sensado con la finalidad de usarlas como información de retroalimentación para los controladores y además, para transformarlas en un conjunto de observaciones binarias que efectuarán la transición entre los estados del autómata. El Capítulo 6 presenta una nueva máquina de Moore cuyo objetivo es mover al robot a una distancia deseada desde la frontera del entorno. En este caso, cada una de las observaciones activan un control específico, conformando así un esquema de control híbrido basado en retroalimentación. La implementación del método presentado en esta tesis es detallada en el Capítulo 7 donde se muestran los resultados de las simulaciones y los experimentos realizados en el robot real. También se presenta el desempeño de los controladores diseñados para navegación tanto dentro de un laboratorio como en una oficina del CIMAT. El Capítulo 8 muestra una estrategia alternativa de navegación para seguir la frontera del ambiente, donde un solo controlador es usado para controlar la velocidad lineal y angular, en el cual solo los valores de referencia de control cambian con el estado del autómata. También se sugiere un controlador de modos deslizantes que tiene la propiedad de alcanzar la consigna de control en tiempo finito [97], [98] y [99]. Esta nueva propuesta de control permite no detener al robot

cuando encuentra un obstáculo. Finalmente, en el Capítulo 9 se presentan las conclusiones y el trabajo futuro de esta tesis.

A continuación, en el Capítulo 3, se presenta de manera detallada el *Gap Navigation Tree*, el cual es un mapa topológico que se utiliza para representar el ambiente y determinar cuando es finalizada la tarea de exploración para el esquema propuesto en este trabajo.

Capítulo 3

Preliminares

Durante la tarea de exploración que se ejecuta en nuestro esquema, es necesaria una herramienta que nos permita obtener una representación funcional del entorno y que sea capaz de determinar que la exploración ha concluido. Por ello se utiliza un mapa topológico llamado *Gap Navigation Tree* (GNT) [13, 60]; esta estructura combinatoria codifica la información de discontinuidades en distancia (*gaps*) y la relación que existe entre ellos. A continuación se presenta el funcionamiento y construcción del GNT.

3.1. Gap Navigation Tree

El *Gap Navigation Tree* es utilizado en [1] para obtener una representación del entorno. El GNT es una eficiente estructura de datos que cambia dinámicamente de acuerdo a algunos eventos críticos durante la exploración hasta que el entorno ha sido del todo descubierto.

3.1.1. Construcción del GNT y eventos críticos

Mientras el robot se mueve en el entorno en una trayectoria τ, el GNT puede ser construido incrementalmente. Inicialmente el GNT consiste en un nodo raíz el cual está conectado a un nodo hoja por cada *gap* en $G(\tau(0))$. Cada tiempo t para el cual $G(\tau(t))$ cambia, ocurre un evento crítico, por lo cual se requiere la actualización del GNT. En total hay 4 diferentes eventos críticos:

1. Un nuevo *gap* g aparece: Un nuevo nodo g es añadido como un hijo de la raíz, mientras se preserve el orden cíclico angular del sensor de *gaps* (Ver Figura 3.1(a)).

2. Un *gap* g desaparece: El nodo g el cual debe ser un nodo hoja es removido (Ver Figura 3.1(b)).

3. Los *gaps* g_1 y g_2 se fusionan dentro del *gap* g: Los nodos g_1 y g_2 se vuelven nodos hijo del nuevo nodo g, el cual es añadido como un hijo del nodo raíz y preservando el orden cíclico de los *gaps* (Ver Figura 3.1(c)).

4. El *gap* g_1 se divide en g_2 y g_3: Si g_1 es un nodo hoja, entonces g_2 y g_3 se convierten en nuevos nodos; de otra manera estos ya existían como hijos del nodo g_1. Tanto g_2 como g_3 son conectados al nodo raíz, preservando el orden de los *gaps* y removiendo g_1 (Ver Figura 3.1(d)).

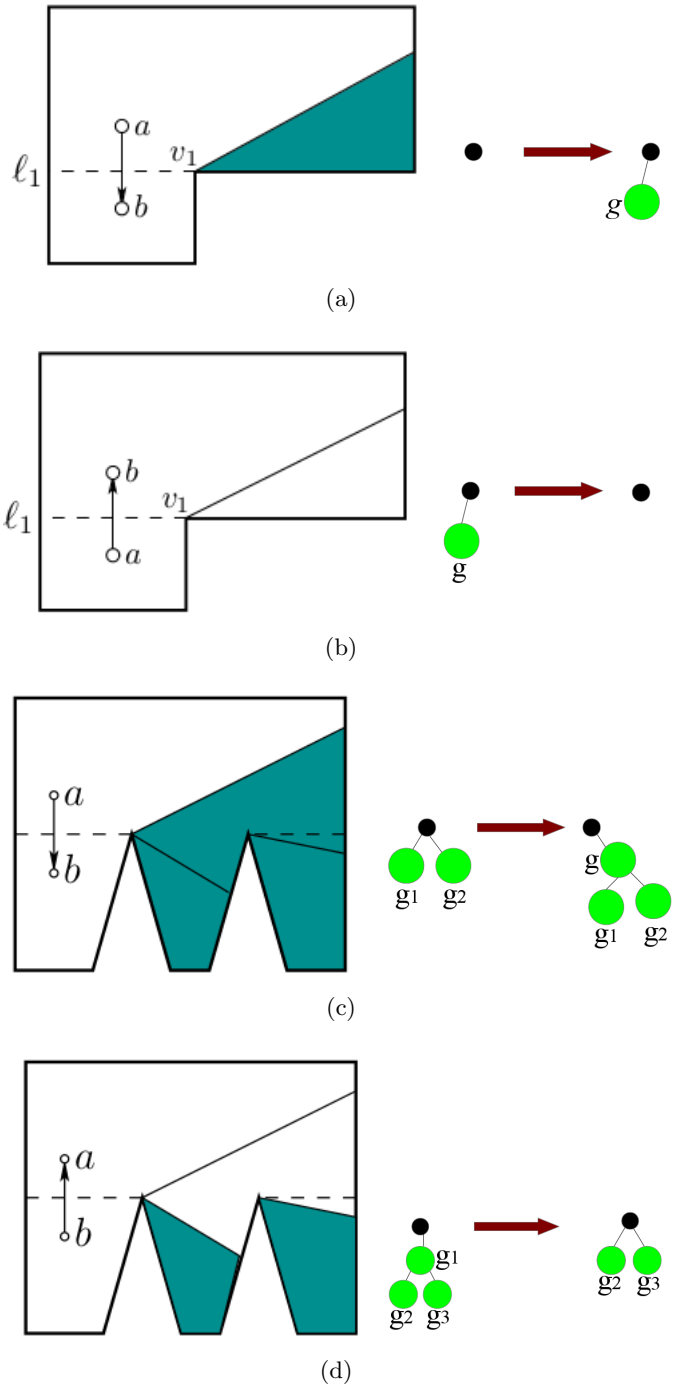


Figura 3.1: Eventos críticos: (a) Gap aparece, (b) Gap desaparece, (c) Gaps se fusionan, (d) Gap se divide.

3.1.2. Construcción del GNT completo

Los nodos del GNT pueden ser primitivos o no primitivos. El nodo primitivo codifica el espacio ocluido que ya ha sido percibido, mientras el nodo no primitivo codifica el que aún no ha sido percibido. Al inicio todos los nodos son no primitivos, y de acuerdo a la aparición de *gaps* los nodos primitivos se van añadiendo. Si algún vértice hoja puede dividirse, entonces el GNT está incompleto ya que puede expandirse, algunos *gaps* se dividen y otros simplemente desaparecen. Los *gaps* que aparecen son llamados primitivos (sus correspondientes nodos en el GNT son también llamados primitivos).

Si todos los nodos hoja del GNT se vuelven primitivos, entonces se dice que el GNT está completo por lo tanto es necesario cazar (perseguir) iterativamente todos los nodos hoja no primitivos para completar en su totalidad el GNT. Cada que un nodo hoja desaparece, otro nodo hoja es elegido para ser cazado. No es relevante el orden en que los nodos hoja son cazados. Mediante este proceso todos los nodos hoja no primitivos serán cazados hasta volverse primitivos, en cuyo caso el GNT será completado. El siguiente Lema en [13] garantiza la terminación de la construcción del GNT.

Lema 1. *([13]) El procedimiento de seguir hojas no primitivas iterativamente termina con un GNT completo como resultado.*

En [13], perseguir un *gap* significa mover el sensor del robot hasta que este toque el vértice que genera el gap, observando así la porción del ambiente ocluida por dicho vértice y por lo tanto, haciendo que el correspondiente *gap* desaparezca. La observación clave para demostrar el Lema 1 es que, *en cualquier momento que un gap nuevo aparezca en el GNT, este debe ser primitivo*. Por lo tanto, los únicos *gaps* que contribuyen a la incompletitud del GNT, son los que aparecen en $G(\tau(0))$ (al comienzo de la exploración) o aquellos que fueron formados mediante una sequencia de divisiones de estos gaps.

En [13], los ángulos de los *gaps* son desconocidos debido a las capacidades limitadas del sensor, pero el sensor es capaz de mantener un orden cíclico angular de éstas. Sea $G(x) = [g_1, ..., g_k]$ que denota la secuencia de *gaps* tal como aparecen en el sensor, cuando este es colocado en $x \in E$, si x yace en el interior de E, hay un orden cíclico tal que pueden hacerse las sentencias como $[g_1, ..., g_k] = [g_2, ..., g_k, g_1]$ (ver Figuras 3.2 a) y b)). Si x yace en ∂E, entonces parte de la visibilidad del sensor está obstruida por la frontera, y se obtiene un orden lineal de los *gaps* (ver Figuras 3.2 c) y d)).

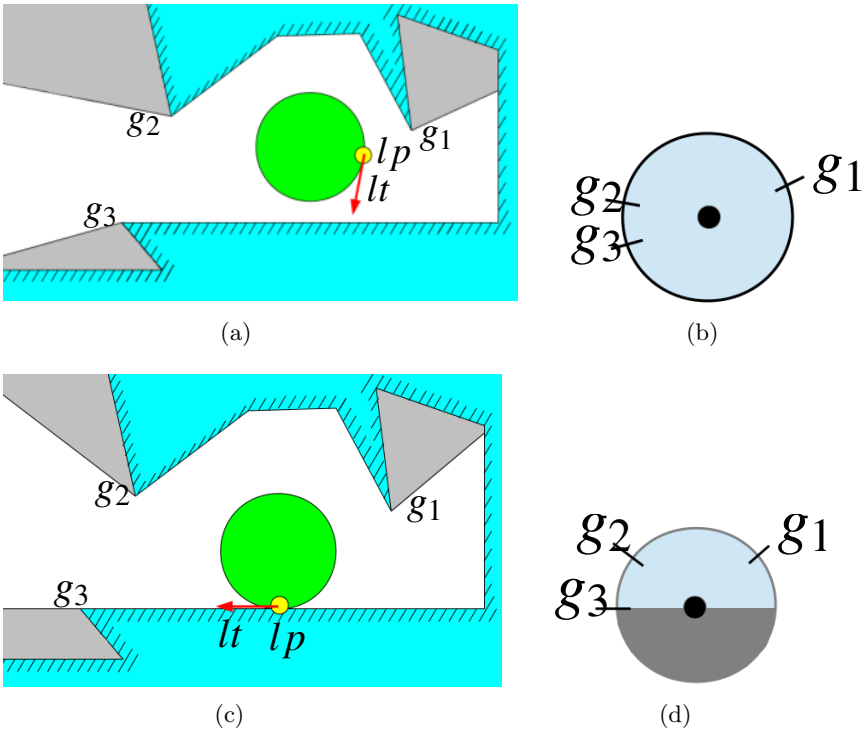


Figura 3.2: a) Sensor de *gaps* en $x \in E$, b) Orden cíclico, c) Sensor de *gaps* en $x \in \partial E$, d) Orden lineal.

Tras presentar el funcionamiento y la construcción del GNT, en el Capítulo 4 se presenta una estrategia de exploración mediante una Máquina de Estados Finitos de Moore para el caso ideal donde el robot navega en contacto con la pared utilizando controles muy precisos.

Capítulo 4

Autómata para Caso Ideal

En [1], se presenta el esquema de navegación y exploración en el caso ideal donde el sensado y los controles son perfectos. Esta referencia es el antecedente principal del trabajo de tesis desarrollado. A continuación se describe las principales características de [1].

4.1. Modelo de sensado

El robot de manejo diferencial tiene su *heading* en la parte frontal. Los puntos en los extremos izquierdo y derecho del robot son llamados lp y rp respectivamente. El robot tiene un sensor omnidireccional, el cual es utilizado para descubrir el entorno E. La dirección de la línea tangente a la frontera del robot sobre rp es llamada rt. La dirección de la línea tangente a la frontera del robot sobre lp es llamada lt (ver Figura 4.1). El sensor omnidireccional, puede ser posicionado en los puntos rp y lp y puede medir sobre la dirección rt o lt dependiendo donde se encuentre ubicado.

El sensor omnidireccional tiene la capacidad de detectar y seguir discontinuidades (*gaps*) en información de profundidad, además tiene la capacidad de detectar cualquiera de los cuatro eventos críticos: aparición de *gap*, desaparición de *gap*, fusión de *gaps* y división de *gaps*. Por estas capacidades es posible construir un detector de *gaps* y referirse a este sensor como un *sensor de gaps*.

El sensor de *gaps* es capaz de detectar y ordenar las direcciones de los *gaps*, detectar la dirección rt o lt y verificar si hay obstrucción en la visibilidad cuando el sensor está en contacto con la pared ∂E. Gracias a esto, es posible detectar eventos como el alineamiento entre las direcciones rt o lt y cualquier *gap*, o el alineamiento entre uno de las dos direcciones lt o rt y la pared (∂E) que está en contacto con el sensor omnidireccional.

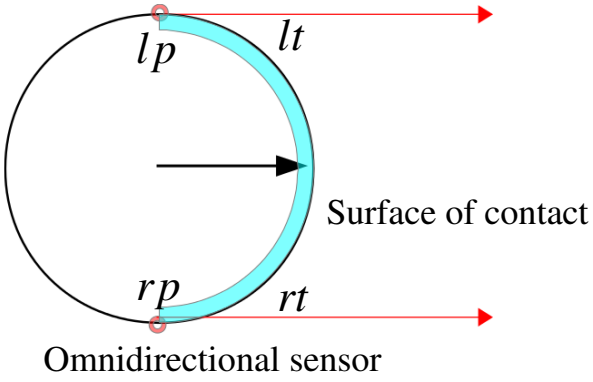


Figura 4.1: Modelo de Sensado del robot [1]. Se presenta el caso donde el sensor omnidireccional está posicionado en el punto rt.

Sea Λ un landmark estático con forma de disco con un radio igual al del robot y que yace sobre el interior de E. Se dice que un *landmark* Λ puede ser identificado si es por lo menos parcialmente visible desde la ubicación del sensor omnidireccional.

Se supone que el *landmark* Λ está pintado en el piso, que puede ser detectado y no tiene volumen, por lo cual no produce *gaps*. Esta suposición se hace para implementar posteriormente una tarea de navegación, en la cual una vez que el entorno ha sido explorado, el robot tendrá como objetivo estacionarse sobre el *landmark*.

La suposición acerca de que el *landmark* es un disco con el mismo radio que el robot, permite establecer que si el subconjunto libre de colisiones del espacio de configuraciones $\mathcal{C}$ del robot es simplemente conectado, entonces el *landmark* será alcanzable. Así mismo esto permite establecer que, en este caso, el *landmark* completo será totalmente visible en algún momento durante la exploración.

Respecto a la codificación del *landmark*, si el *landmark* es completamente visible desde la posición actual, este es conectado directamente al nodo raíz del GNT.

Se supone que el robot puede distinguir si existe contacto con uno o más puntos. Esto es, el robot es capaz de detectar si su periferia frontal está en contacto con un obstáculo. La periferia frontal es llamada superficie de contacto (ver Figura 4.1). El sensor también distingue si el punto rp o lp está en contacto con la pared. Esta información puede ser obtenida con varios diferentes sensores por ejemplo un sensor táctil o un sensor láser omnidireccional. El caso particular donde ambos puntos rp y lp entran al mismo tiempo en contacto con una pared no está considerado, esto solo sucede en un corredor estrecho con

la misma anchura del robot.

4.2. Vector de observaciones

Con las capacidades del sensor que se definieron anteriormente es posible definir un vector de observaciones que contenga toda la información necesaria para accionar un control específico de acuerdo a una estrategia de movimiento.

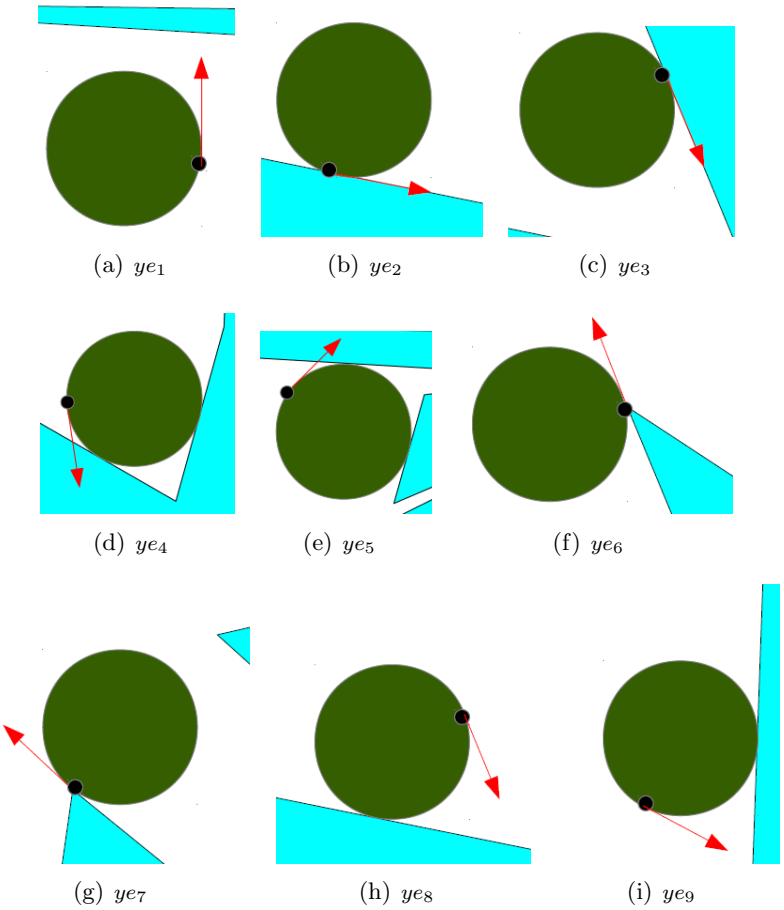
El vector de observaciones se compone de seis elementos binarios determinados por las siguientes situaciones: (1: lp) el robot está tocando ∂E en el punto lp. (2: rp) el robot está tocando ∂E en el punto rp. (3: sc) el robot está tocando ∂E en un solo punto dentro de la superficie sensitiva. (4: bc) el robot está tocando ∂E en dos o más puntos dentro de la superficie sensitiva (uno de estos puntos puede ser lp o rp). (5: $aligned$) cuando rp está en contacto con ∂E: si la dirección rt está alineada al borde poligonal que rp está tocando, o si el punto rp está tocando un vértice reflejo y la dirección de rt está alineada al primer borde poligonal medido en sentido de las manecillas del reloj iniciando desde la dirección rt, entonces $aligned = 1$. Para el caso en que lp está en contacto con ∂E: si la dirección lt está alineada al borde poligonal que lp está tocando, o, si el punto lp está tocando un vértice reflejo y la dirección de lt está alineada al primer borde poligonal medido en sentido contrario de las manecillas del reloj iniciando desde la dirección lt, entonces $aligned = 1$. (6: o) si el sensor omnidireccional se encuentra ubicado en el punto lp (0) o si el sensor omnidireccional está ubicado en el punto rp (1). De esta manera el vector de observaciones que se tiene es $ye_i = \{lp, rp, sc, bc, aligned, o\}$.

El conjunto de todas las 64 observaciones posibles puede ser particionado mediante el uso de una variable binaria x que puede tomar cualquiera de los dos valores binarios:

$$
\begin{aligned}
ye_1 &= (0, 0, 0, 0, x, x) \\
ye_2 &= (0, 1, 1, 0, 1, 1) \\
ye_3 &= (1, 0, 1, 0, 1, 0) \\
ye_4 &= (x, x, 0, 1, x, 1) \\
ye_5 &= (x, x, 0, 1, x, 0) \\
ye_6 &= (0, 1, 1, 0, 0, 1) \\
ye_7 &= (1, 0, 1, 0, 0, 0) \\
ye_8 &= (0, 0, 1, 0, x, 0) \\
ye_9 &= (0, 0, 1, 0, x, 1)
\end{aligned}
$$

A continuación se presenta el significado de cada uno de estos vectores de observaciones:

- ye_1 *No hay contacto*: Esta observación solamente pasa al comienzo de la exploración si el robot yace completamente en el interior de E, de tal manera que no se detecta contacto alguno (Ver Figura 4.2(a)).
- ye_2 *Contacto simple en rp*: El sensor omnidireccional está ubicado en rp, hay un contacto simple detectado en este punto, y la dirección preferencial rt está alineada con el borde poligonal que el punto rp está tocando (Ver Figura 4.2(b)).
- ye_3 *Contacto simple en lp*: Esta observación es análoga a la observación *Contacto simple en rp*, se trata del caso simétrico izquierdo (Ver Figura 4.2(c)).
- ye_4 *Multicontacto, sensor en rp*: El sensor omnidireccional está ubicado en el punto rp y hay un multicontacto detectado (rp puede ser un punto de contacto), mientras el sensor omnidireccional es ubicado en rp. La superficie sensible del robot está tocando más de un punto de ∂E, el contacto puede ser una combinación de bordes o vértices reflejo de E (Ver Figura 4.2(d)).
- ye_5 *Multicontacto, sensor en lp*: Esta observación es el análogo del *Multicontacto, sensor en rp*, este es el caso simétrico izquierdo (Ver Figura 4.2(e)).
- ye_6 *Vértice reflejo rp*: El sensor omnidireccional está ubicado en el punto rp, hay contacto simple entre el punto rp y el vértice reflejo del entorno poligonal, la dirección preferencial rt no está alineada al primer borde poligonal medido en sentido de las manecillas del reloj iniciando desde la dirección de rt (Ver Figura 4.2(f)).
- ye_7 *Vértice reflejo lp*: El sensor omnidireccional está ubicado en el punto lp, hay contacto simple entre el punto lp y el vértice reflejo del entorno poligonal, la dirección preferencial lt no está alineada al primer borde poligonal medido en sentido contrario de las manecillas del reloj iniciando desde la dirección de lt (Ver Figura 4.2(g)).
- ye_8 *No hay contacto simple en lp*: El sensor omnidireccional está ubicado sobre el punto lp y el robot está tocando un borde o un vértice reflejo de ∂E en un único punto diferente a lp (Ver Figura 4.2(h)).
- ye_9 *No hay contacto simple en rp*: Esta observación es el caso análogo de *No hay contacto simple en lp*, es el caso simétrico izquierdo con el sensor omnidireccional ubicado en el punto rp (Ver Figura 4.2(i)).



(a) ye_1 (b) ye_2 (c) ye_3

(d) ye_4 (e) ye_5 (f) ye_6

(g) ye_7 (h) ye_8 (i) ye_9

Figura 4.2: Vector de observaciones

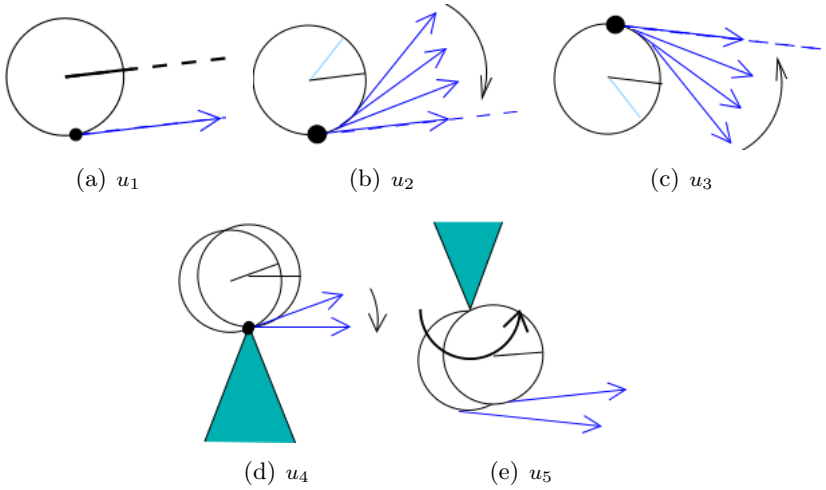
4.3. Modelo de movimiento

El robot de manejo diferencial tiene dos ruedas independientes, cada rueda tiene su propio motor que la acciona.

Sean ω_l y ω_r, las velocidades angulares de las ruedas izquierda y derecha respectivamente, donde ω_l, $\omega_r \in \{-1, 0, 1\}$. Los controles del robot son definidos por el vector $u = \{\omega_l, \omega_r\}$. Esto es equivalente a controlar el sistema usando las velocidades lineal y angular.

Hay cinco primitivas de movimiento generadas por los siguientes controles:

$u_1 = (1, 1)$	Mover hacia adelante en línea recta.
$u_2 = (1, -1)$	Rotación en sitio en sentido de las manecillas del reloj.
$u_3 = (-1, 1)$	Rotación en sitio en sentido contrario de las manecillas del reloj.
$u_4 = (1, 0)$	Rotación en sentido de las manecillas del reloj con respecto al punto rp.
$u_5 = (0, 1)$	Rotación en sentido contrario de las manecillas del reloj con respecto al punto lp.



(a) u_1 (b) u_2 (c) u_3

(d) u_4 (e) u_5

Figura 4.3: Primitivas de movimiento:(a) Mover hacia adelante en línea recta, (b) Rotación en sitio en sentido de las manecillas del reloj, (c) Rotación en sitio en sentido contrario de las manecillas del reloj, (d) Rotación en sentido de las manecillas del reloj con respecto al punto rp, (e) Rotación en sentido contrario de las manecillas del reloj con respecto al punto lp. (Presentado en [1]).

Mediante el accionamiento de los controles definidos anteriormente, el robot es capaz de explorar el entorno mediante un seguimiento de pared en contacto con la misma. Si el sensor omnidireccional está ubicado en el punto rp, entonces el robot sigue la frontera del entorno ∂E en sentido contrario a las manecillas del reloj, y si el sensor está ubicado en el punto lp, entonces el robot sigue la frontera del entorno ∂E en el sentido de las manecillas del reloj.

4.4. Autómata para la exploración

Una máquina de estados finitos (FSM, por sus siglas en inglés) es definida como un modelo matemático de computación, la cual es concebida como una máquina abstracta que puede estar en uno de un número finito de estados [100]. La máquina solo puede estar en uno de esos estados en un determinado momento y puede cambiar de un estado a otro al acontecer un evento o condición conocido como *transición*. Un tipo especial de FSM es conocida como la máquina de Moore, la cual tiene salidas asociadas a cada estado. En [1] se propone representar la estrategia de exploración propuesta mediante el uso de una Máquina de Moore.

La estrategia de exploración del robot es representada mediante una FSM M, donde M incluye la política de movimiento así como ciertas consultas y actualizaciones en el GNT. La política de movimiento es el mapeo directo entre observaciones y controles (ver Sección 4.5). Nótese que la política de movimiento es solo una parte de la estrategia de exploración completa.

La condición de paro que finaliza la tarea de exploración requiere información topológica que no está dada por la lectura actual de los sensores, la información es obtenida del GNT en el transcurso de movimiento del robot, y por ello no es incluida en las políticas de movimiento. En [13], se trabajó en la tarea de exploración con el GNT para un robot modelado como un punto, la tarea de exploración finaliza cuando *todos los nodos hoja del GNT han sido etiquetados como nodos primitivos*, lo cual indica que el entorno ha sido explorado por completo. Para el robot con forma de disco tratado en [1], es similar, pero se incluye las dificultades de que algunos *gaps* pueden nunca desaparecer debido a las dimensiones del robot que generan regiones inalcanzables en el entorno. Para lidiar con este problema, en [1] se presenta un algoritmo llamado *local exploration* (Ver Algoritmo 1)

En la Figura 4.4 se muestra una representación gráfica de M. Hay siete estados, uno de estos es el estado inicial donde no se ejecuta ninguna primitiva de movimiento, hay un estado final el cual establece que el GNT se ha completado, esto significa que la tarea de exploración ha sido finalizada, entonces no se aplica ninguna primitiva de movimiento y el robot se detiene. Hay otros estados que representan la ejecución de las primitivas de movimiento definidas en la Sección 4.5. Todos los enlaces de transición en la Figura 4.4 están etiquetadas con su correspondiente observación, definidas en la Sección 4.1, con la excepción del enlace de transición etiquetado como GNT, este representa una consulta el GNT verificando si todos los nodos hoja están marcados como primitivos.

Las consultas al GNT se hacen en los estados de Rotación en Sitio en sentido contrario a las manecillas del reloj (*CCW* por sus siglas en inglés), Rotación en Sitio en sentido de las manecillas del reloj (*CW* por sus siglas en inglés) y en Línea Recta (Ver Figura 4.4). Esto es porque el GNT puede cambiar dada la ocurrencia de un evento crítico mientras el robot está ejecutando una de estas primitiva de movimiento. Las consultas son requeridas para determinar si las exploración ha terminado o no. El algoritmo *local exploration* se ejecuta en los estados Rotación en Sitio CCW o Rotación en Sitio CW. El algoritmo *Local Exploration* también actualiza las etiquetas de los *gaps* en el GNT.

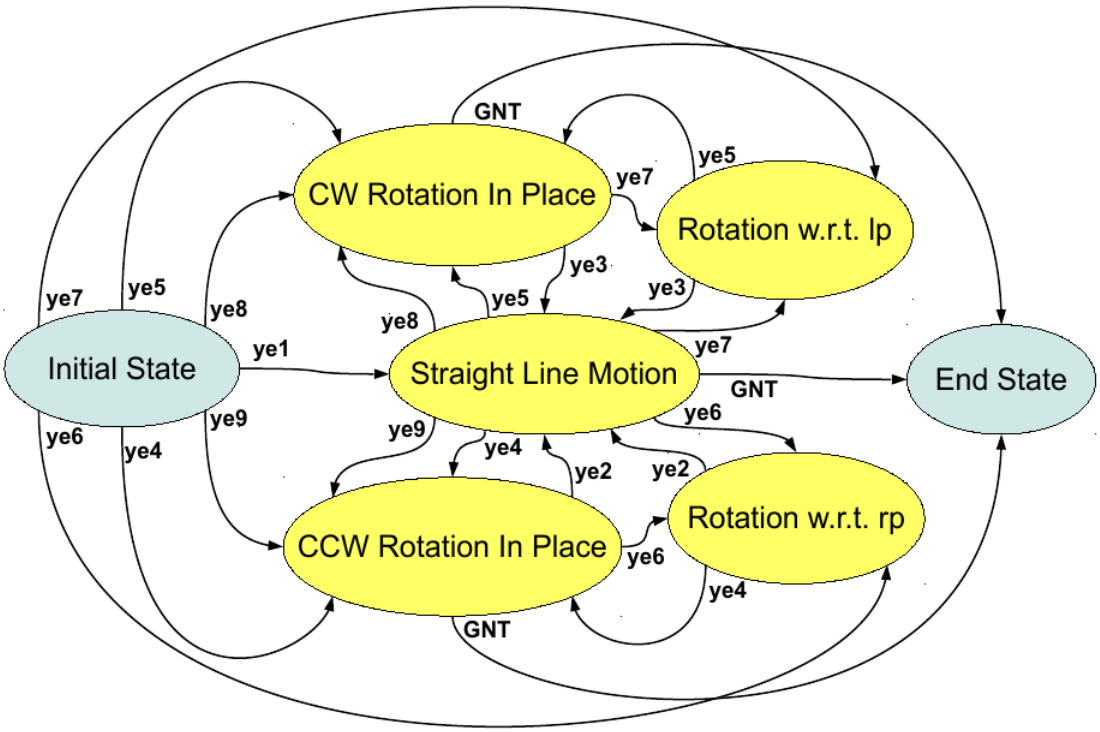


Figura 4.4: Máquina de Estados Finitos que representa la estrategia de exploración de [1].

4.5. Políticas de movimiento

La política de movimiento se basa en el paradigma de evitar la estimación del estado para llevar a cabo dos mapeos consecutivos: $y \rightarrow x \rightarrow u$, esto es, desde la observación y al estado x y después al control u. En [1] se presenta una política de movimiento la cual hace un mapeo directo desde las observaciones a los controles, esto es $y \rightarrow u$.

Sea γ una función de mapeo, la política de movimiento puede establecerse como

$\gamma : \{0,1\}^6 \rightarrow \{-1,0,1\}^2$, entonces la función es expresada como $\gamma\left(ye_i\right) = (\omega_l, \omega_r) = u_j$. La política de movimiento es:

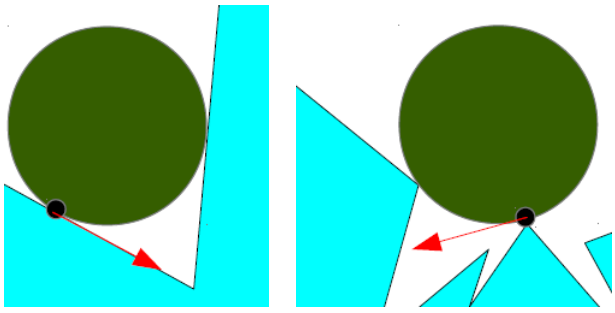
- $\gamma(ye_1 \vee ye_2 \vee ye_3) = u_1$
- $\gamma(ye_5 \vee ye_8) = u_2$
- $\gamma(ye_4 \vee ye_9) = u_3$
- $\gamma(ye_6) = u_4$
- $\gamma(ye_7) = u_5$

Donde $\vee$ significa "or".

La lista previa resume la relación completa entre los controles y las observaciones dadas por el sensor.

4.6. Algoritmo *local exploration*

Las restricciones del espacio de configuraciones de un robot con forma de disco generan regiones inalcanzables en el entorno. En estas regiones los *gaps* pudiesen no desaparecer independientemente del movimiento del robot. Cuando rp y lp yacen en ∂E se identifican las observaciones donde los *gaps* pudiesen no desaparecer, estas observaciones son: $ye_4^R = (0,1,0,1,x,1)$ or $ye_5^L = (1,0,0,1,x,0)$ (Ver Figuras 4.5 (a) and (b)).



(a) ye_4^R (b) ye_5^L

Figura 4.5: Observaciones ye_4^R y ye_5^L (Presentado en [1])

Estos son casos especiales de las observaciones ye_4 y ye_5 respectivamente. El algoritmo *local exploration* se activa cuando cualquiera de estas dos observaciones suceden. Este

algoritmo usa información del GNT y finaliza cuando los *gaps* en los nodos codificados generados por las regiones inalcanzables del entorno son etiquetadas como primitivas. El algoritmo usa la propiedad que establece *el cambio desde el ordenamiento cíclico al orden lineal de los gaps cuando el sensor de gaps está en contacto con el muro de acuerdo al modelo detallado en [13].*

A continuación se presenta el Algoritmo 1 *local exploration*:

Algoritmo 1 Local Exploration

Entrada: GNT, observación actual: ye_i.
Salida: GNT actualizado.
if rp =**true then**
  1. *init-list* ← los *gaps* actuales y la dirección rt inician desde la región de visibilidad obstruida del sensor siguiendo un orden en sentido contrario a las manecillas del reloj;
  if $ye_i = ye_4^R$ (u_3 es ejecutado) **then**
    while $(ye_i \neq ye_2)$ **and** $(ye_i \neq ye_6)$ **do**
      if *evento-GNT* = **true then**
        if *evento-crítico* ≠ *gap-aparece* **then**
          2. Aplicar la actualización sufrida por los nodos hijos de la raíz del GNT a los *gaps* correspondientes en *init-list*;
        end if
      end if
      3. Actualizar la posición de la dirección rt (debida al movimiento del sensor) en *init-list* de acuerdo al actual orden angular en sentido contrario a las manecillas del reloj en las lecturas del sensor;
    end while
    4. *end-list* ← los *gaps* actuales y la dirección rt comienzan desde la región de visibilidad obstruida del sensor siguiendo un orden en sentido contrario de las manecillas del reloj;
    5. $G_1 \leftarrow \{x \in \textit{init-list} \mid init_f \leq x < init_{rt}\}$;
    6. $G_2 \leftarrow \{x \in \textit{end-list} \mid end_{rt} < x \leq end_l\}$;
  end if
end if
8. $G_\cap \leftarrow G_1 \wedge G_2$;
for cada *gap* $g_i \in G_\cap$ **do**
  9. etiquetar nodo g_i en el GNT como nodo primitivo;
  10. Propagar la etiqueta primitiva a la descendencia de g_i;
end for

El Algoritmo 1 considera el caso en el que el punto rp está en contacto con el entorno, para el punto lp en contacto con el entorno es solamente el caso simétrico para el punto rp. Ver [1] para mayor detalle del algoritmo.

La idea principal de este algoritmo es la siguiente. Los *gaps* están ordenados de modo angular en dos listas, una inicia desde la dirección del primer punto de contacto, (antes que el robot comience la rotación en sitio) y la otra inicia desde la dirección del segundo punto de contacto (después que el robot finaliza la rotación en sitio). Adicionalmente, la dirección de la línea rt es usada como una dirección de referencia en común en ambas listas. Con base en estas tres direcciones (dirección del primer punto de contacto, dirección del segundo punto de contacto y la dirección de la línea rt) es posible determinar cuales *gaps* son generados por los vértices no accesibles para el robot.

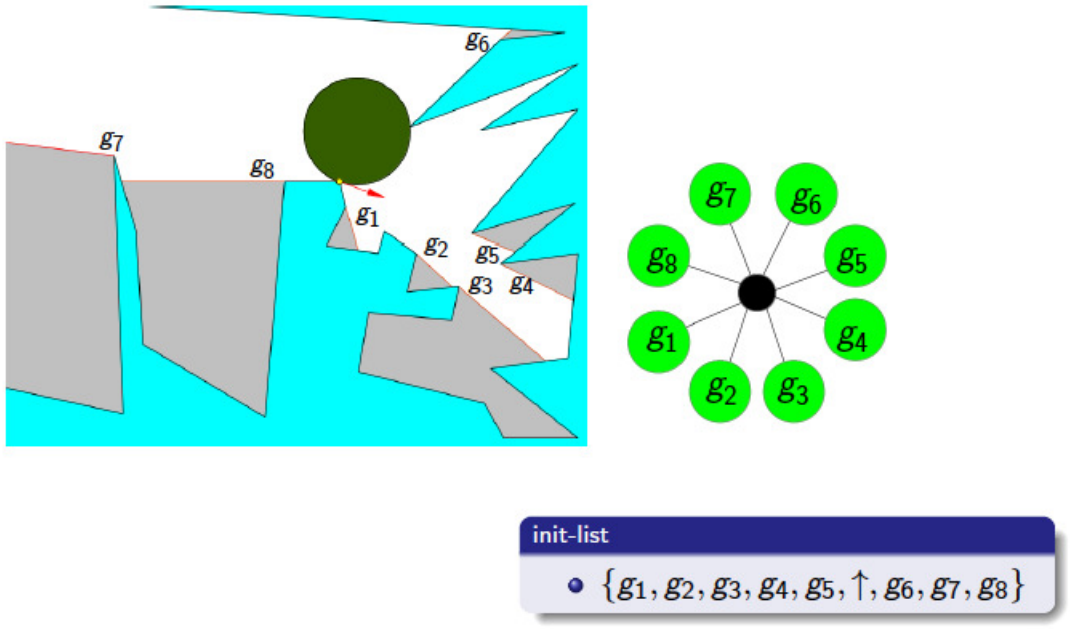


Figura 4.6: Lista *init-list* y *gaps* antes de la rotación en sitio del robot.

Antes que el robot ejecute la rotación en sitio, los *gaps* son ordenados por ángulo iniciando desde la dirección definida por el centro del robot y el punto ubicado en rp. Los *gaps* son almancenados en una lista lineal llamada *init-list* (Ver Figura 4.6). Los cambios (división, fusión, apariciones y desapariciones) en los *gaps*, debidos a la rotación en sitio del robot, son actualizados en la lista *init-list* hasta que el sensor toca el segundo vértice.

Despues de que el robot finaliza la rotación en sitio, se crea un lista llamada *end-list*, en esta lista los *gaps* son ordenados de acuerdo al ángulo iniciando desde la dirección definida por el centro del robot y la ubicación del segundo vértice que está tocando el punto rp. Las listas *init-list* y *end-list* tienen diferentes elementos porque las listas inician y terminan en diferentes direcciones angulares. Ver Figura 4.7. Note que el *gap* g_{13} no se incluye en las

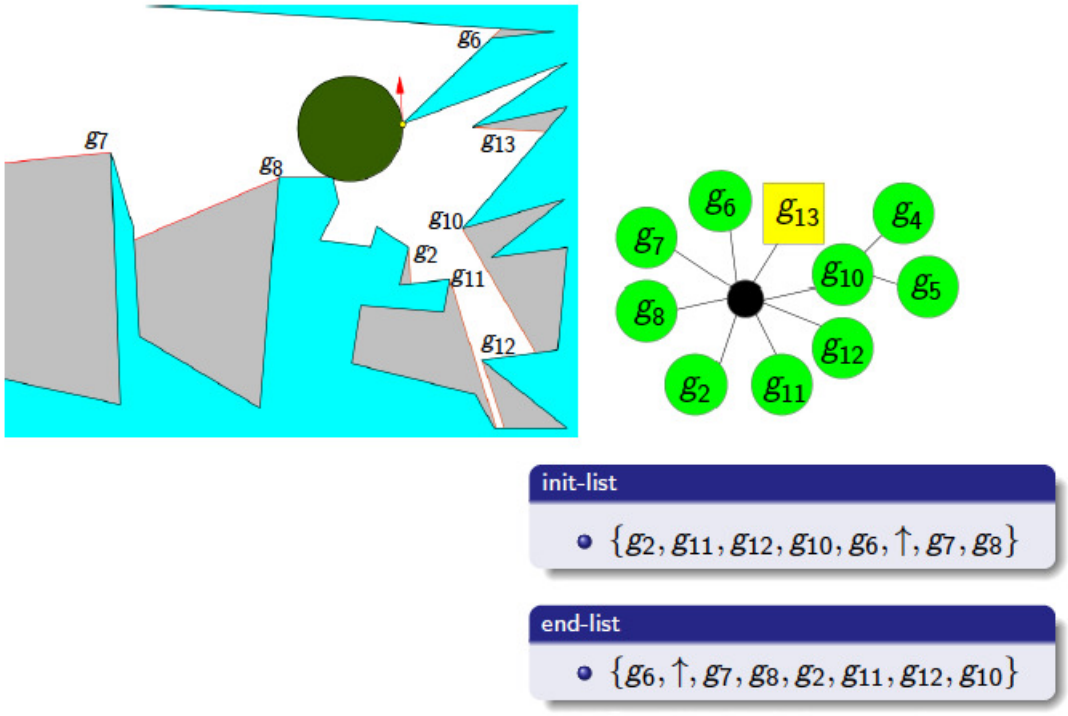


Figura 4.7: Listas *init-list* y *end-list*, y *gaps* despues de la rotación en sitio del robot.

listas, dado que este *gap* aparece durante la rotación en sitio y como se mencionó antes, en cualquier momento que un nuevo *gap* aparece en el GNT, este debe ser primitivo.

Al final de la rotación en sitio, el primer *gap* en *init-list* es el primer *gap* (en sentido contrario a las manecillas del reloj en orden angular) generado por un vértice el cual no es accesible, este *gap* es almacenado en $init_f$. El último *gap* en *end-list* es el último *gap* (en sentido contrario a las manecillas del reloj en orden angular) generado por un vértice que no es accesible, este *gap* es almacenado en end_l. Como se dijo anteriormente, la dirección de la línea rt es usada como una dirección de referencia común entre ambas listas y es almacenada en los elementos $init_{rt}$ y end_{rt}.

G_1 y G_2 son listas auxiliares que contienen subconjuntos específicos de *init-list* y *end-list* respectivamente. G_1 contiene desde $init_f$ hasta los elementos correspondientes a la dirección rt. G_2 contiene desde los elementos correspondientes a la dirección rt hasta end_l. $G_\cap$ incluye los elementos en común de G_1 y G_2. El algoritmo 1 encuentra $G_\cap$ y da como salida un GNT actualizado. En efecto, $G_\cap$ contiene los elementos en común entre G_1 y G_2 que corresponden a los *gaps* dentro de la región inalcanzable. Ver Figura 4.8. Estos *gaps* deben propagar la etiqueta de primitivos a su descendientes en el GNT.

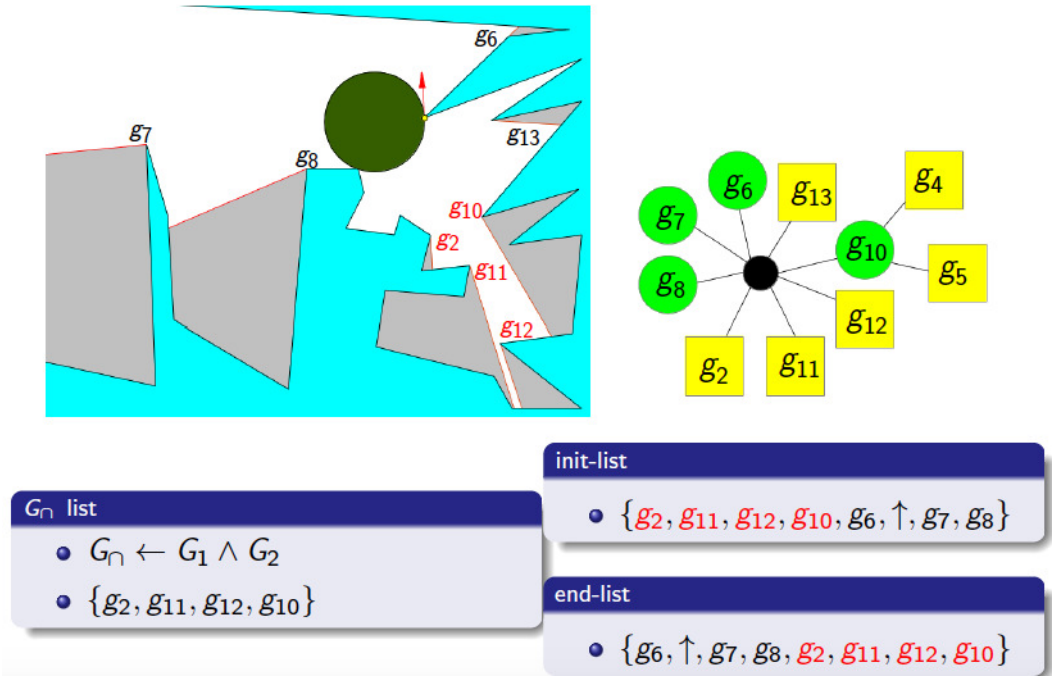


Figura 4.8: Listas G_1, G_2 y $G_\cap$

4.7. Demostraciones de algunas propiedades de la estrategia de exploración

El siguiente Lema, corresponde al caso cuando el robot toca la frontera del entorno con el punto *rp*, el caso cuando el robot toca la frontera con el punto *lp* es solamente el caso simétrico.

Lema 2. *La estrategia de exploración garantiza que todos los nodos hoja (ej. gaps codificados como nodos hoja en el GNT) son etiquetados como gaps primitivos.*

Demostración. Los *gaps* que no desaparecen son procesados por el Algoritmo 1. Si el robot está tocando ∂E con el punto *rp* entonces la lista G_1 incluye todos los *gaps* que pertenecen al intervalo angular entre la dirección definida por el centro del robot y la ubicación del punto *rp* antes de la rotación en sitio y la dirección *rt* después de la rotación en sitio. En este intervalo, el orden de los *gaps* es establecido en sentido contrario a las manecillas del reloj. Además, la lista G_2 incluye todos los *gaps* pertenecientes al intervalo angular entre la dirección *rt* después de la rotación en sitio y la dirección definida por el centro del robot y la ubicación del punto *rp* después de la rotación en sitio. En el segundo intervalo angular, los *gaps* son también ordenados en sentido contrario a las manecillas del reloj. La intersección $G_\cap$ entre G_1 y G_2 incluye solo los *gaps*, ordenados en sentido contrario a las manecillas del reloj, que se encuentran entre la dirección definida por el centro del

robot y la ubicación del punto rp antes de la rotación en sitio y la dirección definida por el centro del robot y la ubicación del punto rp después de la rotación en sitio. Estos *gaps* son generados por vértices reflejo ubicados dentro de la región inalcanzable. La observación ye_4^R detecta una región inalcanzable. La región es inalcanzable porque el sensor *bumper* del robot ha tocado ∂E en dos puntos. Durante la rotación en sitio del robot, el sensor omnidireccional se mueve desde un punto donde toca ∂E hasta el otro, por lo tanto todos los *gaps* dentro de la región inalcanzable son considerados. Debido a los posibles eventos críticos división o fusión de estos *gaps*, la etiqueta de primitivo de tales *gaps* es propagada a todos los *gaps* decendientes de estos en $G_{\cap}$. Cada vez que ocurre la observación ye_4^R, el algoritmo *local exploration* es ejecutado. Por lo tanto, todos los *gaps* codificados como nodos hoja (llamados *gaps* hoja) en el GNT son etiquetados como *gaps* primitivos. □

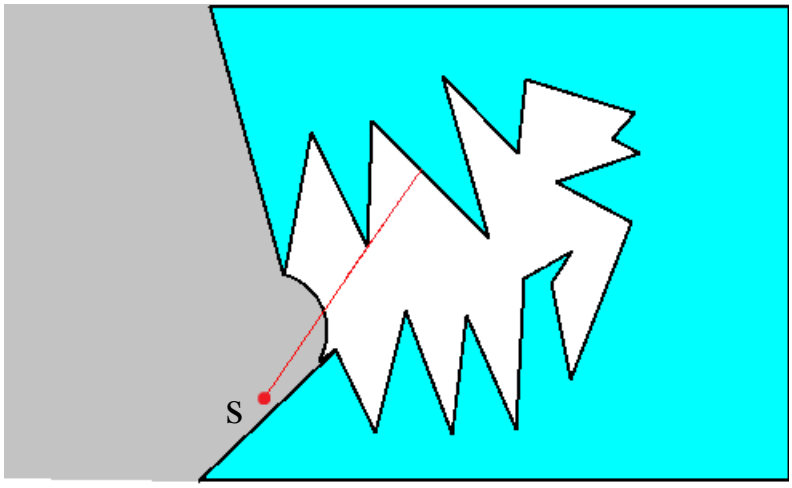


Figura 4.9: R_{na} es mostrado en blanco y R_a en gris obscuro.

Lema 3. *El robot cubre (observa) la mayor porción posible del entorno con la región de visibilidad del sensor omnidireccional.*

Demostración. La trayectoria del sensor omnidireccional durante el movimiento en rotación en sitio del robot es un arco de círculo, dicha trayectoria divide el interior del ambiente en dos regiones, llamadas región accesible R_a y región inaccesible R_{na}, de tal manera que $R_a \cap R_{na} = \emptyset$. La frontera entre estas regiones depende del radio del robot. El sensor omnidireccional es incapaz de penetrar en la región inalcanzable debido a las restricciones del espacio de configuraciones, por tanto, el arco de círculo determinado por el radio del robot es la frontera entre ambas regiones. Refiérase a la Figura 4.9. Es claro que cada rayo de luz que emerge desde cualquier fuente $s \in R_a$ la cual toca R_{na} debe cruzar la frontera de las regiones como se ve en la Figura 4.9. Si el polígono de visibilidad de s incluye una porción de R_{na}, entonces cada rayo de luz emergente desde R_a a R_{na} debe cruzar la frontera de las regiones. Por lo tanto, cada rayo de luz viajando desde cualquier punto $x \in R_a$ a R_{na} debe cruzar la frontera de las regiones. Por consiguiente un sensor

omnidireccional que sigue la trayectoria de arco de círculo garantiza la observación de la mayor parte posible de la región R_{na}. La observación ye_4^R o ye_5^L indica que hay una región inalcanzable, cada vez que esta observación ye_4^R o ye_5^L ocurre, el sensor omnidireccional se mueve sobre la frontera entre la región accesible y la inaccesible. □

Con base en los Lemas 2 y 3 el siguiente teorema establece uno de los principales resultados del trabajo de [1].

Teorema 1. *Para todos los ambientes, la estrategia de exploración modelada como una máquina de Moore M termina. Al finalizar, el robot ha cubierto todo el ambiente con la región de visibilidad del sensor omnidireccional o ha cubierto la máxima porción posible dada la geometría. Consecuentemente, el robot encuentra el landmark o declara que no existe una estrategia de exploración para encontrar dicho landmark.*

Demostración. Ya que el entorno es simplemente conectado, la estrategia de seguimiento de pared es suficiente para la exploración de todo el ambiente para un robot punto debido a la ausencia de obstáculos internos (generados por más de una clase de trayectorias homotópicas). Para un robot disco, los *gaps* que se generan por los vértices reflejo en la región alcanzable son etiquetados como *gaps* primitivos, ya que el robot es capaz de alcanzar los vértices reflejo que generan esos *gaps*, entonces esos *gaps* desaparecen. Si hay regiones inalcanzables, donde algunos *gaps* no desaparecen independientemente del movimiento del sensor, entonces el algoritmo *local exploration* es ejecutado. El Lema 2 garantiza que todos los *gaps* hoja serán etiquetados como primitivos, esta es la condición para parar la tarea de exploración. Por lo tanto, la tarea de exploración es finalizada. El Lema 3 garantiza que el robot descubre la mayor región posible del ambiente. Por lo tanto, si el subconjunto libre de colisiones del espacio de configuraciones $\mathcal{C}$ es simplemente conectado, entonces el *landmark* es encontrado. Si el subconjunto libre de colisiones del espacio de configuraciones $\mathcal{C}$ tiene varios componentes conectados, entonces el *landmark* puede o no ser encontrado. Nuevamente, por el Lema 3 el robot observa (descubre) la mayor parte posible del ambiente, por lo tanto, cuando el *landmark* no es encontrado, no existe una estrategia de exploración para encontrar el *landmark* para el componente conectado del espacio de configuraciones donde se encuentra el robot. □

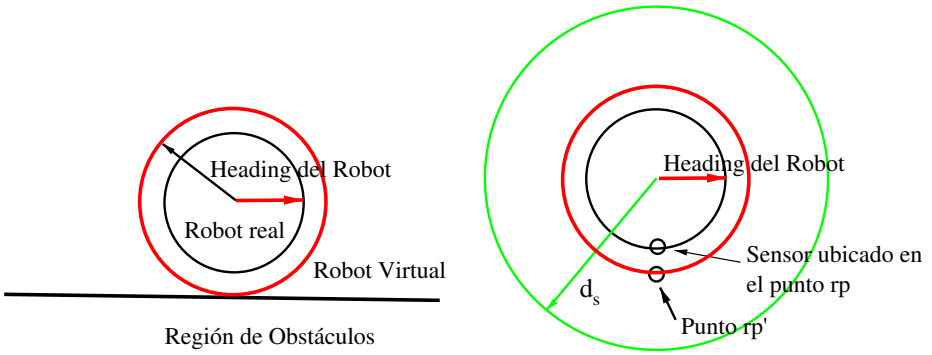
Hasta aquí se ha presentado un resumen del trabajo que es el antecedente principal de la propuesta desarrollada en el presente documento. En este trabajo se propone un esquema de control híbrido para el seguimiento de frontera del entorno manteniendo una distancia deseada diferente de cero entre el robot y ∂E. En este trabajo de tesis, el manejo de observaciones imperfectas es un componente esencial para la implementación de la estrategia de exploración presentada en [1] en un robot real, evitando colisiones con los obstáculos y cuidando el manejo de los motores al mantener la continuidad en las velocidades lineal y angular a pesar de la conmutación entre controladores de acuerdo a las diversas observaciones determinadas durante la etapa de sensado.

Capítulo 5

Observaciones Imperfectas

En los capítulos anteriores, se han presentado las condiciones teóricas que garantizan que el robot descubre la mayor región posible del entorno. Sin embargo, se asume que los controles son perfectos y que el robot se mueve en contacto con la pared (ver Sección 4.3). En esta tesis, uno de los objetivos es que el método propuesto sea capaz de funcionar correctamente con controles y observaciones imperfectas, y tratar con la dinámica del robot (es decir, las variaciones de velocidad).

Para conseguir este objetivo, el robot sigue la frontera del entorno a una distancia deseada diferente de cero. La *idea principal* es mantener un robot circular virtual de radio $d_d > r$, donde r es el radio del robot real, en contacto con ∂E. Refiérase a la Figura 5.1(a).



(a) Robot Virtual (b) Círculo para reducir la velocidad lineal

Figura 5.1: Dos discos virtuales

En general, es más flexible y práctico seguir la pared a corta distancia, en lugar de moverse en contacto con ella, ya que los controles que producen el movimiento del robot pueden ser imprecisos. Por tanto, así es menos probable que el robot colisione con los obstáculos.

Además, toda la información de retroalimentación puede ser obtenida directamente desde el sensor láser de distancia.

De modo adicional, se usa un círculo de radio $d_s > d_d$ para detectar un obstáculo cercano y reducir la velocidad lineal del robot, ver Figura 5.1(b). Llamamos rp' al punto a distancia d_d desde el centro del robot, en la dirección perpendicular al *heading* del robot y a la derecha del *heading*.

En la Sección 4.1, una observación está definida por un vector con seis elementos binarios y solo hay nueve observaciones útiles. Más adelante se muestra que los elementos del vector de observaciones representan información abstracta que puede obtenerse del sensor láser de distancia.

Por simplicidad, el sensor omnidireccional se localiza en rp. El caso cuando el sensor es localizado en lp es solamente el caso simétrico. Por esta razón, los elementos binarios lp y o no son utilizados. Así, solamente se utilizan 4 de los elementos binarios definidos en la Sección 4.2. Además, se agregan dos nuevos elementos binarios los cuales se describen más adelante.

Las mediciones del sensor omnidireccional son utilizadas para calcular diversos ángulos y distancias en marcos de referencia locales que están fijos al robot. Estos ángulos y distancias son utilizadas de dos maneras: 1) Son transformados en observaciones binarias que determinan la transición entre estados en una máquina de estados, la cual representa la estrategia completa de movimiento para la exploración con el robot. 2) Se utilizan como información de retroalimentación en un esquema de control híbrido, cuyo objetivo es la navegación para mantener al robot a una distancia deseada entre el robot y la frontera del entorno. Este objetivo se cumple mediante la convergencia de algunos errores sobre la mediciones de distancias y ángulos.

A continuación, se describe como obtener los elementos binarios de un vector de observaciones yc_i con base en las mediciones del sensor láser de distancia, también se describe las mediciones de retroalimentación que son relevantes (en términos de ángulos y distancias) y que son utilizados en cada primitiva de movimiento ejecutadas en el robot.

5.1. Observaciones para el movimiento en línea recta

Refiérase a la Figura 5.2. La línea que pasa sobre los puntos rp y rp' es llamada línea $rp - rp'$. El rayo que apunta al punto más cercano en el obstáculo sobre el segmento que el

robot está siguiendo es llamado r_{min}. Sea θ_1 el ángulo desde la línea $rp - rp'$ al rayo r_{min} medido en sentido contrario a las manecillas del reloj, $\theta_1 \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Sea d_1 la distancia más corta desde el centro del robot al segmento de línea que el robot está siguiendo.

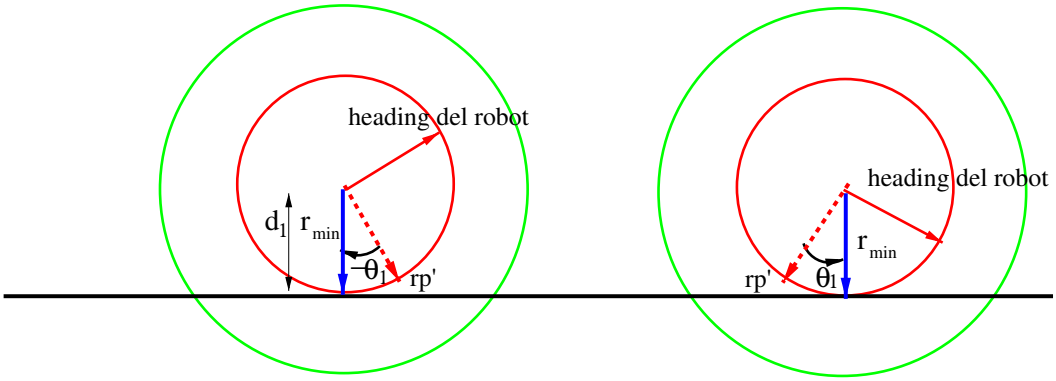


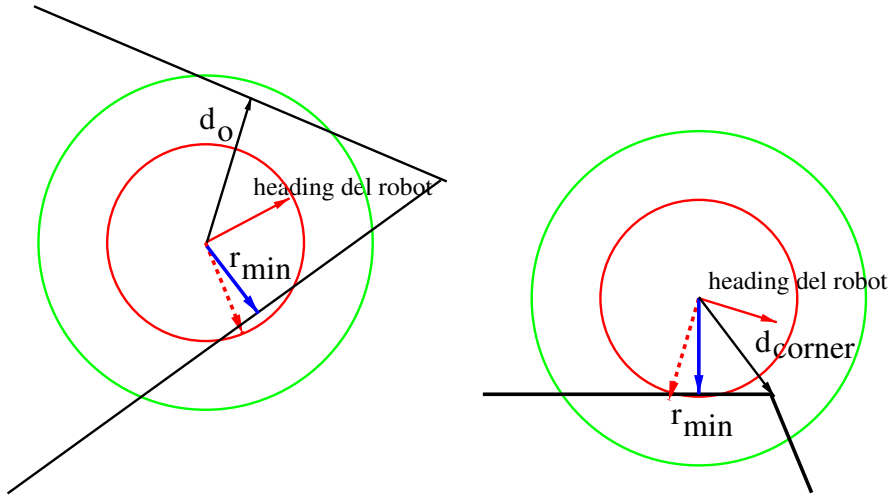
Figura 5.2: Ángulo θ_1 y rayo r_{min}

El robot reduce su velocidad lineal si un obstáculo diferente a la pared que el robot está siguiendo es más cercano que la distancia d_s, el robot también reduce su velocidad lineal si hay una esquina convexa (también llamada vértice reflejo) más cercana que la distancia d_s. Uno de los dos nuevos bits, que se utilizan en el vector de observaciones es llamado st, este bit se activa en 1 si hay un obstáculo o esquina convexa más cerca que la distancia d_s, ver la Tabla 6.1.

El sensor láser omnidireccional es utilizado para medir la distancia d_o, la cual es la distancia entre el centro del robot y el obstáculo más cercano que no pertenece a la pared que el robot está siguiendo. La distancia d_{corner} es también medida mediante el láser, d_{corner} es la distancia entre el centro del robot y la esquina visible más cercana. Ver Figuras 5.3(a) y 5.3(b).

Se utiliza un algoritmo simple de ajuste de líneas (ver Apendice A) para construir segmentos y detectar esquinas cóncavas y convexas, además, hay varios otros algoritmos conocidos para el ajuste de líneas basado en puntos [101]. De modo alternativo, una esquina convexa puede ser detectada, ya que genera una discontinuidad (*gap*), la cual aparece entre dos lecturas consecutivas en ángulo.

Se dice que el punto rp' está en contacto con un obstáculo, si hay un punto del obstáculo más cerca de rp' que una tolerancia ϵ_1. El bit rp' es puesto en 1 en la Tabla 6.1. Ver Figura 5.4 a). Se dice que hay un solo contacto en el punto rp' si hay un sector circular de radio d_d (representado por el robot virtual rojo) que intersecta un obstáculo y el punto rp' pertenece a este sector circular. El bit sc es puesto en 1 en la Tabla 6.1, ver Figura 5.4 b).



(a) d_o (b) d_{corner}

Figura 5.3: Distancias d_o y d_{corner}.

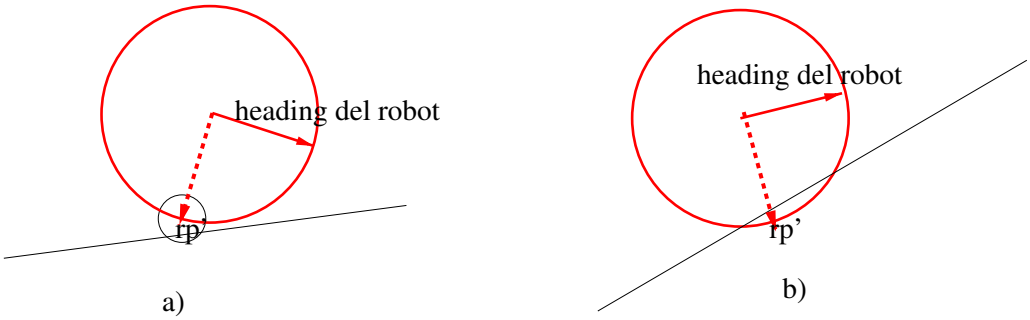


Figura 5.4: Un solo contacto en rp'

Se dice que el *heading* del robot está alineado con la pared que el robot está siguiendo si $|\theta_1| < \epsilon_2$ y no hay puntos de un obstáculo que no pertenezcan a la pared que el robot está siguiendo, más cercanos al centro del robot en un radio d_d. Si esta condición se mantiene, el bit *aligned* es puesto en 1 en la Tabla 6.1.

Los controladores SLW y $SLWD$ presentados en el Capítulo 6 utilizan la distancia d_1 y el ángulo θ_1 como información de retroalimentación para mantener al robot siguiendo la pared y mantener la alineación con esta. Adicionalmente, el controlador $SLWD$ utiliza las distancias d_o ó d_{corner} como información de retroalimentación para reducir la velocidad lineal del robot cuando un obstáculo o una esquina convexa se aproximan al robot, ver Capítulo 6.

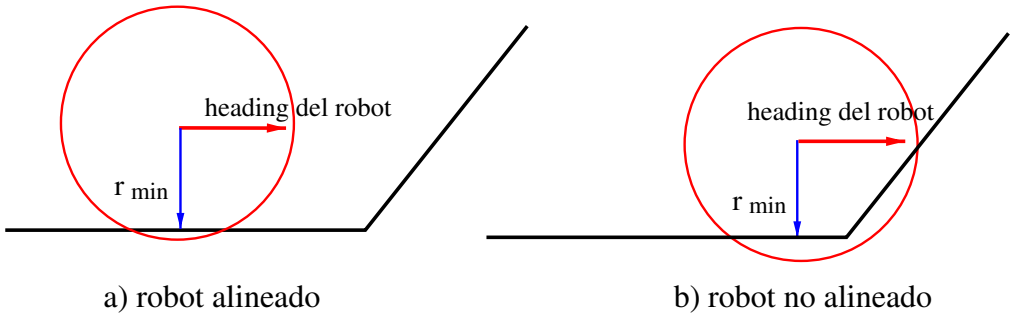


Figura 5.5: Alineación del robot

5.2. Observaciones para la rotación en sitio

Si el robot está en contacto con un obstáculo y el robot no está alineado con este, o si hay un bicontacto entre el robot y una región del obstáculo, entonces el robot debe rotar en sitio.

Se dice que el robot tiene un solo contacto con un obstáculo si un sector circular de del disco virtual (disco rojo) intersecta una región del obstáculo. El bit *sc* se pone en 1 en la Tabla 6.1. La Figura 5.6 a) muestra el caso cuando el robot está en contacto con un segmento poligonal del entorno y este no está alineado con este segmento. El bit *aligned* en la Tabla 6.1 es puesto en 0. Este caso pasa cuando el robot se mueve desde el interior del entorno poligonal para alcanzar la frontera de la región obstáculo.

La Figura 5.6 b) muestra el caso cuando el robot está en bicontacto con más de un segmento poligonal del entorno. Se dice que hay un bicontacto si dos diferentes sectores circulares del disco virtual intersectan la región del obstáculo. El bit *bc* se pone en 1 en la Tabla 6.1.

Se dice que el robot tiene un solo contacto con un obstáculo y no está alineado con este, si el disco virtual (representado por el disco rojo de radio d_d) intersecta más de un segmento poligonal del entorno, y el *heading* del robot no está alineado con el último segmento en sentido contrario a las manecillas del reloj, ver Figura 5.6 c). El bit *sc* se pone en 1 y el bit *aligned* es puesto en 0 en la Tabla 6.1. Este caso pasa cuando el robot está siguiendo un segmento y se encuentra una esquina cóncava. En contraste, se dice que el robot tiene un solo contacto con un obstáculo y el robot está alineado con este si el disco virtual intersecta más de un segmento poligonal del entorno, y el *heading* del robot está alineado con este último segmento en sentido contrario a las manecillas del reloj, ver Figura 5.6 d). El bit *sc* se pone en 1 y el bit *aligned* se pone en 1 en la Tabla 6.1.

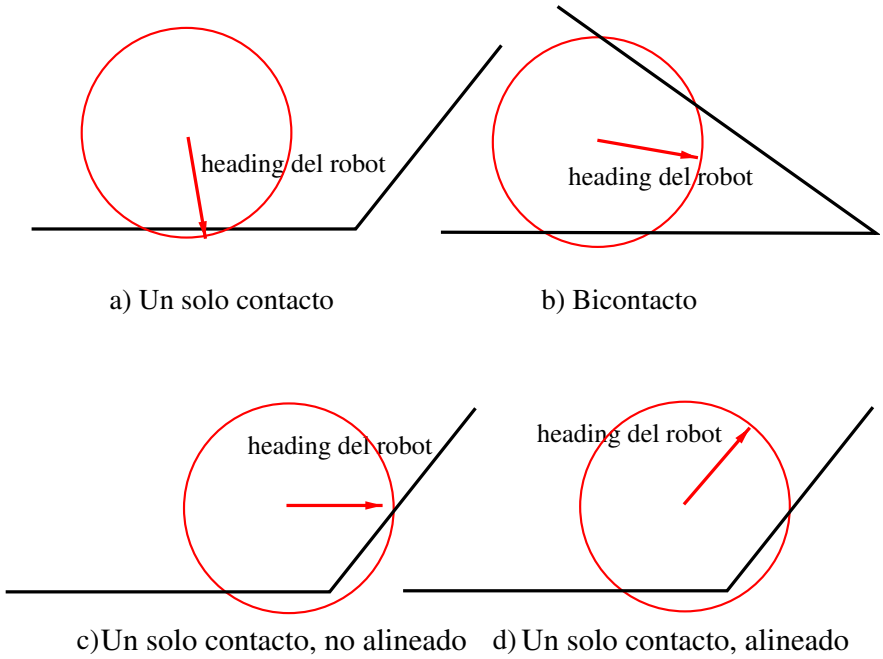


Figura 5.6: Un contacto, bicontacto y robot no alineado

Si el robot con forma de disco intersecta más de un segmento poligonal, entonces hay ocasiones en que el robot deberá ejecutar dos o más rotaciones consecutivas. El robot puede estar alineado con un segmento, sin embargo, puede haber otro obstáculo bloqueando al robot.

El uso del ajuste de líneas permite encontrar el último segmento en sentido contrario a las manecillas del reloj, de la región de la frontera del obstáculo para alinear el *heading* del robot con dicho segmento. Ver en Apéndice A, un algoritmo simple de ajuste de líneas que encuentra el segmento con el que el robot se debe alinear.

El algoritmo de ajuste de líneas utilizado considera que el robot (y por tanto también el sensor omnidireccional) está dentro del entorno poligonal y no tiene acceso al mapa, solamente tiene acceso a los puntos medidos con el láser que son visibles, esto equivale a razonar sobre el polígono de visibilidad, ver Figura 5.7. En la figura, los segmentos azules son segmentos visibles y los segmentos rojos son *gaps* (también llamados segmentos libres).

θ_2 es el ángulo entre la línea $rp - rp'$ y el rayo r_{min}, $\theta_2 \in (0, \pi)$. Ver Figura 5.8.

El controlador RP que se presenta en el Capítulo 6, utiliza el ángulo θ_2 como información de retroalimentación para realizar la rotación en sitio del robot.

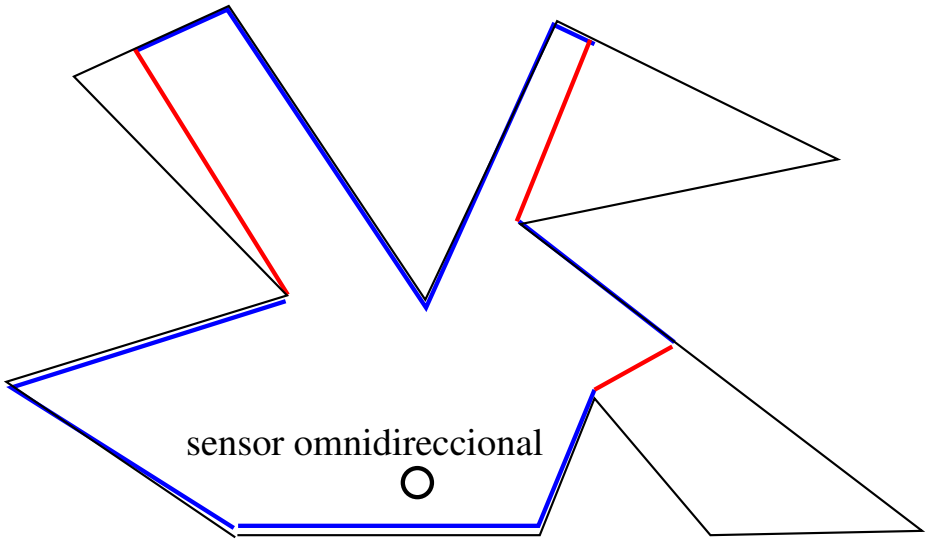


Figura 5.7: Polígono de Visibilidad

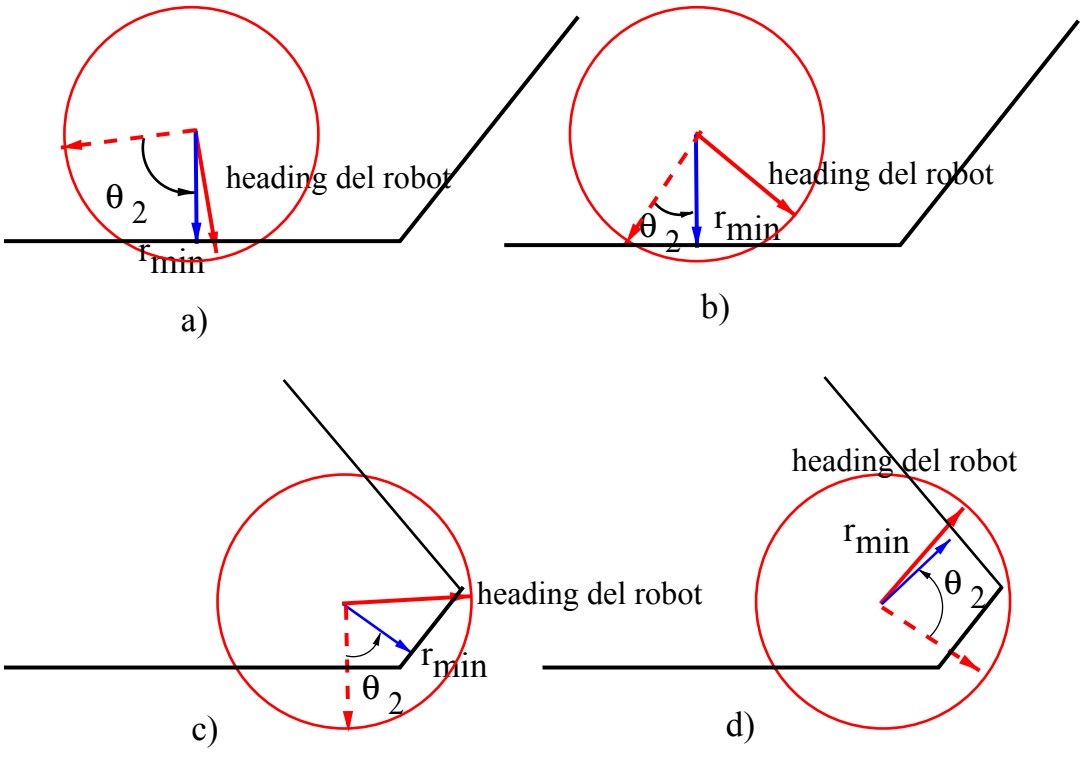


Figura 5.8: Ángulo θ_2 y rayo r_{min}

Si el *heading* del robot está alineado (es decir, $|\theta_2| < \epsilon_2$) con el último segmento poligonal, en sentido contrario a las manecillas del reloj, de la frontera de la región del obstáculo entonces la rotación en sitio finaliza con éxito, ver Figura 5.8 d).

El nuevo bit llamado $rp' - e$ (see Tabla 6.1), se pone en 1 si el punto rp' está más cerca de la esquina convexa que una tolerancia ϵ_1; esta condición también puede finalizar la rotación en sitio. Ver Figura 5.9 d).

Si el disco virtual está en bicontacto con la región poligonal, entonces solamente el último sector del disco en sentido contrario de las manecillas del reloj debe ser considerado para finalizar la rotación en sitio.

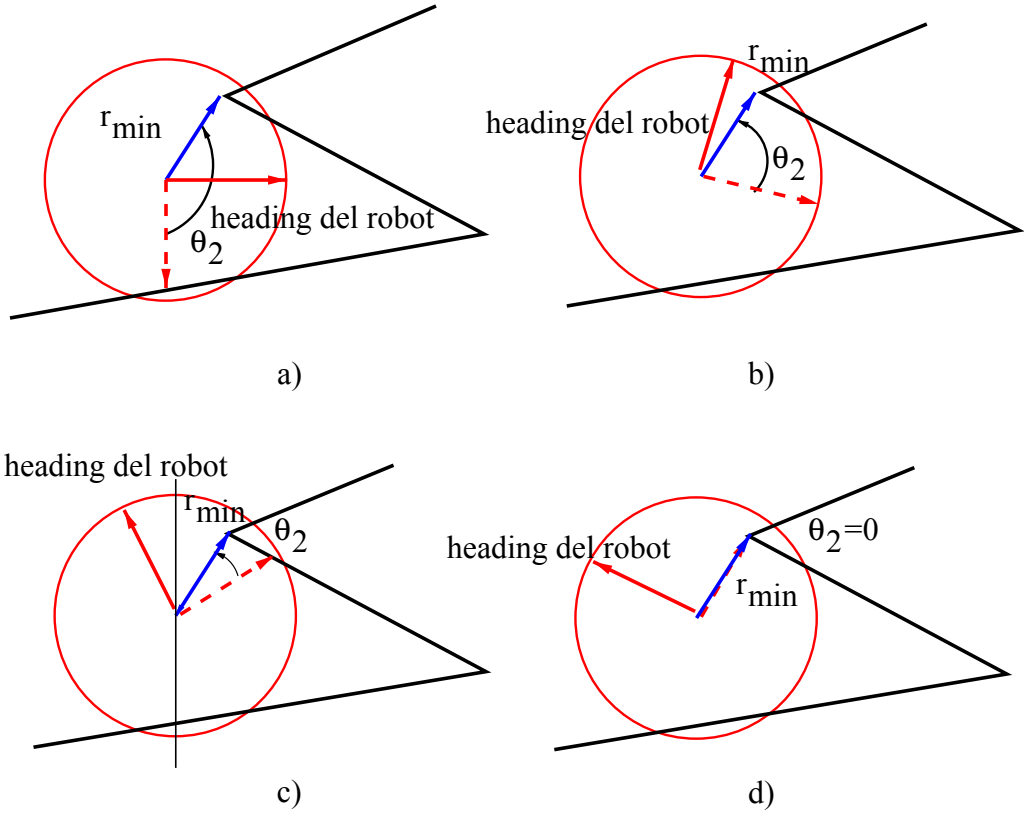


Figura 5.9: Ángulo θ_2 y rayo r_{min}, apuntando a una esquina convexa

5.3. Observaciones para la rotación con respecto a una esquina convexa

El robot rota con respecto a una esquina convexa o punto rp', siguiendo un arco de círculo, siempre que el punto rp' este más cercano a una esquina convexa que una tolerancia ϵ_1. Esta condición pone a 1 el bit $rp' - e$. El rayo que va desde el centro del robot y apunta a la esquina convexa es llamado r_{corner}.

El ángulo θ_3 se utiliza como información de retroalimentación para este tipo de rotación. Aquí se describe el caso general, en el cual la línea $rp - rp'$ no es colineal con el rayo r_{corner}, esto pasa por la imprecisión en el movimiento del robot al aproximarse a la esquina convexa (Ver Figura 5.10). Se utilizan dos distancias para calcular θ_3, la distancia a la esquina d_{corner} y la distancia al obstáculo en la dirección de la línea $rp - rp'$, esta segunda distancia es llamada d_w. Se utiliza la ley de cosenos para calcular un ángulo auxiliar denotado por A, así $\theta_3 = \frac{\pi}{2} - A$. Nótese que el arco de círculo que el robot ejecuta está centrado en el punto rp' y no en la esquina convexa. Nótese también que algunas veces el robot alinea su *heading* a la línea virtual (denotada como s_v), ver Figuras5.10 a) y b). Sin embargo, a medida que la línea que pasa sobre el punto rp y rp' se vuelve perpendicular a este segmento de línea, se debe sensar y considerar el segmento correcto, ver Figura 5.10 c). El arco de círculo finaliza cuando θ_3 es más pequeño que una tolerancia ϵ_2, esto es equivalente a tener el *heading* del robot alineado con el segmento posterior a la esquina en sentido contrario a las manecillas del reloj, ver Figura 5.10 d).

Una complicación adicional puede suceder al calcular el ángulo que el robot debe rotar, cuando este se mueve en arco de círculo alrededor de una esquina convexa. Esta complicación corresponde a que el robot puede chocar con un obstáculo antes de que su *heading* este alineado con el segmento posterior a la esquina (ver Figura 5.11). Se utiliza una técnica simple de ajuste de líneas para detectar posibles puntos de colisión que no pertenezcan al segmento de línea posterior a la esquina convexa. El método de ajuste de líneas utilizado se presenta en el Apéndice A.

Así, si existe un punto de colisión potencial más cercano al robot que una distancia d_s, entonces se utiliza un ángulo auxiliar θ_4 el cual es calculado para determinar el ángulo que el robot debe rotar. El cálculo de θ_4 se basa en la siguiente observación: cuando el robot rota siguiendo un arco de círculo, todos los puntos sobre la periferia del disco virtual (disco rojo) rotan respecto a un punto dado y con el mismo ángulo. Por simplicidad, se describe el procedimiento suponiendo que la línea $rp - rp'$ y el rayo r_{corner} son colineales, entonces el robot rota siguiendo un arco de círculo centrado en la esquina. Sin embargo, en general

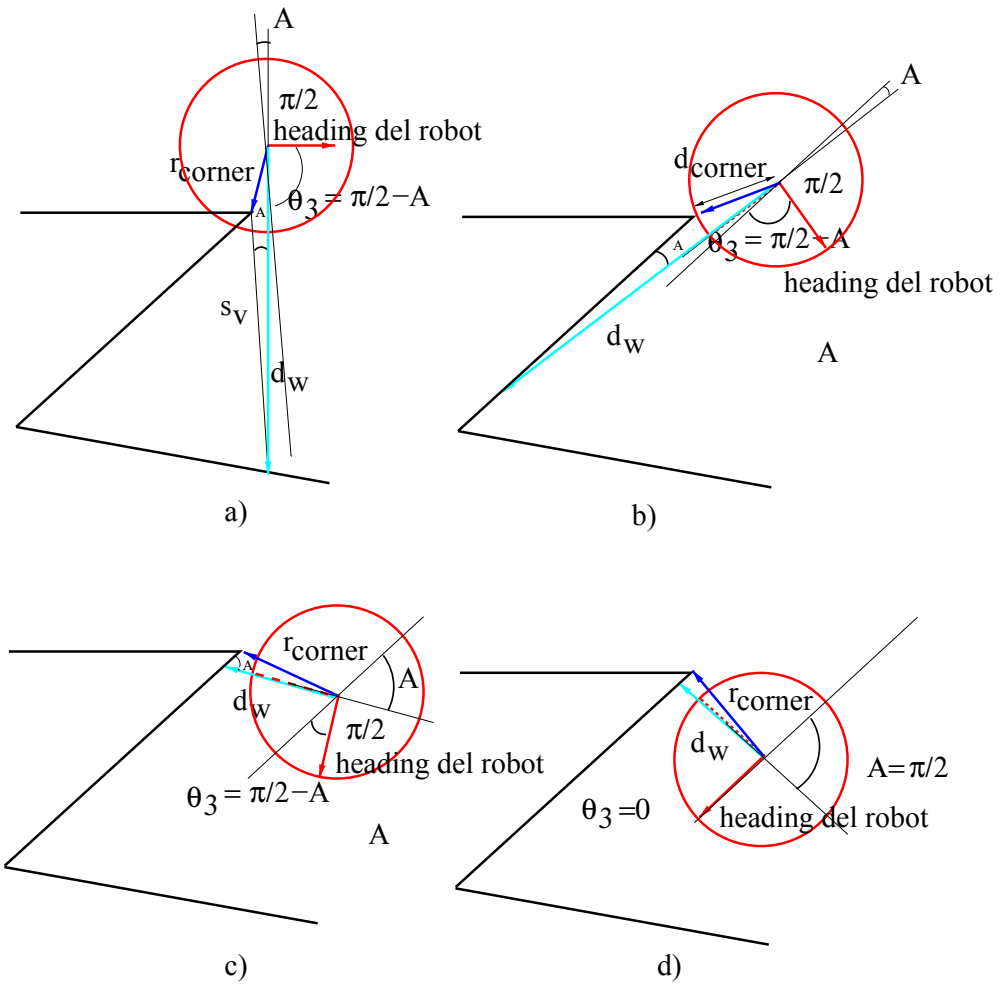


Figura 5.10: Ángulo θ_3, línea $rp - rp'$ y rayo r_{corner} no son colineales, rotación con respecto al punto rp'

la línea $rp - rp'$ y el rayo r_{corner} no son colineales, por lo tanto el robot rota siguiendo un arco de círculo respecto al punto rp'.

Para calcular θ_4, se utilizan las distancias d_{corner} y d_o, d_o es la distancia al punto del obstáculo más cercano al centro del robot. También se utiliza la ley de cosenos para calcular la distancia entre la esquina convexa y el punto a distancia d_o desde el centro del robot. Esta distancia es llamada d_{co}. Para encontrar el punto que colisiona con el obstáculo más cercano (desde el centro del robot), se utiliza un círculo centrado en la esquina convexa (centro de rotación) con un radio d_{co}. Se procede a intersectar el círculo centrado en la esquina convexa y el círculo que representa al disco virtual (disco rojo). El punto de intersección más cercano al obstáculo más cercano (desde el centro del robot) es llamado punto IC. Así, θ_4 es el ángulo entre el rayo que va desde la esquina al punto IC y el rayo que va desde la esquina al punto del obstáculo más cercano al centro del robot. La Figura 5.11 a) muestra el caso cuando el obstáculo es una esquina y la Figura 5.11 b) cuando el obstáculo es un segmento. Finalmente el ángulo que el robot debe rotar alrededor de la esquina es $\min\{\theta_3, \theta_4\}$.

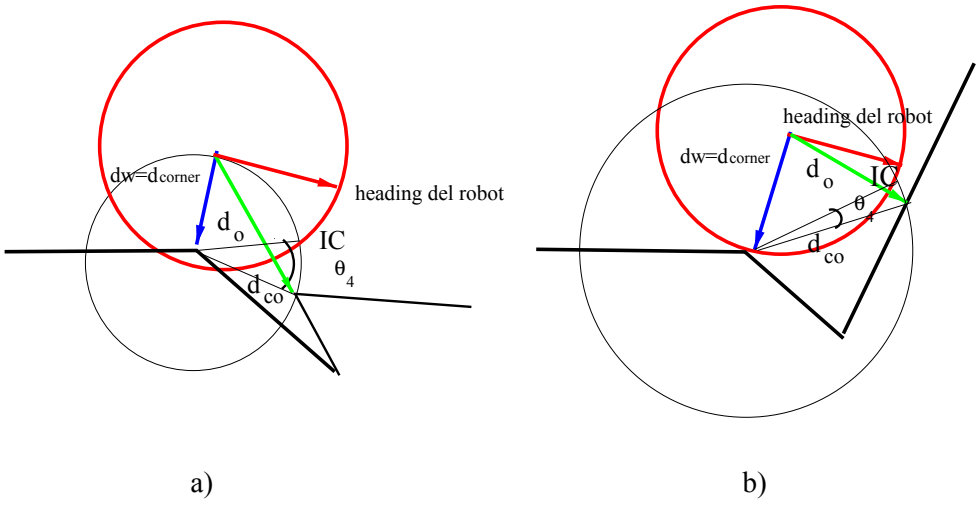


Figura 5.11: Ángulo θ_4, línea $rp - rp'$ y rayo r_{corner} colineales, rotación con respecto a la esquina convexa.

El ángulo θ_4 es calculado en cada iteración de este método, nótese que es posible decidir cual de los ángulos θ_3 o θ_4 es más pequeño en cada instante de tiempo.

El controlador AC que se presenta en el Capítulo 6 utiliza ya sea el ángulo θ_3 o el ángulo θ_4 como información de retroalimentación para realizar la rotación en arco de círculo respecto a la esquina convexa.

5.4. Algunas aproximaciones

Se hace hincapié en que el método permite realizar aproximaciones, para ello se utilizan dos tolerancias. La tolerancia ϵ_1 es el radio del círculo para modelar el punto rp' que determina si hay o no una esquina convexa tocando el punto rp' o si hay o no contacto con la pared en el punto rp' del robot. La tolerancia ϵ_2 es una aproximación angular que determina si el robot se encuentra o no alineado con la pared que se está siguiendo, esta misma tolerancia es utilizada para los ángulos θ_1, θ_2, θ_3 y θ_4. La sintonización de estos parámetros es un compromiso entre una acción precisa del control y la robustez contra la las lecturas imperfectas proporcionadas por el sensor.

Capítulo 6

Máquina de Estados Capaz de Lidiar con Incertidumbre en Sensado y Control y Controladores Basados en Retroalimentación

6.1. Máquina de estados y esquema de control conmutado

Análogamente al caso en el cual el robot se mueve en contacto con la frontera del entorno, es posible obtener una nueva máquina de Moore, para el caso donde el robot tiene como objetivo moverse a una distancia deseada con respecto a la frontera del entorno.

En este último caso, una o más observaciones activan un controlador en específico. En la Tabla 6.1 se presentan las observaciones que activan cada controlador.

$yc_i =$	$(rp',$	$sc,$	$bc,$	$st,$	$rp' - e,$	$aligned)$	Control
$yc_1 =$	(0,	0,	0,	0,	X,	X)	SLI
$yc_2 =$	(0,	0,	0,	1,	X,	X)	$SLID$
$yc_3 =$	(1,	X,	X,	0,	X,	1)	SLW
$yc_4 =$	(1,	X,	X,	1,	X,	1)	$SLWD$
$yc_5 =$	(X,	1,	0,	X,	0,	0)	RP
$yc_6 =$	(X,	0,	1,	X,	X,	0)	RP
$yc_7 =$	(1,	X,	X,	X,	1,	0)	AC

Tabla 6.1: Observaciones yc_i

Recuérdese que los controladores utilizan los ángulos θ_1, θ_2, θ_3 y θ_4 y las distancias

d_1, d_o y d_{corner} como información de retroalimentación para ejecutar los movimientos del robot.

La relación entre una observación y la activación de un controlador está dada por:

- $yc_1 \rightarrow SLI$
- $yc_2 \rightarrow SLID$
- $yc_3 \rightarrow SLW$
- $yc_4 \rightarrow SLWD$
- $yc_5 \vee yc_6 \rightarrow RP$
- $yc_7 \rightarrow AC$

Donde $\vee$ significa "or".

El robot ejecuta primitivas de movimiento básicas: Línea recta, rotación en sitio y arco de círculo, sin embargo, la información sensada es usada para corregir posibles desviaciones en las primitivas de movimiento.

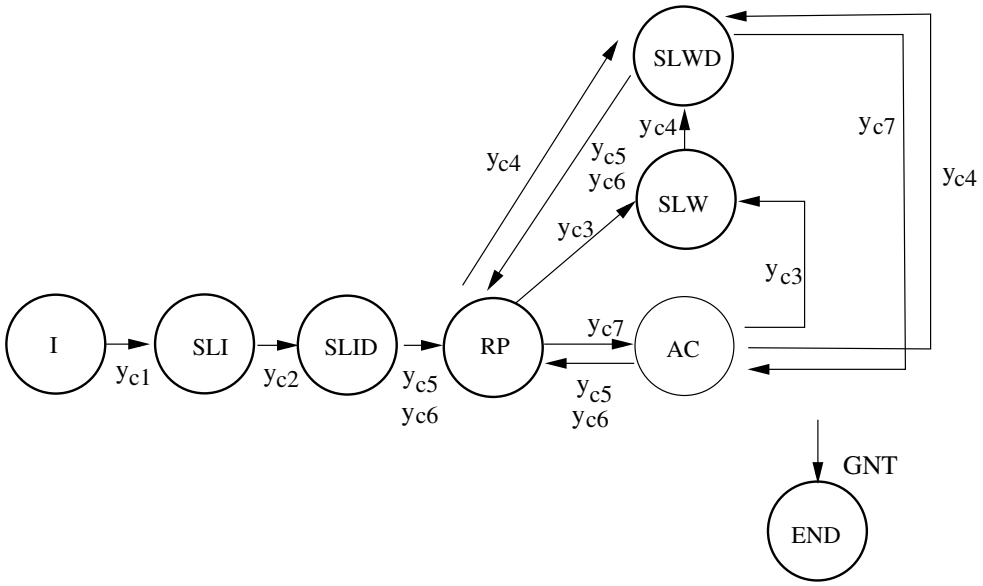


Figura 6.1: Máquina de estados finitos.

La máquina de Moore define las posibles transiciones entre estados las cuales están dadas por una o más observaciones y la condición de terminación de la exploración que

determina el GNT. En la Figura 6.1 se muestra una representación gráfica de la máquina de Moore.

La activación de un controlador en específico depende tanto de la observación como del estado en la máquina de estados, así el autómata restringe las transiciones de estados filtrando las observaciones espurias debido al ruido en las lecturas del sensor. En este enfoque, la etapa de planificación corresponde al diseño de la FSM y se realiza antes de la ejecución, una vez que esto se logra, para cualquier instante durante la ejecución y para cualquier entorno, el método es reactivo, él relaciona las observaciones con los controles.

El GNT da la condición de terminación a la tarea de exploración. Un enlace al GNT representa una consulta al GNT verificando si todos los nodos hoja están marcados como primitivos.

El algoritmo *local exploration* presentado en la sección 4.6 está diseñado para el caso en el cual el robot se mueve en contacto con la frontera del entorno. El algoritmo es utilizado para detectar gaps, los cuales son generados por vértices reflejo ubicados dentro de una región inalcanzable.

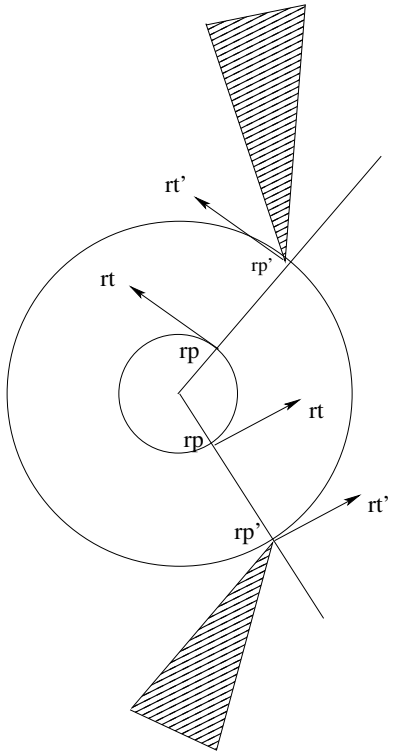


Figura 6.2: Robot virtual de radio d_d

Cuando el robot no se mueve en contacto con la frontera del entorno, ya que el sensor no está ubicado en el punto rp', el sensor no descubre la misma porción del entorno, en comparación con el caso en que el sensor está ubicado en el punto rp'. Sin embargo, el algoritmo *local exploration* puede ser utilizado para etiquetar los gaps observados que son generados por los vértices reflejo que se ubican dentro de una región inalcanzable como gaps primitivos. Una región inalcanzable se presenta, si el robot de radio d_d no puede alcanzar esta región.

Recuérdese que el algoritmo *local explortion* está basado en la dirección del primer punto de contacto, la dirección del segundo punto de contacto y la dirección de la línea rt. Las líneas rt y rt' tienen la misma dirección, y también la dirección de la línea desde el centro del robot al punto rp es la misma que la dirección de la línea que pasa por los puntos rp y rp', ver Figura 6.2. Por lo tanto, el orden angular de la dirección de esas líneas con respecto a los gaps no cambian a pesar de que la frontera del entorno esté tocando el punto rp o rp'. Así, el algoritmo *local exploration* puede ser utilizado y la estrategia de exploración termina.

La única diferencia es que una nueva observación iniciará la ejecución del algoritmo. Esta observación es un caso particular de la observación yc_6 en la cual el punto rp' está en contacto con la frontera del entorno.

Desafortunadamente, ya que el robot no se mueve en contacto con el entorno, la porción de una región inalcanzable que se cubre con la región de visibilidad del sensor omnidireccional es en general más pequeña comparada con un robot que se mueve en contacto con el entorno.

6.2. Tratando con acciones imperfectas: controladores basados en retroalimentación

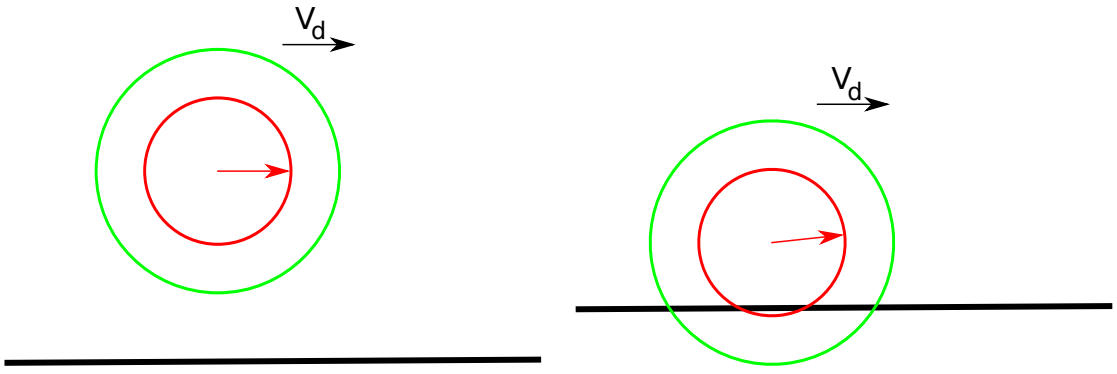
En esta sección se detalla el esquema de control conmutado que se propone. La activación de un controlador depende tanto de la observación como del estado en la máquina de estados finitos. En el esquema de control V_c es la velocidad actual en el momento cuando sucede un evento crítico, ya sea que un controlador sea activado (porque una observación cambia) o al ejecutar un controlador hay un cambio de medición utilizada para la retroalimentación.

6.2.1. Línea recta desde el interior (SLI)

Este controlador es activado cuando la observación yc_1 es detectada. En este caso, el robot está en el interior del entorno como se muestra en la Figura 6.3(a). El controlador tiene que dirigir al robot en línea recta hacia la frontera del entorno. Para hacerlo, la velocidad angular se fija a cero y la velocidad lineal se incrementa suavemente desde V_c (esta puede ser cero) hasta la máxima velocidad lineal deseada V_d. Nótese que V_c es la velocidad actual en el momento que el controlador es activado porque hubo un cambio de observación; esta velocidad es denotada por $V_{c|t=0}$. Así, el controlador busca alcanzar la velocidad V_d en lazo abierto ya que no hay información disponible para la retroalimentación. El controlador SLI es el único que trabaja en lazo abierto. Dicho comportamiento se logra mediante el siguiente controlador:

$$\begin{aligned} V &= \frac{V_d - V_{c|t=0}}{2}(1 + \tanh(\alpha(t-\beta))) + V_{c|t=0} \\ \omega &= 0 \end{aligned} \tag{6.1}$$

donde α es un factor de escala y β es un desplazamiento en el tiempo. Estos parámetros están relacionados con la duración de la transición de la velocidad que va desde $V_{c|t=0}$ hasta V_d, estos se calculan de modo tal que no se exceda la aceleración máxima del robot.



(a) Línea recta desde el interior (b) Línea recta al seguir una pared

Figura 6.3: Primitiva de movimiento en Línea Recta.

6.2.2. Línea recta desde el interior con desaceleración ($SLID$)

La observación que acciona este controlador es yc_2. En este caso, el robot está en el interior del entorno y este se está moviendo en línea recta (con velocidad angular cero). Un

obstáculo es detectado a una distancia menor que d_s como se muestra en la Figura 6.4.

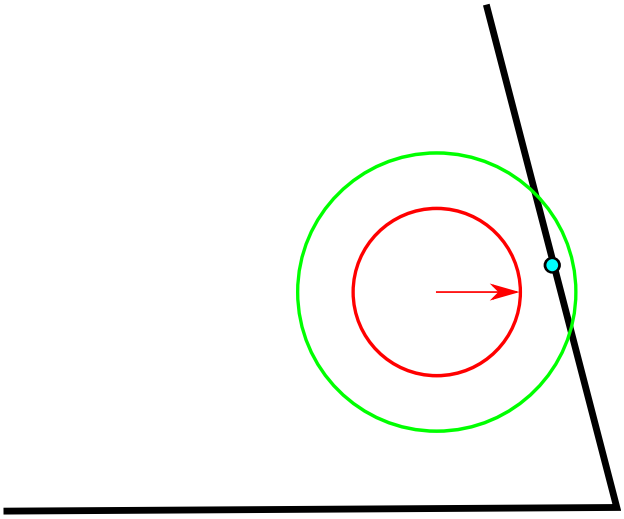


Figura 6.4: Línea recta desde el Interior con Desaceleración.

El robot reduce su velocidad líneal hasta detenerse cuando la distancia al obstáculo es igual a d_d. Para lograrlo, se define el controlador que se muestra a continuación:

$$\begin{aligned} V &= k_1 e_o \\ \omega &= 0 \end{aligned} \tag{6.2}$$

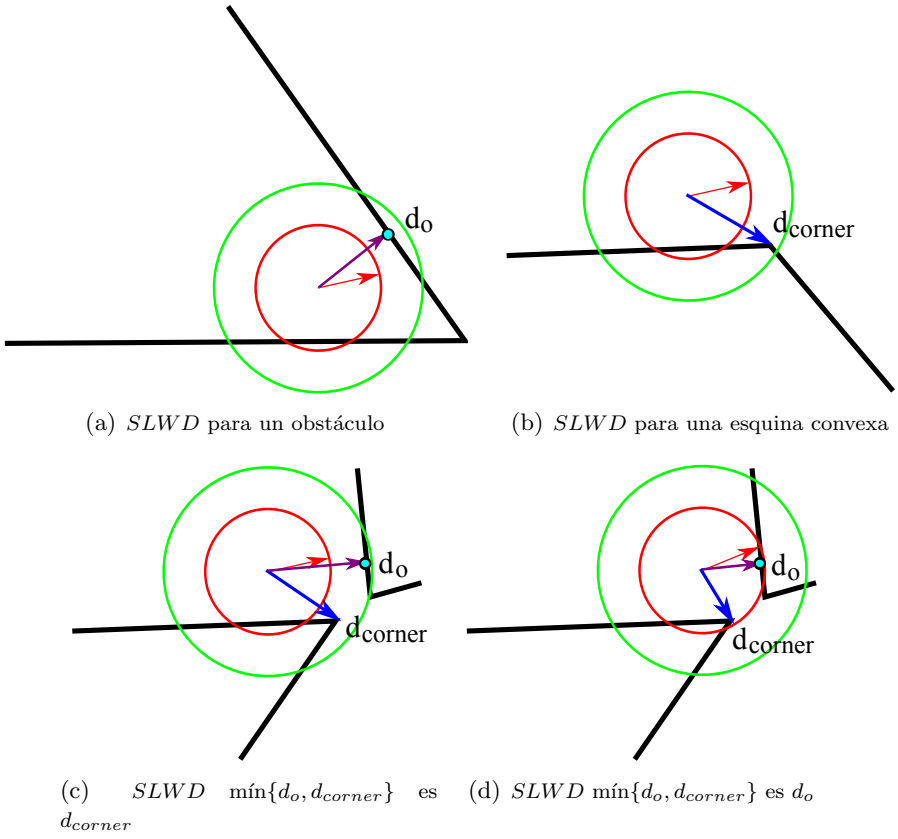
donde $e_o = d_d - d_o$, k_1 es una ganancia de control dada por $k_1 = \frac{V_{c|t=to}}{d_d - d_s}$. Aquí $V_{c|t=to}$ es la velocidad del robot en el momento que se detecta un obstáculo más cercano que una distancia d_s. Esta ganancia en particular permite mantener la continuidad en la velocidad lineal al momento que el controlador conmuta desde SLI a $SLID$.

6.2.3. Línea recta al seguir una pared (SLW)

La observación que activa este controlador es yc_3. La Figura 6.3(b) muestra un caso donde el controlador SLW es utilizado. Similar al controlador SLI, este busca alcanzar la velocidad deseada V_d mediante una transición suave desde la velocidad actual $V_{c|t=0}$, aquí $V_{c|t=0}$ es la velocidad del robot en el momento que el controlador es activado. Además, la orientación del robot debe ser controlada para alinear el *heading* del robot con la pared, esto es logrado mediante un controlador con dos componentes de retroalimentación que están en términos de distancia y desviación angular. El controlador está dado por:

$$\begin{aligned} V &= \frac{V_d - V_{c|t=0}}{2}(1 + \tanh(\alpha(t-\beta))) + V_{c|t=0} \\ \omega &= k_2 e_d + k_3 \theta_1 \end{aligned} \tag{6.3}$$

donde $e_d = d_d - d_1$, k_2, k_3 son ganancias de control. Nótese que estos controladores inician cuando el robot está detenido y alineado con la pared. Así, la velocidad angular es cercana a cero mientras este controlador está activo y se logra mantener la continuidad de velocidad angular y lineal.



(a) $SLWD$ para un obstáculo

(b) $SLWD$ para una esquina convexa

(c) $SLWD$ mín$\{d_o, d_{corner}\}$ es d_{corner}

(d) $SLWD$ mín$\{d_o, d_{corner}\}$ es d_o

Figura 6.5: Línea recta al seguir una pared con desaceleración ($SLWD$).

6.2.4. Línea recta al seguir una pared con desaceleración ($SLWD$)

En este caso el robot está siguiendo la pared corrigiendo su orientación a través de su velocidad angular como en el controlador SLW. La diferencia que existe con el controlador SLW es, si hay un obstáculo o esquina convexa que haya sido detectado a una distancia

menor que d_s, entonces el robot debe reducir su velocidad lineal desde el valor actual hasta cero. Ver Figuras 6.5(a) y 6.5(b). El robot se detiene cuando la distancia al obstáculo o a la esquina convexa es igual a d_d. Si un obstáculo o una esquina convexa son detectados a distancia menor que d_s, entonces el robot debe desacelerar suavemente hasta detenerse cuando mín$\{d_o, d_{corner}\}$ sea igual a d_d. Ver Figuras 6.5(c) y 6.5(d). La observación yc_4 activa el controlador $SLWD$, el cual está definido como

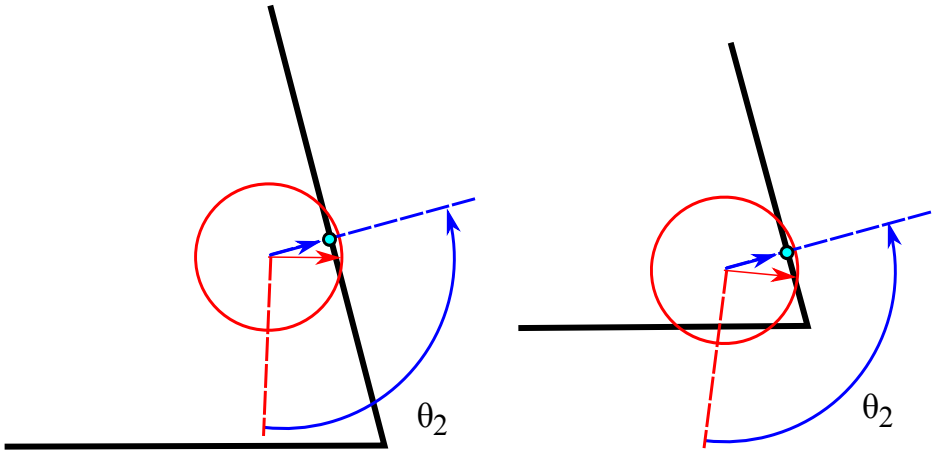
$$\begin{aligned} V &= k_4 e_p \\ \omega &= k_2 e_d + k_3 \theta_1 \end{aligned} \tag{6.4}$$

donde $e_p = \bar{e}_d - e_o$, $e_o = d_d - \text{mín}\{d_o, d_{corner}\}$ y k_2, k_3 y k_4 son ganancias de control. La señal de referencia $\bar{e}_d$ se fija dependiendo del estado previo en la FSM. Si el estado previo es SLW entonces $\bar{e}_d = 0$ y la ganancia de control debe ser $k_4 = \frac{V_{c|t=to}}{d_d - d_s}$. Aquí $V_{c|t=to}$ es la velocidad del robot en el momento que un obstáculo se encuentra más cerca que una distancia d_s. La ganancia k_4 es ajustada utilizando la ecuación anterior para evitar una discontinuidad en la velocidad del robot.

Si el estado previo es RP o AC entonces el controlador $SLWD$ inicia cuando el robot se encuentra detenido. En este caso, la señal de referencia es puesta como un perfil que varía con el tiempo $\bar{e}_d = \frac{e_{o|t=0}}{2}(1 + \cos(\frac{\pi t}{\tau_1}))$. Este control de seguimiento de referencia genera una velocidad lineal suave que inicia en cero y termina en cero en un lapso de tiempo τ_1. Así, se logra mantener la continuidad en la velocidad lineal si el controlador $SLWD$ es activado y hay obstáculos más cercanos que una distancia d_s. Si el obstáculo más cercano al robot cambia (de una esquina a una pared o viceversa), entonces la ganancia k_4 es modificada como $k_4 = \frac{V_{c|t=ts}}{d_d - \text{mín}\{d_o, d_{corner}\}}$ para mantener la continuidad de la velocidad lineal. Aquí $V_{c|t=ts}$ es la velocidad del robot en el momento que ocurre el cambio de obstáculo más cercano al robot que una distancia d_s. Este controlador también mantiene la continuidad en la velocidad angular.

6.2.5. Rotación en sitio (RP)

Este controlador es activado con dos diferentes observaciones. La primera ocurre con la observación ye_5, la cual sucede cuando el robot está a una distancia d_d de la frontera del entorno pero este no está alineado con el entorno. Ver Figura 6.6(a). La segunda ocurre cuando la observación yc_6 es medida. Esto pasa cuando hay multicontacto entre el robot y la frontera del entorno, ver la Figura 6.6(b). En ambos casos el ángulo θ_2 debe ser llevado



(a) RP con respecto a un punto de contacto sobre el disco de radio d_d

(b) RP con múltiples contactos sobre el disco de radio d_d

Figura 6.6: Primitiva de movimiento de rotación en sitio (RP).

desde su valor inicial hasta cero de manera suave. Para lograrlo, se propone el siguiente controlador de seguimiento de trayectoria:

$$\begin{aligned} V &= 0 \\ \omega &= k_5 e_{\theta_2} \end{aligned} \tag{6.5}$$

donde $e_{\theta_2} = \theta_d - \theta_2$, $\theta_d = \frac{\theta_{2|t=0}}{2}\left(1 + \cos\left(\frac{\pi t}{\tau_2}\right)\right)$ y k_5 es una ganancia de control. Nótese que las observaciones yc_5 y yc_6 suceden cuando el robot está detenido (El estado previo en la FSM puede ser $SLID$, $SLWD$ o AC, los cuales finalizan con el robot sin movimiento).

El control de seguimiento de trayectoria genera suavidad en la velocidad angular y permite alinear al robot con la pared, la duración de este movimiento generado por el controlador es τ_2, iniciando y finalizando con una velocidad angular $\omega = 0$.

6.2.6. Arco de círculo (AC)

Este controlador es activado mediante la observación yc_7. El objetivo es mover el robot en un arco de círculo de radio d_d y alinear su heading con la pared siguiente a la esquina. El centro de su movimiento rotacional es una esquina convexa o el punto rp'. Sin embargo, el robot no puede alinearse con el siguiente borde después de la esquina si hay un obstáculo que intersecte el disco de radio d_d durante el movimiento del robot en arco de círculo.

Si el robot no detecta obstáculos que eviten su alineamiento con la siguiente pared a la esquina, entonces la medición del ángulo θ_3 es utilizado para la retroalimentación (Ver Figura 6.7(a)).

Si el robot detecta obstáculos que eviten su alineamiento con la siguiente pared a la esquina (Ver Figura 6.7(b)), entonces la medición del ángulo θ_4 es utilizada como retroalimentación. Si tanto θ_3 como θ_4 son medidos durante el movimiento de arco de círculo, entonces la medición mínima entre θ_3 y θ_4 es utilizada como información de retroalimentación, θ_{AC} toma el valor correspondiente a mín$\{\theta_3, \theta_4\}$. Ver Figuras 6.7(c) y 6.7(d).

El siguiente controlador AC es propuesto, dicho controlador también logra mantener la continuidad en las velocidades lineal y angular:

$$\begin{aligned} V &= k_t \theta_{AC}, \\ \omega &= \omega_n + k_6 e_c + k_7 \frac{de_c}{dt} \end{aligned} \tag{6.6}$$

donde $\omega_n = -V/d_d$ representa una velocidad angular nominal para lograr que el robot se mueva en un arco de círculo de radio d_d y el error está dado por $e_c = d_d - d_{corner}$. Cualquier desviación es corregida por medio de los términos proporcional y derivativo del error e_c, los cuales son pesados por las ganancias de control k_6 y k_7.

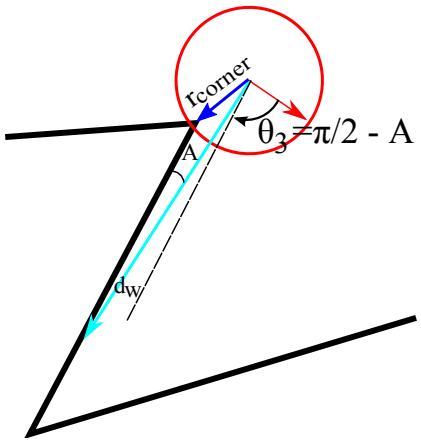
La FSM se encarga que el controlador AC inicie cuando el robot está detenido, en este controlador, una ganancia variable de control es utilizada como sigue:

$$\begin{aligned} k_t &= \frac{V_d}{2\theta_{AC|t=0}}(1 - \cos(\frac{\pi t}{\tau_3})) \text{ if } t < \tau_3 \\ k_t &= \frac{V_{c|t+\tau_3}}{\theta_{AC|t=\tau_3}} \text{ if } t \geq \tau_3 \end{aligned} \tag{6.7}$$

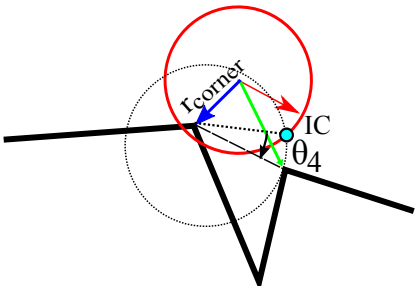
Esta ganancia variable nos permite iniciar el arco de círculo desde una velocidad inicial igual a cero, entonces el robot tiene que aumentar la velocidad hasta alcanzar su máximo valor permitido V_d y finalmente la velocidad lineal regresa a cero cuando el ángulo θ_{AC} es cero. τ_3 es un parámetro que determina el tiempo necesario para alcanzar el valor máximo de la ganancia.

En la Figura 6.7(e), se presenta un caso en el cual es posible medir tanto el ángulo θ_3 como el ángulo θ_4 al mismo tiempo. La Figura 6.7(e) muestra un caso donde $\theta_3 < \theta_4$ y θ_3 es utilizado para la retroalimentación. Subsecuentemente, durante el movimiento

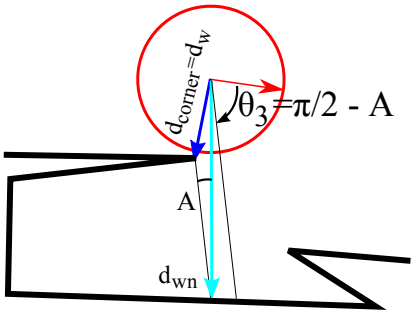
del robot trazando un arco de círculo, puede suceder que $\theta_4 < \theta_3$, lo cual se muestra en la Figura 6.7(f). Consecuentemente, el ángulo para la retroalimentación cambia de θ_3 a θ_4, lo cual puede provocar una discontinuidad en las velocidades del robot. Para resolver este problema, se propone una ganancia de control adaptable k_t para la cual $k_t = \frac{V_{c|t=t_s}}{\text{mín}\{\theta_{3|t=t_s}, \theta_{4|t=t_s}\}}$, donde t_s es el tiempo en que el ángulo mínimo cambia de θ_3 a θ_4 o viceversa de acuerdo al valor mínimo. Aquí $V_{c|t=t_s}$ es la velocidad del robot en el momento que $\text{mín}\{\theta_{3|t=t_s}, \theta_{4|t=t_s}\}$ cambia el ángulo para la retroalimentación de θ_3 a θ_4 o viceversa.



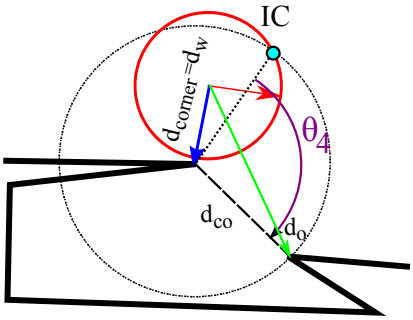
(a) *AC* utilizando θ_3 para la retroalimentación



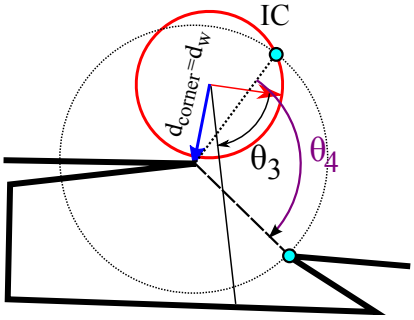
(b) *AC* utilizando θ_4 para la retroalimentación



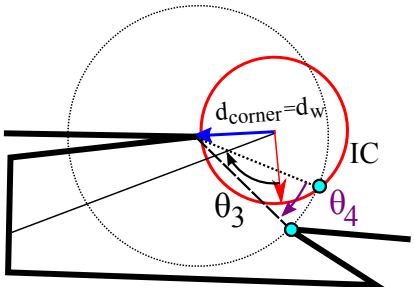
(c) *AC* midiendo θ_3



(d) *AC* midiendo θ_4



(e) *AC* donde el mín$\{\theta_3, \theta_4\}$ es θ_3



(f) *AC* donde el mín$\{\theta_3, \theta_4\}$ es θ_4

Figura 6.7: Primitiva de movimiento de Arco de Círculo (*AC*).

Capítulo 7

Implementación: Simulaciones y Experimentos en un Robot Físico

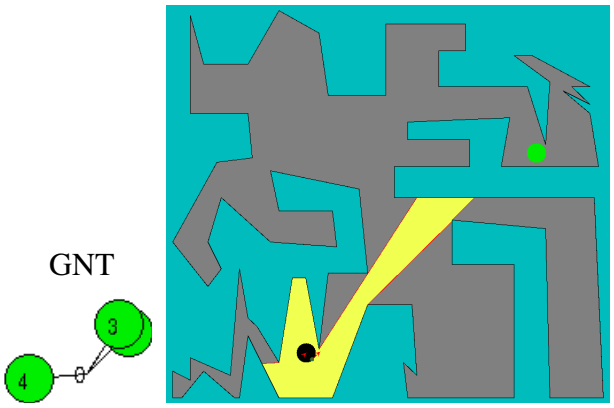
7.1. Simulaciones de la exploración

El método presentado en el Capítulo 6 ha sido implementado y simulado. Las simulaciones fueron ejecutadas en una PC con un procesador de cuatro núcleos Intel Core i7-2670QM a 2.2 GHz, 8 GB de RAM, sistema operativo Linux y fue programado en C++ mediante la librería LEDA. La implementación en software emula exactamente la FSM presentada en la Figura 6.1.

La región explorada del entorno es mostrada en blanco. La región de visibilidad actual del robot es mostrada en amarillo, la región del entorno que no ha sido vista aún es mostrada en gris obscuro. Los obstáculos se muestran en azul. El robot es representado como un disco negro, el sensor omnidireccional es un punto sobre la periferia del robot. Una flecha pequeña sobre el robot es utilizada para mostrar la dirección rt del sensor. El *landmark* es representado por un disco verde. En el GNT, los nodos hoja primitivos son mostrados como cuadrados amarillos, el nodo del *landmark* es un triángulo azul, y los nodos no primitivos son mostrados como círculos verdes.

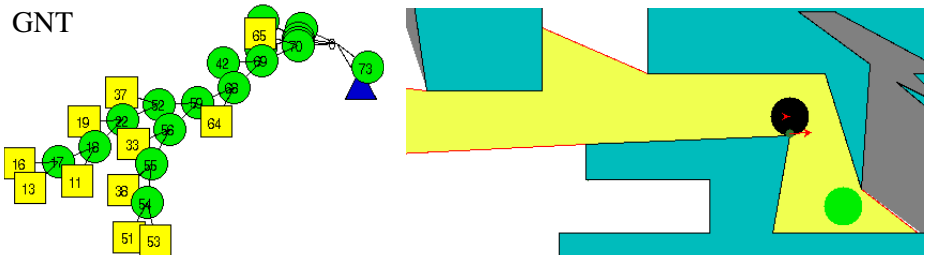
Algunas capturas de imagen de la simulación son presentadas en esta sección. La Figura 7.1 muestra al robot ejecutando un control u_1 que genera una primitiva de movimiento en línea recta, su posición inicial yace en el interior de E, este se mueve hacia adelante hasta que se detecta un contacto con ∂E, este es el único caso en el que el robot no está siguiendo la frontera del entorno. Después el robot sigue ∂E en sentido contrario a las manecillas del reloj en todo momento hasta que la exploración es terminada. La Figura 7.2 muestra

que el *landmark* es totalmente visible desde la ubicación del sensor omnidireccional, por lo tanto el *landmark* es codificado como un nodo hijo de la raíz del GNT.



(a) (b)

Figura 7.1: Ejecución de la primitiva de movimiento de línea recta. Se muestra el GNT correspondiente.

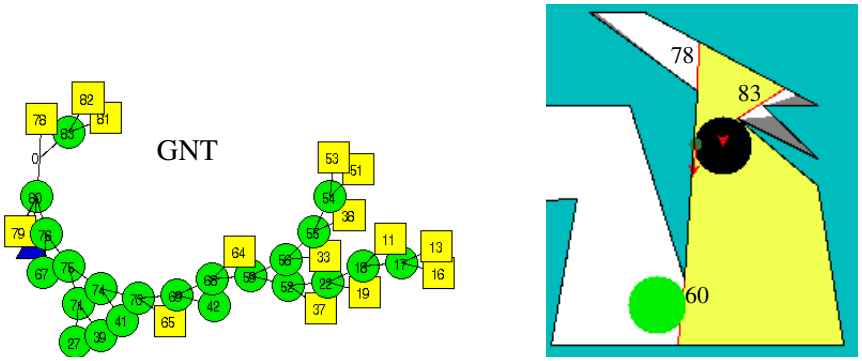


(a) (b)

Figura 7.2: El *landmark* es totalmente visible desde la ubicación del sensor omnidireccional, por lo tanto es codificado como un nodo hijo de la raíz del GNT.

La Figura 7.3 muestra el GNT al final de la ejecución del algoritmo *Local Exploration*, los gaps 78 y 83 reciben la etiqueta de primitivo, y el nodo 83 lo propaga a toda su descendencia (los nodos hoja 81 y 82).

En la Figura 7.4 se muestra el GNT completo en el momento que el robot ha finalizado la exploración del entorno. Nótese que el robot ha finalizado la exploración del entorno antes de viajar por completo por la frontera del entorno.



(a) (b)

Figura 7.3: El GNT al final de la ejecución del algoritmo *Local Exploration*.

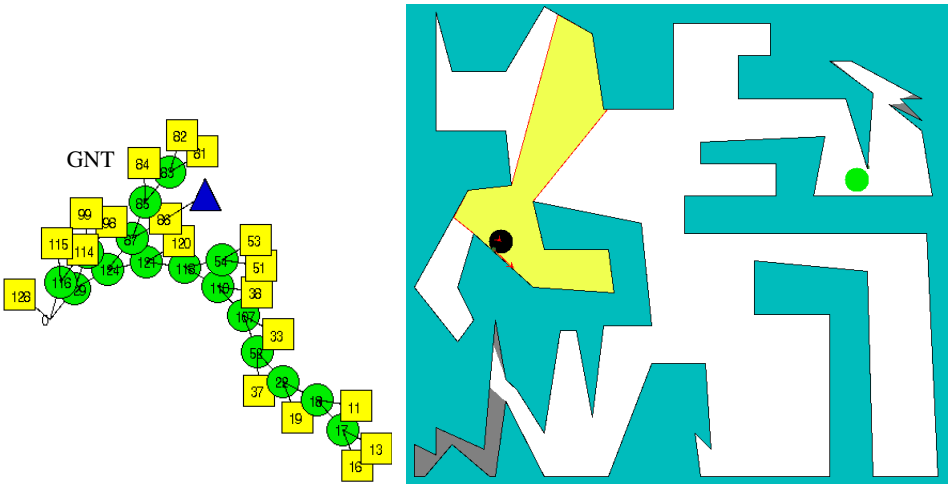
7.2. Experimentos en el robot real

En todos los experimentos, se utilizó un robot Pioneer P3-DX, este es un sistema de manejo diferencial. El robot está modelado como un disco de radio 0.2 m, este tiene una velocidad máxima de traslación de 1.2 m/s y una velocidad máxima de rotación de 5.236 rad/s. Para los experimentos, la velocidad lineal máxima deseada V_d, fue puesta a 0.33 m/s. La distancia deseada d_d entre el centro del robot y la frontera del entorno es de 0.4 m. Así, la distancia entre la frontera del robot y el entorno se controla para ser de 0.2 m.

En la implementación, todos los algoritmos se ejecutan directamente en la computadora del robot, la cual tiene un procesador Pentium M a 1.8 Ghz con 1 GB de RAM. El sistema operativo es Linux y se utiliza algunas funciones de ROS, el ciclo de control se ejecuta a 12.5 Hz. El software es programado en C++.

El sensor omnidireccional fue implementado usando 2 sensores láser Hokuyo modelo URG-04LX, los cuales fueron posicionados sobre el robot y en direcciones opuestas, ver Figura 8.12. La implementación del detector de gaps es simple, este utiliza directamente los datos obtenidos de los sensores láser para detectar 2 mediciones láser consecutivas en orden angular con una diferencia en distancia más grande que un umbral determinado, el detector de gaps (brechas) fue codificado para probar el método completo, tanto en el autómata como en los controladores en el robot real.

Los experimentos fueron realizados en dos diferentes entornos, ver Figuras 7.6(a) y 7.6(b). El primer entorno está construido mediante una estructura de tablaroca dentro del laboratorio de robótica del CIMAT, dicho entorno está compuesto por 6 segmentos de línea



(a) (b)

Figura 7.4: GNT completo.

recta, 8 esquinas cóncavas y 2 esquinas convexas (ver Figura 7.6(a)). El segundo entorno es una oficina del CIMAT con muebles comunes (ver Figura 7.6(b)).

El principal objetivo del primer conjunto de experimentos en el laboratorio de robótica del CIMAT es mostrar la evolución del GNT, un aspecto que es relevante en este experimento es que este muestra que el robot no necesita viajar por toda la frontera del entorno para finalizar la exploración, tan pronto como todos los nodos hoja en el GNT se vuelven primitivos la exploración es finalizada. En efecto, como se menciona en la Sección 6.1, el principal objetivo que tiene el GNT en este enfoque es indicar que la tarea de exploración ha finalizado, esto se realiza sin la necesidad de localizar al robot en el entorno. La Figura 7.7 muestra la evolución del GNT durante la tarea de exploración y la posición correspondiente del robot en el entorno.

La Figura 7.8 muestra las velocidades lineal y angular del robot y la distancia a la pared, mientras el robot seguía la frontera del entorno hasta que la condición de paro fue dada por el GNT. Nótese que las velocidades del robot son continuas a pesar de la conmutación entre los controladores.

El segundo conjunto de experimentos (ver Figura 7.6(b)) tiene por objetivo probar los controladores basados en retroalimentación y la capacidad para seguir la pared en un entorno típico de interiores en el CIMAT. Las estadísticas resultantes se muestran en la Tabla 7.1. Estas estadísticas muestran el desempeño del robot al seguir la frontera del entorno durante 3 vueltas utilizando la estrategia propuesta. Para cada vuelta que el robot

Figura 7.5: Robot y sensores láser.

(a) Laboratorio

(b) Oficina

Figura 7.6: Entornos utilizados para los experimentos.

ejecuta se presenta la media de la distancia entre el centro del robot y la frontera del entorno, la correspondiente desviación estándar, los valores máximos y mínimos de esta distancia, así como el tiempo por vuelta que le tomo al robot viajar por la frontera del entorno. En promedio, el robot siguió la frontera del entorno a una distancia de 0.36 m, lo cual implica que hay un error de 4 cm con respecto a la distancia deseada que fue fijada en $d_d = 0.4$ m. La máxima medición de error durante el movimiento completo fue de 13 cm. En promedio el tiempo para completar una vuelta fue de 59.61 s y el tiempo total para viajar las 3 vueltas fue de 178.84 s para el perímetro del entorno que es de 16.7 m.

La Figura 7.9 muestra un experimento en una oficina del CIMAT. En la Figura 7.9(a) se muestra la ejecución de una rotación en sitio en una esquina cóncava, el controlador activado es RP. La Figura 7.9(b) muestra la ejecución de un arco de círculo alrededor de una esquina convexa, el controlador activado es AC.

(a) Robot iniciando la exploración y su correspondiente GNT inicial

(b) Un evento crítico de división de un gap

(c) Un nuevo gap aparece como un gap primitivo

(d) La exploración termina, solo hay un gap en el GNT el cual es primitivo

Figura 7.7: Evolución del GNT durante la exploración.

Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.364	0.034	0.447	0.291	59.04
2	0.359	0.038	0.441	0.293	58.368
3	0.358	0.041	0.431	0.27	61.44

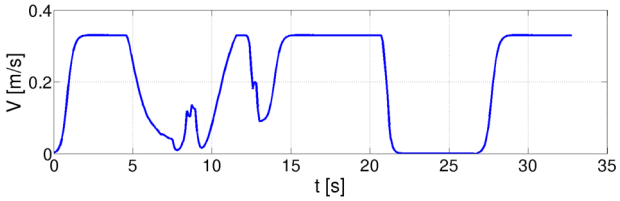
Tabla 7.1: Estadísticas de la distancia más corta medida entre el centro del robot y la frontera del entorno: una oficina del CIMAT

Las Figuras 7.10, 7.11 y 7.12 muestran las velocidades lineal y angular del robot durante el transcurso del viaje en la oficina. La Figura 7.10(a) muestran la velocidad lineal del robot mientras se ejecuta el controlador SLW. En la Figura 7.10(b) se muestra la velocidad lineal que es generada durante la ejecución del controlador $SLWD$ cuando el robot se aproxima a una esquina y la Figura 7.10(c) muestra la velocidad lineal generada por el mismo controlador cuando el robot se aproxima a una pared.

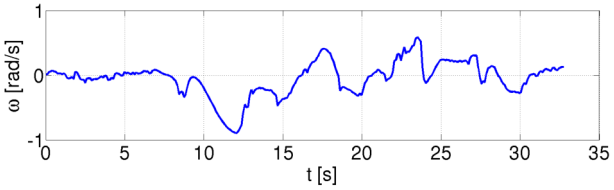
La Figura 7.11 muestra la velocidad angular del robot mientras este está rotando en sitio en una esquina cóncava.

La Figura 7.12(a) muestra la velocidad angular generada por el controlador AC durante la ejecución de la rotación en arco de círculo alrededor de la esquina y en la Figura 7.12(b) se muestra la velocidad lineal generada con el mismo controlador.

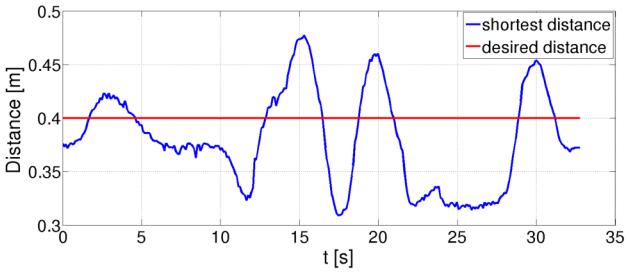
Los experimentos permiten verificar que las velocidades del robot no presentan discontinuidades (ver Figura 7.8), en general la velocidad angular es más ruidosa que la velocidad



(a) Velocidad lineal del robot



(b) Velocidad angular del robot



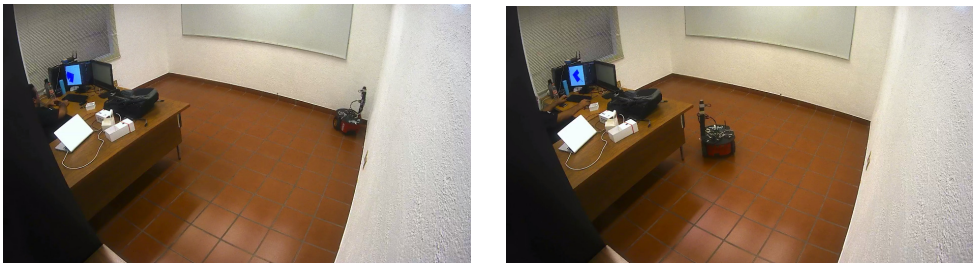
(c) Distancia a la pared

Figura 7.8: Velocidades lineal y angular del robot y la distancia a la pared.

lineal, esto es debido a errores en las lecturas del sensor que hacen variar la dirección al obstáculo más cercano. La siguiente liga invoca un video que muestra una simulación y dos experimentos en el robot real. El primer experimento fue realizado en el laboratorio de robótica del CIMAT y el segundo en una oficina del CIMAT.

Click aquí para enlace a video

Con la finalidad de probar la pertinencia del autómata, se implementó un esquema de control alternativo en el cual la activación de cada controlador es solamente dependiente de la observación, sin considerar el estado en la FSM. Se observó que los controladores en uso cambian frecuentemente generando discontinuidades y ruido en las velocidades del robot. Aunque esta implementación fue capaz de cumplir con la tarea, para explorar los



(a) Rotación en sitio en una esquina cóncava

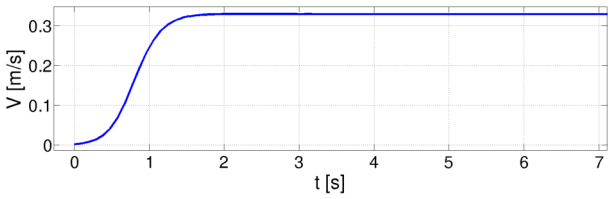
(b) Arco de círculo alrededor de una esquina convexa

Figura 7.9: Experimentos en la oficina.

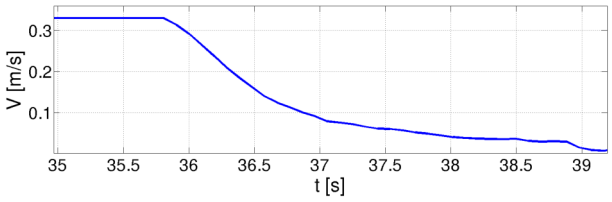
mismos entornos el robot tardó 10 veces mas tiempo que cuando se utiliza la FSM. Por lo tanto, se enfatiza que el autómata restringe las posibles transiciones de estado filtrando observaciones espurias que se deben al ruido en las lecturas del sensor.

Algunas limitaciones de la actual implementación son las siguientes: Si la velocidad lineal máxima deseada es cambiada, entonces las ganancias de los controladores deben modificarse en consecuencia de ello. Una posibilidad para solucionar esta limitación es usar ganancias adaptables que dependan de la máxima velocidad lineal utilizada. Los gaps (brechas) espurios algunas veces aparecen instantáneamente, es posible resolver este problema mediante el filtrado de los datos. Sin embargo, este problema nunca ha impedido que el robot finalice la exploración. No obstante, para tratar con entornos mas complejos, debe mejorarse la implementación del detector de gaps ya sea usando técnicas de filtrado sobre los datos sin procesar o mediante el uso de un método robusto de ajuste de líneas para detectar esquinas convexas, haciendo que el detector de gaps sea mas robusto.

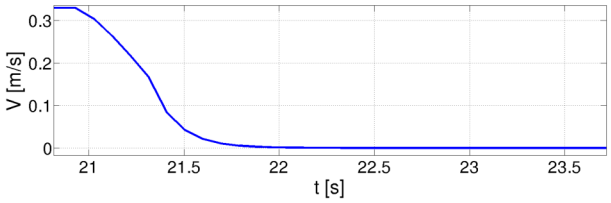
Con base en los resultados experimentales, se concluye que el modelo teórico presentado en los primeros capítulos puede ser adaptado para tratar con lecturas láser reales e imperfectas y la ejecución imperfecta de las primitivas de movimiento. La implementación actual funciona apropiadamente para los entornos probados.



(a) Velocidad lineal, controlador SLW



(b) Velocidad lineal, controlador $SLWD$, aproximándose a esquina



(c) Velocidad lineal, controlador $SLWD$, aproximándose a obstáculo

Figura 7.10: Velocidad lineal: Controladores SLW y $SLWD$.

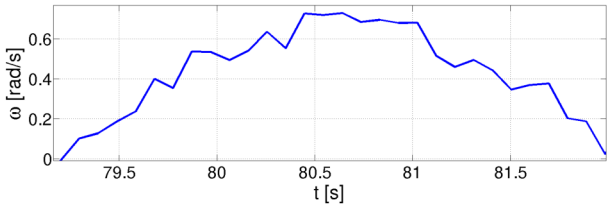
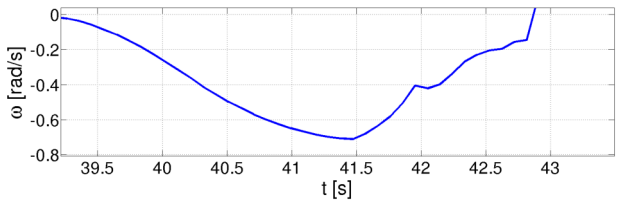
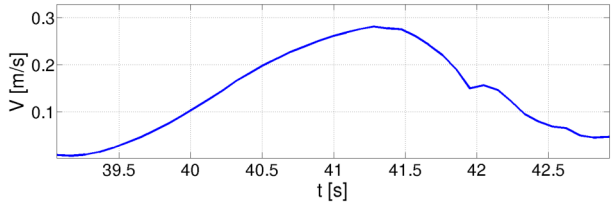


Figura 7.11: Velocidad Angular: Controlador RP.



(a) Velocidad Angular



(b) Velocidad Lineal

Figura 7.12: Velocidades angular y lineal: Controlador AC.

Capítulo 8

Un Autómata y un Control de Modo Deslizante Super-Twisting para el Seguimiento de Pared

8.1. Introducción

A continuación se propone un método de exploración utilizando un autómata que gestiona las observaciones del robot y elige los valores de referencia de acuerdo al estado en el autómata. Para ello, se integra un control de modos deslizantes llamado *Super-Twisting* el cual alcanza rápidamente los valores de referencia, esto es debido a que este tipo de control tiene la propiedad de converger en tiempo finito [97], [98] y [99]. Este enfoque permite que el robot pueda moverse continuamente sin necesidad de detenerse cuando se presentan obstáculos en la trayectoria del robot. Así, es posible realizar el seguimiento de pared de manera rápida y robusta. Este seguidor de pared puede utilizarse para tareas de exploración de entornos desconocidos (como el presentado en el Capítulo 6) o para encontrar objetos en el entorno.

Para modelar las trayectorias del robot, se utilizan dos discos. Un disco está centrado en el centro del robot, este es utilizado como un robot virtual circular para seguir la frontera del entorno estando en contacto o muy cerca de dicha frontera. Existe otro disco más grande, el cual es utilizado para seguir las trayectorias de arco de círculo sobre la frontera de dicho círculo. El robot ejecuta 3 primitivas de movimiento de acuerdo a la estructura de la frontera del entorno. Si el robot está siguiendo un segmento de línea entonces, ejecuta un movimiento en línea recta. Si el robot se mueve alrededor de una esquina convexa, este ejecuta un movimiento de rotación alrededor de la esquina en sentido de las manecillas del

reloj. Si el robot encuentra una esquina cóncava, entonces el robot busca ejecutar un arco de círculo en sentido contrario a las manecillas del reloj, ver Figura 8.1, donde el robot real es representado por el disco negro.

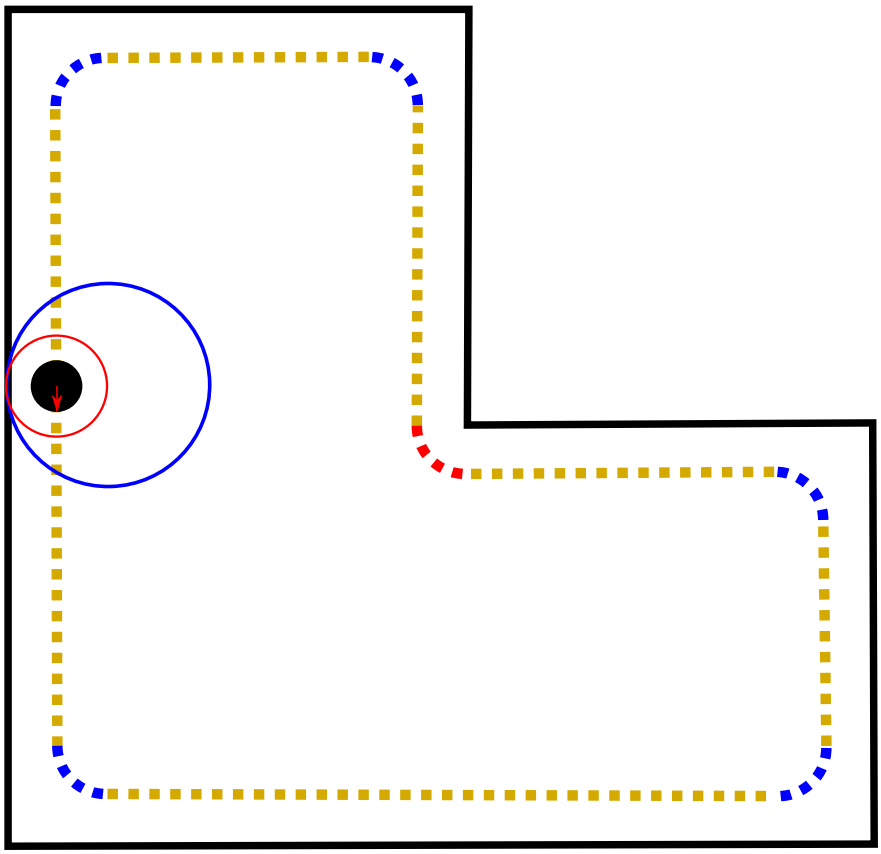


Figura 8.1: Trayectoria del robot al seguir la frontera del entorno

En este trabajo, se realiza el control de un robot de manejo diferencial para realizar la tarea de navegación mediante el seguimiento de pared y está relacionado al problema de planificación de trayectorias para que el robot evite colisiones con los obstáculos como en los trabajos [102], [103], y particularmente con robots noholonómicos como el enfoque de [104] y en el trabajo presentado en los Capítulos 5 y 6 de esta tesis. En [104], proponen un esquema para evitar colisiones con obstáculos durante la navegación visual con un robot móvil con ruedas. La navegación visual consiste en seguir una trayectoria, representada mediante un conjunto ordenado de imágenes clave. La navegación visual se realiza usando control basado en imagen, para evitar colisiones los obstáculos son sensados mediante un sensor *lidar* a bordo del robot. En [105], los autores proponen un enfoque que integra la planificación y el control basado en imagen para seguir la carretera y evitar obstáculos en movimiento.

8.1 Introducción

Uno de los objetivos principales de este Capítulo es representar una estrategia en forma de una máquina de estados finitos (autómata). El presente trabajo se relaciona con el de [105] ya que este último también usa un autómata. Sin embargo, en [105] se utiliza control visual para dirigir al robot, mientras que en este trabajo el robot es manejado mediante un controlador de modos deslizantes llamado *Super-Twisting*. Además, las tareas principales difieren, en este trabajo, la tarea es seguir la pared sin colisionar con esta, mientras que en [105] el objetivo es mantener al robot dentro del carril de carretera y evitar obstáculos en movimiento. En los Capítulos 5 y 6 se presenta un trabajo enfocado en la exploración de entornos desconocidos, planos, poligonales y simplemente conectados, se muestran las condiciones teóricas que garantizan que el robot descubrirá la mayor región posible del entorno. La estrategia de exploración está basada en la navegación mediante seguimiento de pared. El autómata propuesto, filtra observaciones espurias para activar los controladores basados en retroalimentación. El esquema de control conmuta entre los controladores de acuerdo a las observaciones obtenidas por el sensor del robot. El método de navegación presentado en este Capítulo, es una mejora del esquema de navegación presentado en los Capítulos anteriores. En los Capítulos 5 y 6, cada vez que el robot encuentra una esquina convexa o cóncava, el robot debe detenerse, en contraste con el trabajo presentado en este Capítulo, el robot se mueve continuamente independientemente de si este encuentran o no esquinas durante el seguimiento de pared.

8.1.1. Principales contribuciones de este capítulo

A continuación resaltamos las principales diferencias entre este método de navegación y el método presentado en los Capítulos 5 y 6. Las principales diferencias son:

1. En este método el robot es operado mediante un controlador de modos deslizantes llamado *Super-Twisting*, mientras que en el método presentado en el Capítulo 6 se utilizan controladores PD.

2. En el Capítulo 6, se utilizan dos diferentes controladores para regular las 2 velocidades lineal y angular del robot en cada estado del autómata, en contraste con el método presentado en este Capítulo, se utilizan los mismos controladores para regular las velocidades lineal y angular en todos los estados del autómata y solamente los valores de referencia (*set points*) de control son los que cambian.

3. El tiempo de navegación para recorrer la frontera del entorno es reducido significativamente en comparación con el método presentado en el Capítulo 6.

8.1.2. Planteamiento del Problema

Realizar la tarea de seguimiento de la frontera del entorno a velocidad constante aún cuando se presenten obstáculos en la trayectoria del robot. Para ello se utiliza un robot de manejo diferencial equipado con un sensor láser de rango para detectar las características de un entorno poligonal y evitar colisiones con los obstáculos.

8.2. Autómata

Se propone una nueva máquina de estados finitos (FSM), la Figura 8.2 muestra la representación de dicho autómata. Este autómata tiene 3 estados: SL (línea recta), CWT (giro en sentido de las manecillas del reloj), $CCWT$ (giro en sentido contrario a las manecillas del reloj), estos estados corresponden a las primitivas de movimiento que ejecuta el robot, suponiendo que la frontera del entorno se encuentra a la derecha del *heading* del robot. El autómata está en el estado SL cuando el robot está siguiendo un segmento de línea, el autómata está en el estado $CCWT$ si el robot está ejecutando un giro en sentido contrario a las manecillas del reloj, esto pasa cuando el robot está en una esquina cóncava. El autómata está en el estado CWT cuando el robot está ejecutando un giro en sentido de las manecillas del reloj alrededor de una esquina convexa.

Las medidas obtenidas del sensor láser de rango se utilizan de dos maneras: 1) Se utilizan como información de retroalimentación para el controlador de velocidad angular (ver Secciones 8.2.5 y 8.3). 2) Son utilizadas como respuestas a preguntas binarias, las cuales permiten realizar o no el cambio de un estado a otro, estas decisiones elementales son llamadas "observaciones". Las observaciones son etiquetadas mediante y_i en la Figura 8.2. Estas se describen a detalle en la Sección 8.2.4.

8.2.1. Sensor del Robot y Discos

El robot de manejo diferencial tiene su *heading* en la parte frontal y se modela como un disco de radio r. El punto en el extremo derecho es llamado rp. El robot tiene un sensor láser omnidireccional, el cual es utilizado para medir distancias y ángulos en el entorno. El sensor se ubica en el punto rp.

Se utilizan dos discos para modelar las trayectorias del robot y ayudar a definir las transiciones entre los estados del autómata. De acuerdo a la Figura 8.3, se define un disco centrado en el centro del robot con un radio $d_d > r$, el punto en el extremo derecho de este

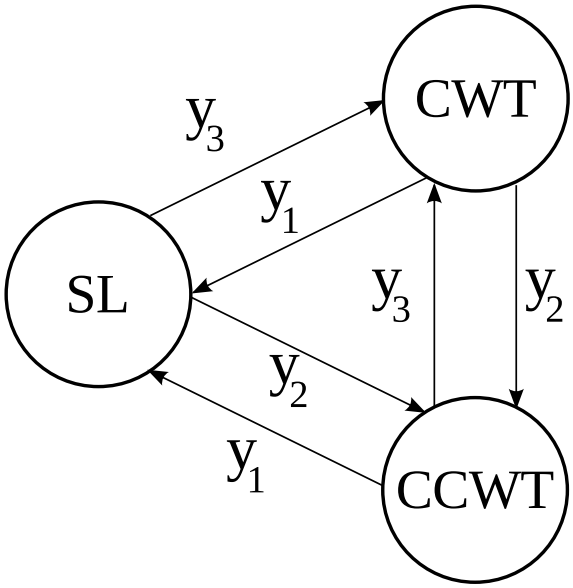


Figura 8.2: Autómata que representa la estrategia de navegación.

disco es llamado rp'. Este disco de radio d_d es utilizado como un robot circular virtual para seguir la frontera del entorno en contacto con dicha frontera. Otro disco más grande es utilizado para moverse en trayectorias correspondientes a arcos de círculo sobre la frontera de dicho disco. Este segundo disco tiene un radio d_t (donde $d_t > d_d$), el cual está centrado en otro punto diferente al centro del robot, pero comparte el punto rp' con el disco de radio d_d. El segmento de línea que pasa sobre los puntos rp y rp' es llamado línea $rp - rp'$ (Ver Figura 8.3).

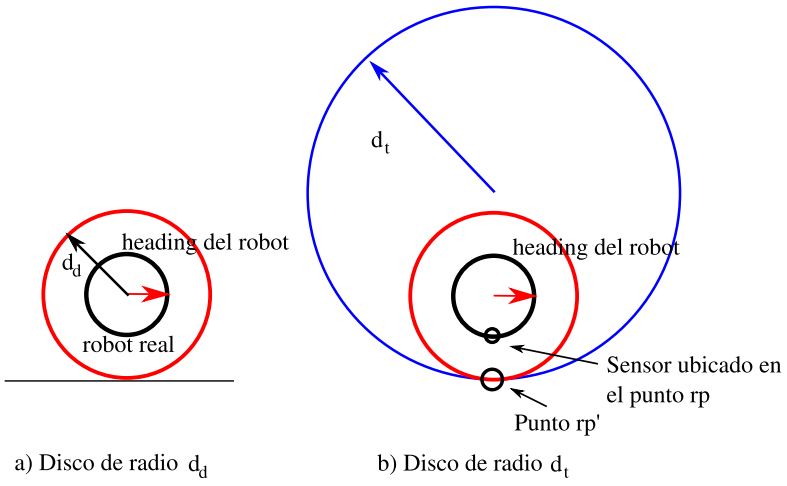


Figura 8.3: Dos discos

8.2.2. Detección de características

Para detectar las características del entorno (esquinas que delimitan las paredes), se utiliza una técnica simple, similar a la propuesta en el Apéndice A, de ajuste de líneas local para encontrar esquinas convexas y esquinas cóncavas. Primero, se detecta el punto más cercano desde el sensor láser. Segundo, se miden los ángulos entre el rayo que va desde el sensor láser al punto más cercano y los rayos entre el punto más cercano y los siguientes 10 rayos sensados (en sentido contrario de las manecillas del reloj). Estos 10 valores de ángulos son promediados y el ángulo resultante es llamado *ángulo de referencia*. Tercero, se mide el ángulo entre el rayo que va desde el sensor láser y el punto más cercano en la frontera de los obstáculos y el rayo entre el punto más cercano y un punto sensado determinado, este es llamado *ángulo del punto*. Si el *ángulo del punto* es más pequeño que el *ángulo de referencia* más una tolerancia, entonces se ha detectado una esquina cóncava. Analogamente, si el *ángulo del punto* es más grande que el *ángulo de referencia* más una tolerancia, entonces se ha detectado una esquina convexa. Para entornos más complejos, existen otros algoritmos de ajuste de líneas conocidos a partir de puntos, [10] y [101].

8.2.3. Medidas del Sensor: Ángulos y Distancias

Refiérase a la Figura 8.4. El rayo que apunta al punto más cercano del obstáculo sobre el segmento de línea que el robot está siguiendo, es llamado r_{min}. θ_1 se define como el ángulo medido desde la línea $rp - rp'$ al rayo r_{min} en sentido contrario de las manecillas del reloj. d_1 es la distancia más pequeña medida desde el centro del robot hasta el segmento de línea que el robot está siguiendo. El ángulo θ_1 y la distancia d_1 son usadas como información de retroalimentación en el estado SL (ver Sección 8.2.5).

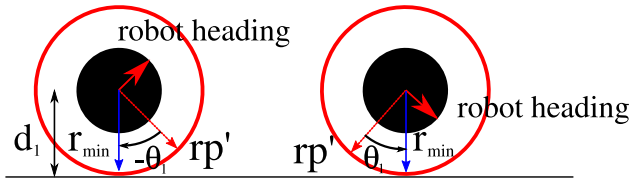


Figura 8.4: Ángulo θ_1 y rayo r_{min}

Refiérase a la Figura 8.5. θ_2 se define como el ángulo entre la línea $rp - rp'$ y el rayo r_{min} apuntando al punto más cercano en el último segmento poligonal, en sentido contrario a las manecillas del reloj de la frontera de la región del obstáculo que el círculo de radio d_t intersecta.

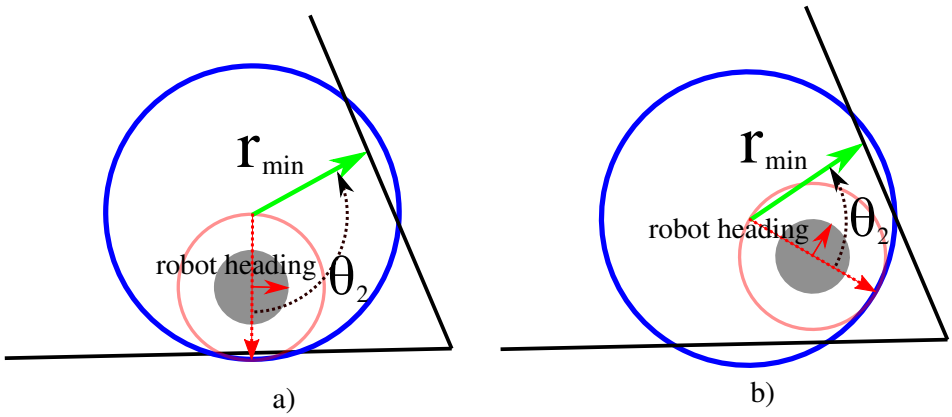


Figura 8.5: Ángulo θ_2

Refiérase a la Figura 8.6. La distancia h está definida como la distancia entre el centro del disco de radio $d_t = a + d_d$ y la esquina cóncava. En el momento que se detecta un bicontacto entre el disco de radio d_t y el borde del obstáculo, la distancia entre el centro del robot y la esquina cóncava (distancia d_{corner}) es medida. Usando las distancias d_t y d_{corner} y el ángulo γ (el cual puede ser medido con el láser), y aplicando la ley de cosenos, se calcula la distancia h. La primera vez que h es calculada, es llamada h_0. Después, durante el movimiento del robot, h es calculada en cada iteración del ciclo de control. El ángulo θ_2 y la distancia h son utilizadas como información de retroalimentación en el estado $CCWT$ (Ver Sección 8.2.5).

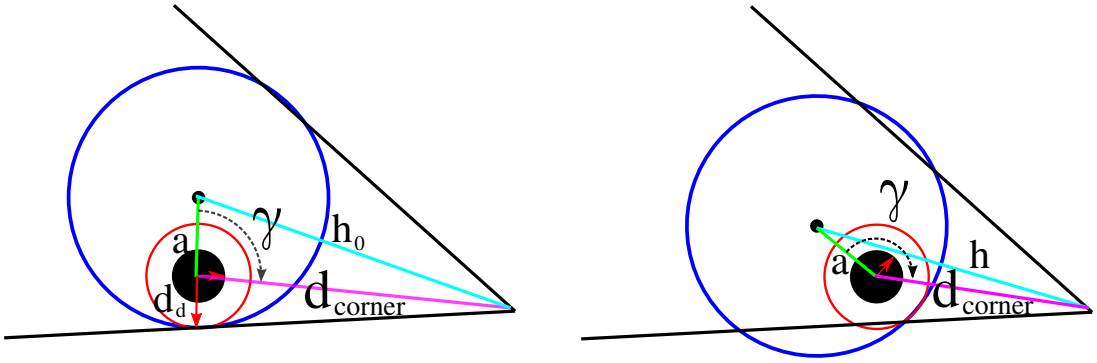


Figura 8.6: Distancia h

Refiérase a la Figura 8.7. El rayo que va desde el centro del robot y apunta a la esquina convexa es llamado r_{corner}. Se utilizan dos distancias para calcular el ángulo θ_3, la distancia a la esquina d_{corner} y la distancia a un obstáculo en dirección de la línea $rp - rp'$, esta segunda distancia es llamada d_w. Utilizando la ley de cosenos, es posible calcular el

ángulo auxiliar llamado A, así $\theta_3 = \frac{\pi}{2} - A$. Nótese que la línea $rp - rp'$ y el rayo r_{corner} no son necesariamente colineales, algunas veces el robot está alineando su *heading* con algún segmento de línea virtual (denotado por s_v), ver Figura 8.7 a) y b). Sin embargo, como la línea que pasa sobre el punto rp y rp' progresivamente se vuelve perpendicular al segmento de línea s_v, el segmento correcto será sensado y considerado, ver Figura 8.7 c). El movimiento finaliza cuando el ángulo θ_3 es más pequeño que una tolerancia ϵ_2, esto es equivalente a tener el *heading* del robot alineado con el segmento posterior a la esquina, en sentido contrario de las manecillas del reloj, ver Figura 8.7 d).

Si existen puntos de los obstáculos dentro del disco de radio d_t, entonces se calcula el ángulo auxiliar θ_4 para determinar el ángulo que el robot debe rotar para evitar colisionar con los obstáculos. Un punto del obstáculo es un punto que no pertenece a los segmentos que compartidos por la esquina convexa.

Refiérase a la Figura 8.8. Para calcular el ángulo θ_4, se utilizan las distancias d_{cc} y d_{cdo}; d_{cc} es la distancia desde el centro del disco de radio d_t a la esquina convexa. d_{cdo} es la distancia que va desde el centro del disco de radio d_t al punto del obstáculo más cercano. Se utiliza la ley de cosenos para calcular la distancia entre la esquina convexa y el punto a distancia d_{cdo} desde el centro del disco de radio d_t. Esta distancia es llamada d_{co}. Para encontrar el punto que colisiona con el obstáculo más cercano (desde el centro del disco de radio d_t), se utiliza un círculo centrado en la esquina convexa de radio d_{co}. El círculo centrado en la esquina convexa y el círculo de radio d_t se intersectan. El punto de intersección más próximo al obstáculo más cercano (desde el centro del disco de radio d_t) es llamado punto IC. θ_4 es el ángulo entre el rayo que va desde la esquina al punto IC y el rayo que va desde la equina al punto del obstáculo más cercano al centro del disco de radio d_t. La Figura 8.8 a) muestra el caso cuando el obstáculo es una esquina y la Figura 8.8 b) muestra cuando el obstáculo es un segmento. Finalmente, el ángulo que el robot debe rotar al rededor de la esquina convexa es $min\{\theta_3, \theta_4\}$. El ángulo θ_4 es calculado en cada iteración del método, notese que es posible decidir cual de los ángulos θ_3 o θ_4 es más pequeño, en cada instancia de tiempo. Los ángulos θ_3 o θ_4 y la distancia d_{corner} son utilizados como información de retroalimentación en el estado CWT (ver Sección 8.2.5).

8.2.4. Observaciones y Bits de las Observaciones

En este trabajo, las operaciones "and" del álgebra booleana dadas entre varias respuestas binarias (valores de bits) disparan los cambios entre estados en el autómata. Dichas operaciones lógicas "and" entre diferentes bits son llamadas observaciones ya que son es-

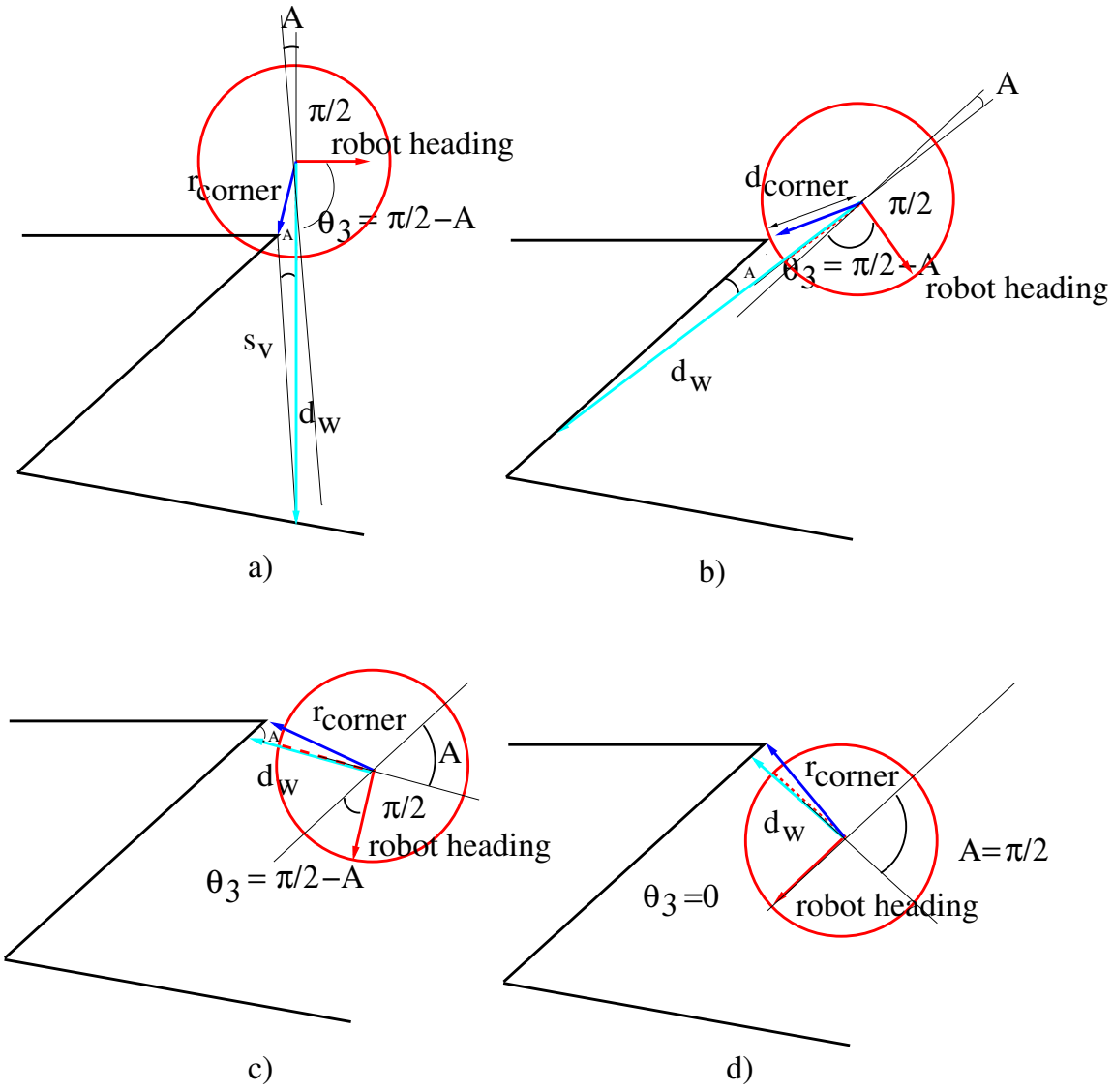


Figura 8.7: Ángulo θ_3

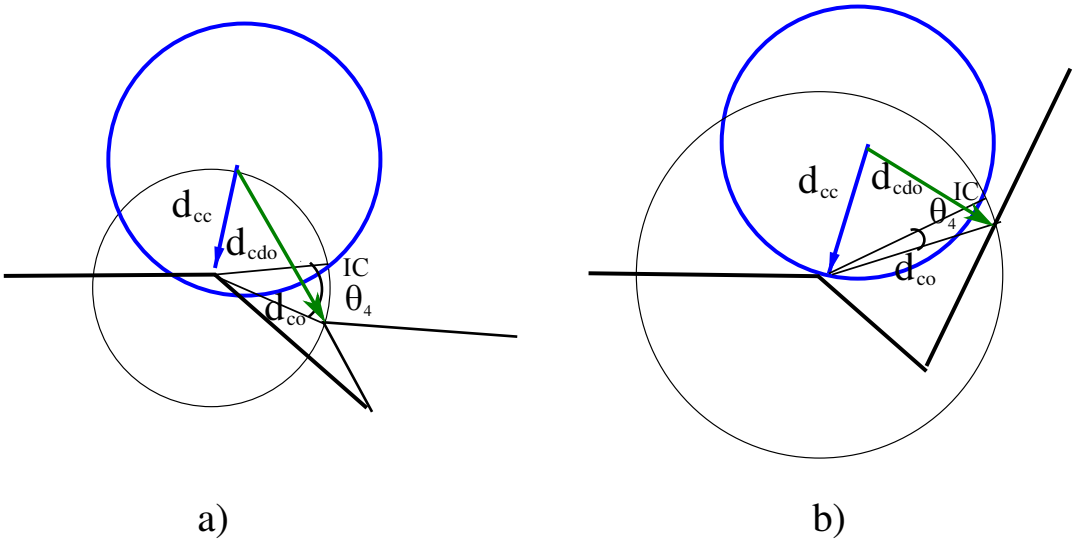


Figura 8.8: Ángulo θ_4

tablecidas directamente desde las mediciones del sensor láser de rango. Los bits de estas observaciones son respuestas "si" o "no" a preguntas binarias.

Los 4 bits que definen la una observación son los siguientes:

- rp': Si hay un punto del obstáculo (lectura del sensor) más cercano a rp' que una tolerancia ϵ_1, entonces este bit es verdadero, de otro modo este será falso. Este bit es relevante únicamente para transitar al estado CWT.
- bc: si hay dos o más diferentes sectores circulares del disco de radio d_t intersectando la región del obstáculo, entonces este bit es verdadero de otro modo este será falso. Este bit es relevante para transitar a los estados SL y $CCWT$.
- $rp' - e$: Si el punto rp' se encuentra más cercano a una esquina convexa que una tolerancia ϵ_1 entonces este bit es verdadero, de otra manera será falso. Este bit es relevante para transitar al estado CWT.
- $aligned$: Si $|\theta_1| < \epsilon_2$ o $|\theta_2| < \epsilon_2$ o $|\theta_3| < \epsilon_2$ o $|\theta_4| < \epsilon_2$, entonces este bit es verdadero, si $|\theta_1| > \epsilon_2$ o $|\theta_2| > \epsilon_2$ o $|\theta_3| > \epsilon_2$ o $|\theta_4| > \epsilon_2$ entonces será falso. Este bit es relevante para transitar a los 3 estados SL, $CCWT$ and CWT.

Las 3 observaciones que establecen la transición entre estados en el autómata son presentados en la Tabla 8.1. En esta tabla X puede tomar cualquiera de los dos valores binarios.

$y_i =$	$(rp'$,	bc,	$rp' - e$,	$aligned)$	STATE
$y_1 =$	(X,	0,	X,	1)	SL
$y_2 =$	(X,	1,	X,	0)	CCWT
$y_3 =$	(1	X	1,	0)	CWT

Tabla 8.1: Observaciones y_i

8.2.5. Valores de Referencia del Control Según el Estado del Autómata

La ley de control se encarga de llevar la orientación del robot desde un valor inicial θ_i a un valor final θ_f. En este trabajo, los valores de referencia de control correspondientes a la orientación final θ_f cambian de acuerdo al estado en el autómata. También se tiene un error que está relacionado a la distancia deseada, los valores de referencia correspondientes a la distancia también cambian según el estado del autómata. A continuación se especifica tanto los valores de referencia de control para el ángulo como los de la distancia.

Viajando en Línea Recta SL

En este estado $\theta_f = \theta_1 = 0$ ya que se quiere que el robot esté alineado con el segmento de línea que está siguiendo.

Con respecto al valor de referencia de control que se relacionan con la distancia, en este estado se quiere que $d_1 = d_d$.

Giro en Sentido Contrario a las Manecillas del Reloj CCWT

En este estado, se quiere que el robot termine el giro en sentido contrario a las manecillas del reloj $CCWT$ para alcanzar la alineación con el último segmento poligonal en sentido contrario de las manecillas del reloj (después de la esquina cóncava) de la región del obstáculo, así: $\theta_f = \theta_2 = 0$. El valor de referencia de control relacionado con la distancia es: $h = h_0$.

Giro en Sentido de las Manecillas del Reloj

En este estado, la tarea del robot es finalizar su giro en sentido de las manecillas del reloj CWT alrededor de una esquina convexa para alcanzar la alineación del robot con el segmento posterior (en sentido de las manecillas del reloj) a la esquina, así: $\theta_f =$

mín$\{\theta_3, \theta_4\} = 0$. En este estado, se quiere que $d_{corner} = d_d$.

La Tabla 8.9 resume los bits importantes en las observaciones que hacen transitar de un estado a otro en el autómata, así como los valores de referencia de control para cada estado.

Transición entre Estados	Valores de referencia del control	Bits principales del estado inicial	Bits principales del estado final
SL → CCWT	$\theta_1 = 0$ $d_1 = d_d$	$bc = 0$ $aligned = 1$	$bc = 1$ $aligned = 0$
SL → CWT	$\theta_1 = 0$ $d_1 = d_d$	$bc = 0$ $aligned = 1$	$rp' = 1$, $rp' - e = 1$ $aligned = 0$
CCWT → SL	$\theta_2 = 0$ $h = h_0$	$bc = 1$ $aligned = 0$	$bc = 0$ $aligned = 1$
CCWT → CWT	$\theta_2 = 0$ $h = h_0$	$bc = 1$ $aligned = 0$	$rp' = 1$, $rp' - e = 1$, $aligned = 0$
CWT → SL	mín$\{\theta_3, \theta_4\} = 0$ $d_{corner} = d_d$	$rp' = 1$, $rp' - e = 1$ $aligned = 0$	$bc = 0$ $aligned = 1$
CWT → CCWT	mín$\{\theta_3, \theta_4\} = 0$ $d_{corner} = d_d$	$rp' = 1$, $rp' - e = 1$ $aligned = 0$	$bc = 1$ $aligned = 0$

Figura 8.9: Transiciones entre estados, bits que determinan la transición y los valores de referencia de control para cada estado.

8.3. Esquema General del Control

En esta sección, se describe el esquema de control propuesto que proporciona las velocidades lineal y angular, llamadas v y ω respectivamente.

8.3.1. Control de Velocidad Lineal

La velocidad lineal del robot se define a continuación:

$$v = \frac{v_n}{2}(1 + \tanh(\alpha(t - \beta))) \tag{8.1}$$

donde α es un factor de escala y β es un desplazamiento en el tiempo. Estos parámetros están relacionados con la duración de la transición desde que el robot está en reposo hasta que alcanza la velocidad nominal v_n. Los parámetros α y β son calculados de tal manera que no se exceda la máxima aceleración del robot. El perfil de velocidad en la Ec. 8.1 permite un suave incremento en la velocidad del robot desde 0 hasta la velocidad nominal v_n.

8.3.2. Control de Velocidad Angular

La velocidad angular es utilizada para corregir las desviaciones desde los valores de referencia en ángulos y distancias.

Para obtener la velocidad angular del robot, se utiliza un control de modos deslizantes *Super-Twisting*.

La ventaja de este tipo de control es su propiedad de convergencia en tiempo finito del error para la superficie deslizante [98], [99], las cuales rápidamente alcanzan los valores de referencia de control. En este trabajo, la superficie deslizante es una combinación lineal de los valores de referencia de distancia y ángulo los cuales deben ser llevados a cero. La velocidad angular ω_s se calcula como sigue:

$$\omega_s = -k_4|s|^{\frac{1}{2}}\text{sgn}(s) + \sigma \tag{8.2}$$

donde $\dot{\sigma}$ está dado por:

$$\dot{\sigma} = -k_3\text{sgn}(s) \tag{8.3}$$

y la superficie deslizante s está dada por la siguiente ecuación:

$$s = k_1 e_d + k_2 e_\theta \tag{8.4}$$

k_1, k_2, k_3 y k_4 son ganancias de control, las ganancias utilizadas son sintonizadas manualmente, se inicia con pequeñas ganancias positivas y después estos valores son incrementados hasta obtener la más rápida convergencia del error a cero. Este proceso se repite mientras persista la oscilación en la trayectoria del robot.

Finalmente, los errores del control e_θ y e_d, que están relacionados respectivamente con la orientación y la distancia entre robot y los obstáculos, dichos valores están dados por las siguientes ecuaciones:

$$e_\theta = \begin{cases} \theta_1 & if \quad \text{SL} \\ \theta_d - \theta_2 & if \quad \text{CCWT} \\ \theta_d - \text{mín}\{\theta_3, \theta_4\} & if \quad \text{CWT} \end{cases} \tag{8.5}$$

$$e_d = \begin{cases} d_d - d_1 & if \quad \text{SL} \\ h_0 - h & if \quad \text{CCWT} \\ d_d - d_{corner} & if \quad \text{CWT} \end{cases} \tag{8.6}$$

donde θ_d sigue un perfil evitando discontinuidades no deseadas en los controles. Este perfil se describe en la siguiente sección.

8.3.3. Transición Suave entre Especificaciones de la Tarea

Debido a que la orientación del robot con respecto a una referencia puede cambiar abruptamente desde un estado a otro, entonces el error en la orientación puede cambiar de un valor a otro valor muy diferente. Esto genera saltos no deseados en los controles. Con la finalidad de evitar estas discontinuidades en los errores, se utiliza un ángulo deseado θ_d el cual cambia suavemente desde un valor inicial θ_i hasta un valor final $\theta_f = 0$ en un tiempo predeterminado (τ) de acuerdo con la siguiente ecuación:

$$\theta_d = \frac{\theta_{i\in\{2,3,4\}|t=0}}{2}\left(1+\cos\left(\frac{\pi t}{\tau}\right)\right) \tag{8.7}$$

8.4. Simulaciones y Experimentos en un Robot Real

8.4.1. Resultados de las Simulaciones

Todas las simulaciones fueron realizadas en una Computadora con un procesador octa-core Core i7-4790 a 3.6 GHz, equipada con una memoria RAM de 16 GB, sistema operativo Linux, y programadas en el lenguaje C++ y utilizando la librería de geometría LEDA. La implementación en software emula exactamente el autómata que se presentó en la Figura 8.2.

Las Figuras 8.10 y 8.11, presentan algunas capturas tomadas de la simulación realizada. Un conjunto de segmentos de líneas color negro conforman el entorno poligonal sobre el cual se realiza la tarea de navegación. El robot real es representado como un disco negro y su *heading* se muestra como una línea roja gruesa que va desde el centro del robot hasta la periferia de este. Un círculo rojo centrado en el centro del robot representa un robot virtual circular para seguir la frontera del entorno estando en contacto o muy cerca de dicha frontera. El círculo azul representa las trayectorias de arco de círculo que debe seguir el robot sobre la frontera de dicho círculo cuando el robot pasa sobre una esquina. En color cian se muestra el punto rp'. El punto verde se representa la ubicación del sensor en el punto rp. En amarillo se muestra la región de visibilidad del robot que es posible obtener por medio del sensor láser de rango. En blanco se muestra todo aquello que el robot no es capaz de ver desde su ubicación actual. Las líneas rojas punteadas son discontinuidades en

distancia detectadas por el sensor láser de rango.

La Figura 8.10 muestra la simulación del robot moviéndose en arco de círculo en sentido contrario a las manecillas del reloj siguiendo la trayectoria dada por el círculo azul cuando se presenta una esquina cóncava.

En la Figura 8.11 se muestra al robot siguiendo una trayectoria de arco de círculo en sentido de las manecillas del reloj cuando se presenta una esquina convexa.

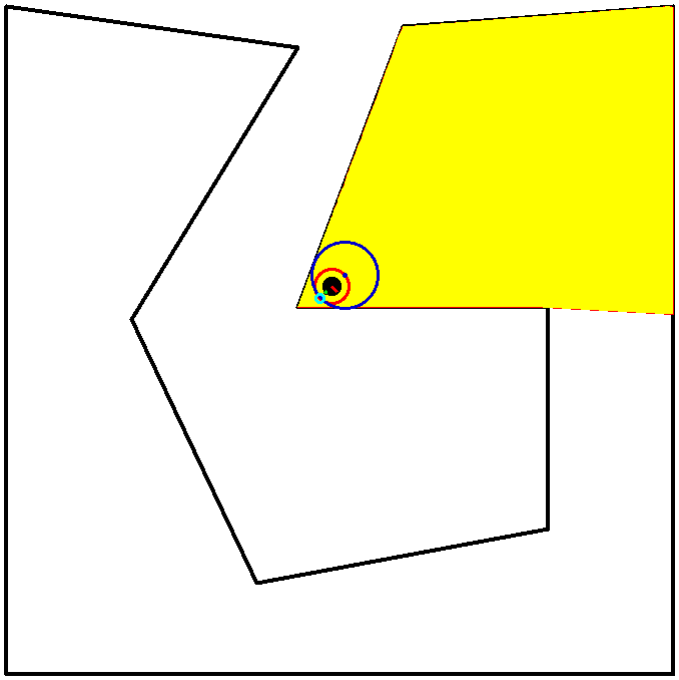


Figura 8.10: $CCWT$ en esquina cóncava

8.4.2. Experimentos con un Robot Real

En todos los experimentos, se utilizó un robot de manejo diferencial llamado Pioneer P3-DX. El robot es modelado como un disco de radio igual a $0.2m$, el robot tiene una velocidad de traslación máxima de 1.2 m/s y una velocidad rotacional máxima de 5.236 rad/s. Para los experimentos, la máxima velocidad lineal nominal v_n, se fija en 0.3 m/s. La distancia deseada d_d entre el centro del robot y la frontera del entorno se fija a 0.4 m. Así la distancia entre la frontera del robot y el entorno es controlada para mantenerse en 0.2 m.

En la implementación, los algoritmos son ejecutados directamente desde la computadora del robot, la cual cuenta con un procesador Pentium M a 1.8 GHz y una memoria RAM

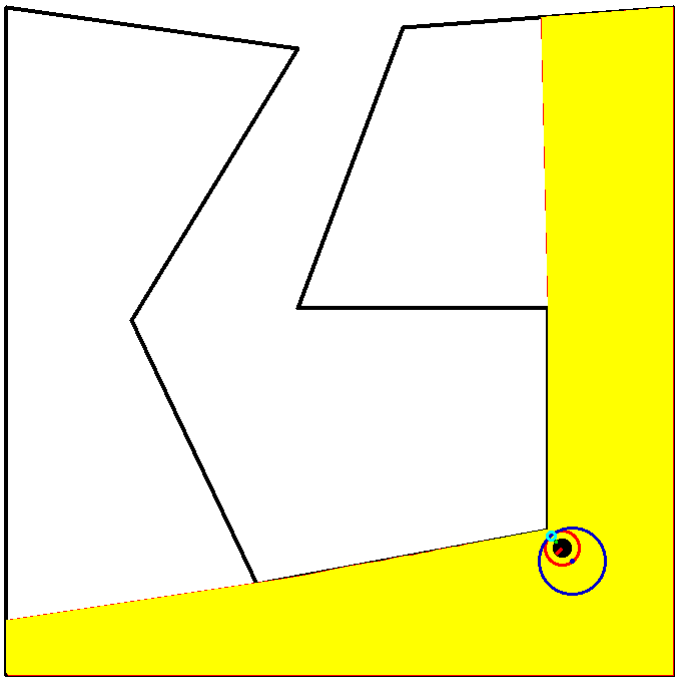


Figura 8.11: *CWT* en esquina convexa

de 1 GB. El sistema operativo es Linux y se utiliza algunas funciones de ROS, y los ciclos de control corren a 12.5 Hz. El software es programado en el lenguaje C++. El sensor omnidireccional fue implementado utilizando dos láser de rango Hokuyo modelo URG-04LX, los cuales son montados sobre el robot en direcciones opuestas, ver Figura 8.12.

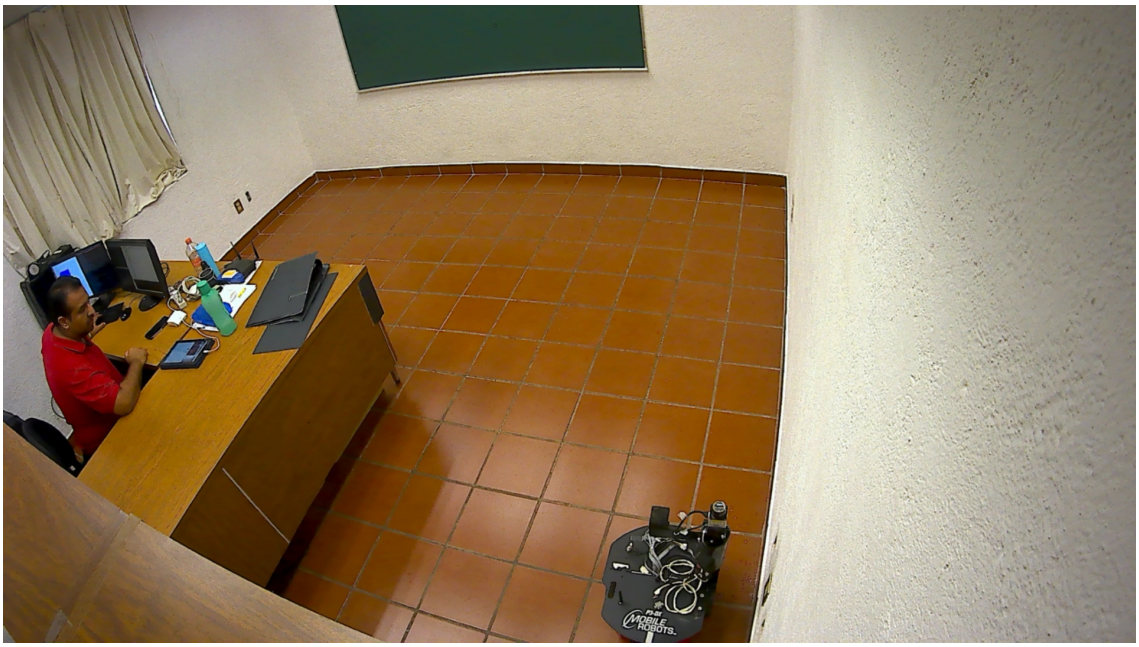
Los experimentos fueron realizados en dos entornos diferentes, ver Figuras 8.13(a) y 8.13(b), que son los mismos entornos utilizados los experimentos de la Sección 7.2. El principal objetivo de estos experimentos es probar el desempeño y la capacidad para seguir pared del autómata, el cual es operado por el controlador de modos deslizantes llamado *Super-Twisting*.

En la Figura 8.14, se muestra un experimento realizado en el laboratorio de robótica de CIMAT. En la Figura 8.15, se presenta un experimento realizado en una oficina de CIMAT.

A continuación se presentan las estadísticas resultantes de la navegación para esos dos entornos: el laboratorio de robótica y una oficina del CIMAT. Para demostrar el desempeño del robot al ejecutar la tarea de navegación que se presenta en este Capítulo, se hace una comparativa contra los resultados estadísticos obtenidos mediante el esquema de navegación presentado en los Capítulos 5, 6 y 7. Los resultados son presentados en las Tablas 8.2 y 8.3. Para cada estrategia y para cada entorno se presentan las estadísticas de 3 vueltas.

Figura 8.12: El Robot y los sensores láser.

(a) Laboratorio



(b) Oficina

Figura 8.13: Entornos utilizados para los experimentos.

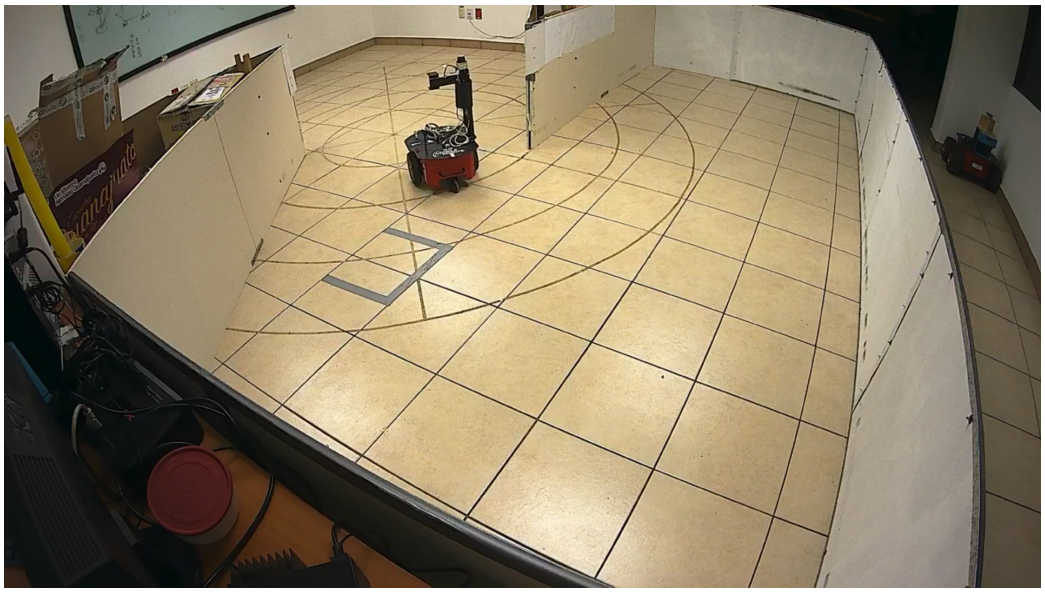


Figura 8.14: *CWT* en esquina convexa en el laboratorio.

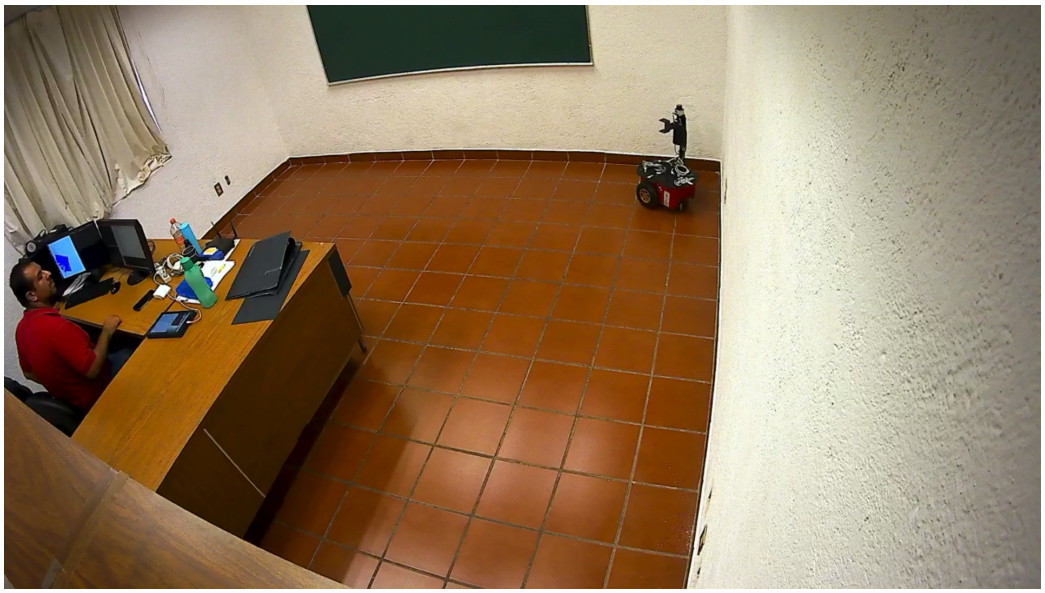


Figura 8.15: *CCWT* en esquina cóncava en una oficina.

Para cada vuelta que el robot ejecuta se presenta la media de la distancia medida entre el centro del robot y la frontera del entorno, la correspondiente desviación estándar, los valores máximos y mínimos de esta distancia, así como el tiempo por vuelta que le tomo al robot viajar por la frontera del entorno. Nótese que la ventaja principal que tiene el esquema propuesto en este Capítulo con respecto al esquema presentado en los Capítulos 5, 6 y 7 es que reduce el tiempo para recorrer el entorno.

Para el entorno en laboratorio, la estrategia propuesta en los Capítulos 5, 6 y 7, presenta un tiempo promedio por vuelta de 162.136 s. El esquema presentado en este Capítulo, muestra un tiempo por vuelta de 83.62 s, lo cual significa que, al no tener que detenerse en las esquinas, hay una mejora en el tiempo por vuelta de aproximadamente 78.516 s.

Para el entorno de Oficina, el esquema propuesto en los Capítulos 5, 6 y 7, presenta un tiempo promedio por vuelta de 59.616 s. En cuanto al esquema presentado en este Capítulo, muestra un tiempo por vuelta de 43.1093 s, por tanto, existe una mejora en el tiempo por vuelta de aproximadamente 16.506 s. Nótese que el entorno en el laboratorio es mucho más grande que el de la oficina, el perímetro del mapa correspondiente al laboratorio es de 27.4 m y el de la oficina es de 16.04 m.

Estadísticas en el laboratorio de robótica del CIMAT					
Estadísticas del esquema presentado en los Capítulos 5, 6 y 7					
Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.41	0.066	0.813	0.285	163.488
2	0.418	0.076	0.811	0.271	162.032
3	0.412	0.064	0.81	0.292	160.888
Estadísticas obtenidas del esquema propuesto en este Capítulo					
Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.416	0.07	0.604	0.267	84.42
2	0.436	0.078	0.665	0.238	82.92
3	0.412	0.08	0.632	0.234	83.52

Tabla 8.2: Comparación de estadísticas obtenidas para la navegación en el laboratorio de robótica del CIMAT entre el esquema presentado en los Capítulos 5, 6 y 7 vs el esquema propuesto en éste Capítulo.

En la Figura 8.16, se presenta la evolución de la velocidad angular ω en el entorno del laboratorio del CIMAT así como el intervalo de tiempo que dura la ejecución de cada estado en el autómata (la duración de cada estado es etiquetada en la Figura 8.16) el cual

Estadísticas en oficina del CIMAT					
Estadísticas de la Tabla 7.1					
Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.364	0.034	0.447	0.291	59.04
2	0.359	0.038	0.441	0.293	58.368
3	0.358	0.041	0.431	0.27	61.44
Estadísticas obtenidas del esquema propuesto en este Capítulo					
Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.43	0.06	0.598	0.293	43.102
2	0.444	0.073	0.69	0.286	42.568
3	0.463	0.109	0.818	0.251	43.658

Tabla 8.3: Comparación de estadísticas obtenidas para la navegación en una oficina del CIMAT entre el esquema presentado en los Capítulos 5, 6 y 7 vs el esquema propuesto en éste Capítulo.

es operado por el control de modos deslizantes llamado *Super-Twisting*. De igual manera, en la Figura 8.17, se presentan las mimas gráficas pero para un entorno de oficina en el CIMAT.

Los experimentos permiten la validación del método de navegación desarrollado en este capítulo. Por medio de las gráficas mostradas en las Figuras 8.16 y 8.17 es posible verificar que la velocidad angular es ruidosa mientras que la velocidad lineal permanece constante, esto es debido a errores en las lecturas del sensor que hacen variar la dirección a la que se encuentra el obstáculo más cercano. El siguiente enlace da acceso a un video que muestra los dos experimentos realizados con en el robot real. El primer experimento fue realizado en el laboratorio de robótica del CIMAT y el segundo en una oficina del CIMAT.

Click aquí para enlace a video

En este Capítulo se ha presentado un autómata que mediante las observaciones del robot es capaz de elegir los valores de referencia que requiere el control de acuerdo al estado interno del autómata. El control de dicho autómata es realizado mediante un control de modos deslizantes llamado *Super-Twisting*, esto permite al robot moverse continuamente sin necesidad de parar cuando los obstáculos (esquinas) se presentan en la trayectoria del robot, es por ello que las estadísticas muestran una mayor rapidez al realizar cada vuelta en cualquiera de los dos entornos presentados (entorno en el laboratorio y una oficina en CIMAT). Este esquema, permite que un solo controlador regule las velocidades lineal y

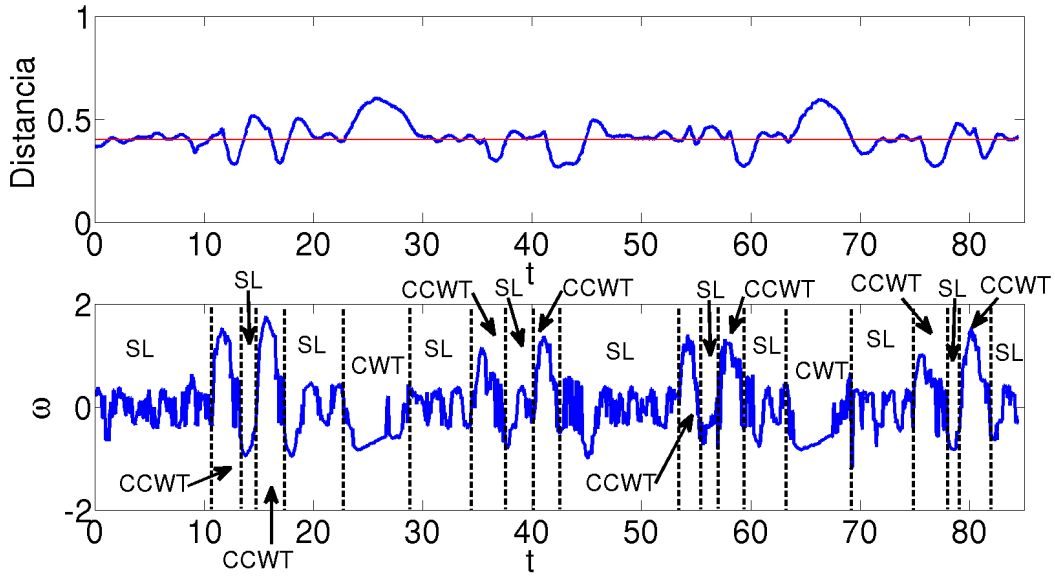


Figura 8.16: Gráficas de distancia más corta y velocidad angular controlada por el control de modos deslizantes *Super-Twisting* para el entorno en el Laboratorio de CIMAT

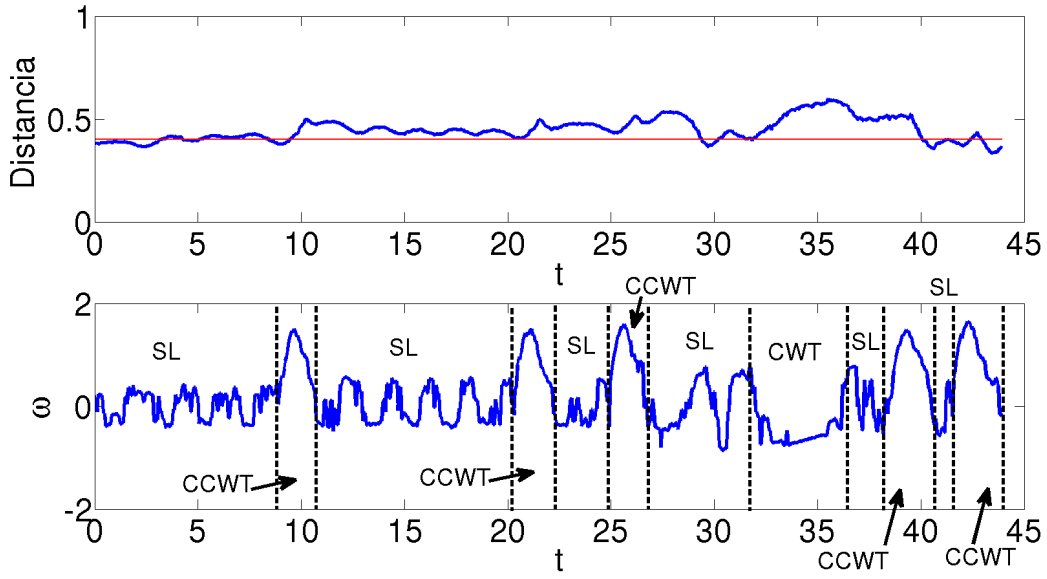


Figura 8.17: Gráficas de distancia más corta y velocidad angular controlada por el control de modos deslizantes *Super-Twisting* para el entorno de Oficina del CIMAT

angular utilizadas en cada uno de los estados del autómata.

8.5. Un Esquema de Control donde la Velocidad Lineal del Robot se Reduce

Considerando la inercia del robot y los deslizamientos que el robot puede presentar al no detenerse en las esquinas, el controlador de velocidad lineal es rediseñado de manera que el robot pueda reducir gradualmente su velocidad lineal a un valor diferente de cero al acercarse a las esquinas cóncavas y convexas durante el estado SL del autómata. Para diseñar este controlador, es necesario agregar un tercer disco sobre el robot y así definir el espacio que tiene el robot para ejecutar una desaceleración, de manera que al detectar un obstáculo, el robot reduzca su velocidad lineal hasta una velocidad a la que se desea ejecutar los arcos de círculo relacionados con los estados $CCWT$ y CWT del autómata, dicha velocidad es llamada v_{corner}. Este nuevo disco de radio d_s es similar al presentado en el Capítulo 5, solo que éste es concéntrico al circulo de radio d_t, de manera que $d_s > d_t$, ver Figura 8.18.

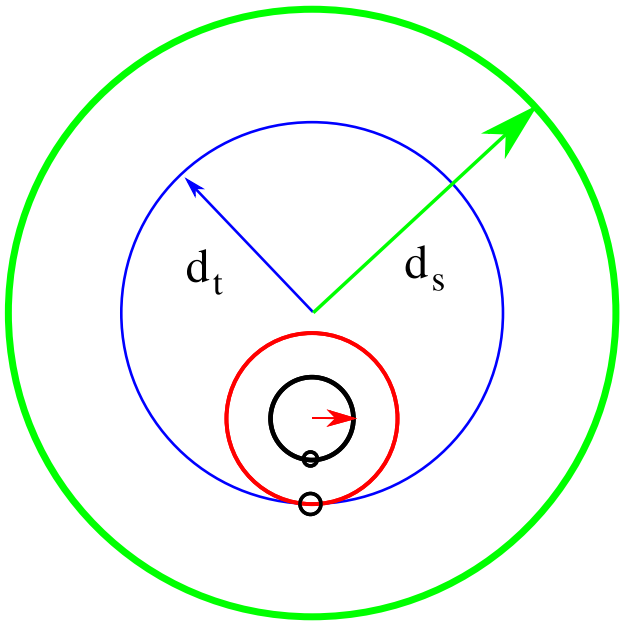


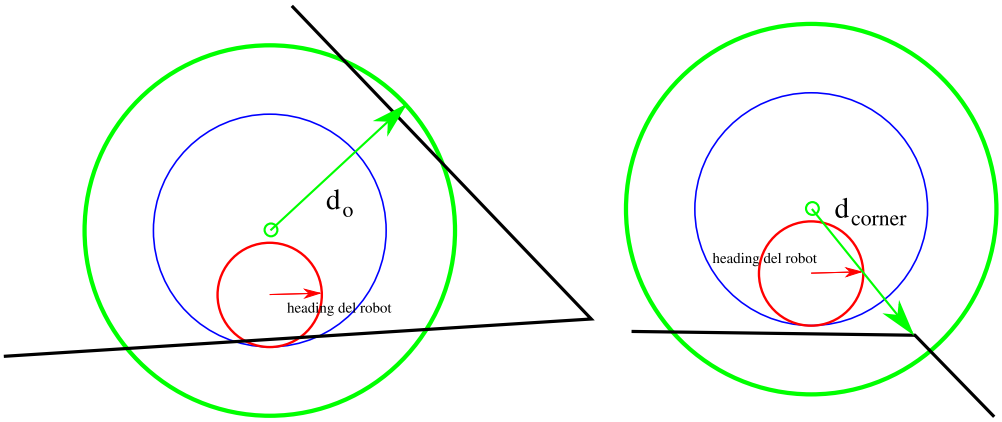
Figura 8.18: Disco de radio d_s en color verde

Si un obstáculo es detectado a distancia menor que d_s durante la ejecución del estado SL en el autómata, se activará un nuevo control para la velocidad lineal que esta determinado

por la siguiente ecuación:

$$V = \begin{cases} \frac{v_n - V_{c|t=0}}{2}(1 + \tanh(\alpha(t-\beta))) + V_{c|t=0} & if \quad \text{SL y } !Obstacle \\ k_5 e_o + v_{corner} & if \quad \text{SL y } Obstacle \\ v_{corner} & if \quad \text{CCWT o CWT} \end{cases} \tag{8.8}$$

donde k_5 es una ganancia proporcional y $e_o = d_t - min\{d_o, d_{corner}\}$. Nótese que d_o es la distancia medida desde el centro del círculo de radio d_s hasta el obstáculo más cercano que no pertenece a la pared que el robot está siguiendo (ver Figura 8.19(a)) y d_{corner} es la distancia entre el centro del círculo de radio d_s y la esquina visible más cercana (ver Figura 8.19(b)). Ambas medidas d_o y d_{corner} son similares a las utilizadas en el Capítulo 5 pero medidas desde un origen diferente. Para evitar discontinuidades en la velocidad lineal, k_5 se calcula con la siguiente ecuación: $k_5 = \frac{Vc - v_{corner}}{d_t - min\{d_o, d_{corner}\}}$ únicamente al instante en que se activa el controlador.



(a) d_o (b) d_{corner}

Figura 8.19: Distancias d_o y d_{corner}.

8.5.1. Simulación

A continuación se muestra las simulaciones obtenidas dadas las modificaciones para que el robot pueda reducir su velocidad lineal al acercarse a las esquinas convexas y cóncavas.

En la Figura 8.20 se puede observar cuando la pared que no se está siguiendo entra al área de detección definida por el disco de radio d_s, entonces la velocidad lineal se reduce de acuerdo a la medición d_o. De igual manera, en la Figura 8.21 se presenta el momento en

que la esquina convexa activa la reducción de la velocidad lineal del robot tomando como información de retroalimentación a la medición d_{corner}.

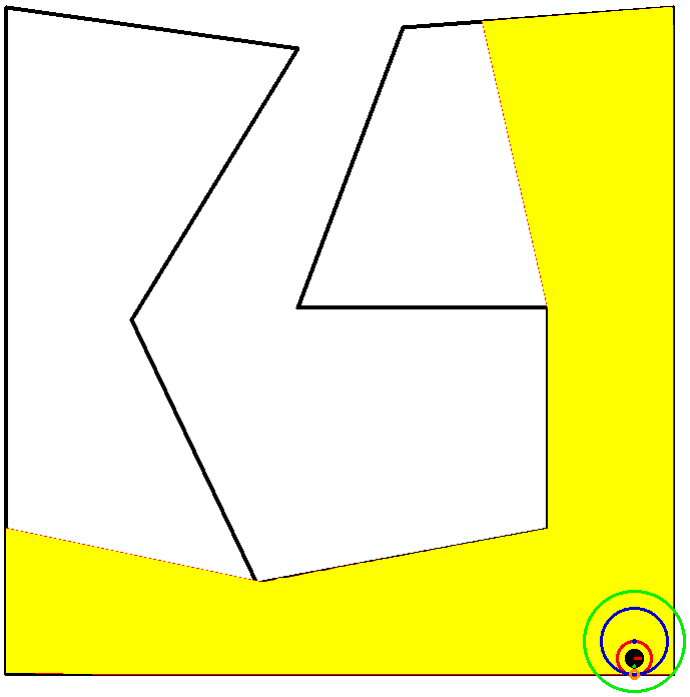


Figura 8.20: Reducción de velocidad con distancia d_o

Finalmente en la Figura 8.22 se presentan las gráficas correspondientes a la distancia más corta, y velocidades lineal y angular respecto al tiempo, para la simulación realizada. En las gráficas que muestran la velocidad lineal y angular se presenta las velocidades ejecutadas durante el intervalo de tiempo que dura cada estado del autómata al recorrer una vuelta en el entorno simulado. Para la velocidad lineal se presenta valores entre $v_n = 1.0$ y $v_{corner} = 0.5$.

El siguiente enlace presenta un video, el cual muestra dos vueltas de la simulación correspondiente al esquema presentado en esta sección, en donde la velocidad lineal del robot se reduce:

Click aquí para enlace a video

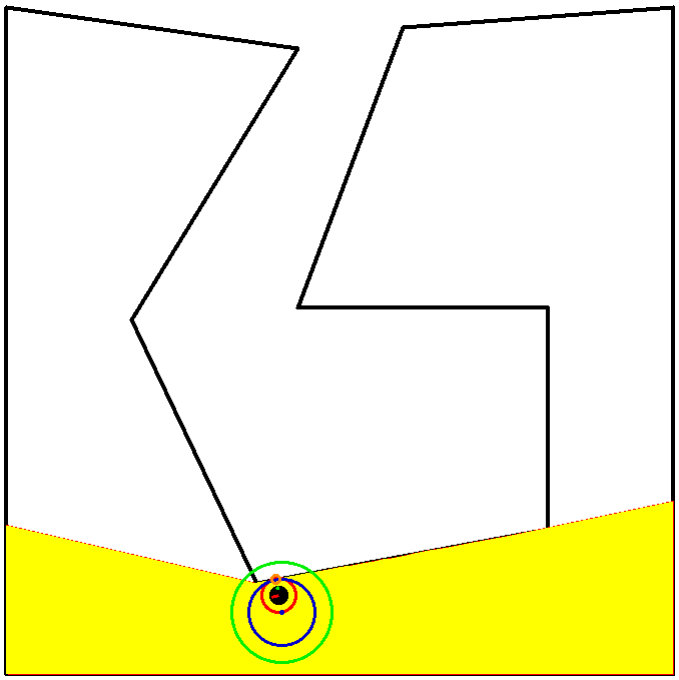


Figura 8.21: Reducción de velocidad con distancia d_{corner}

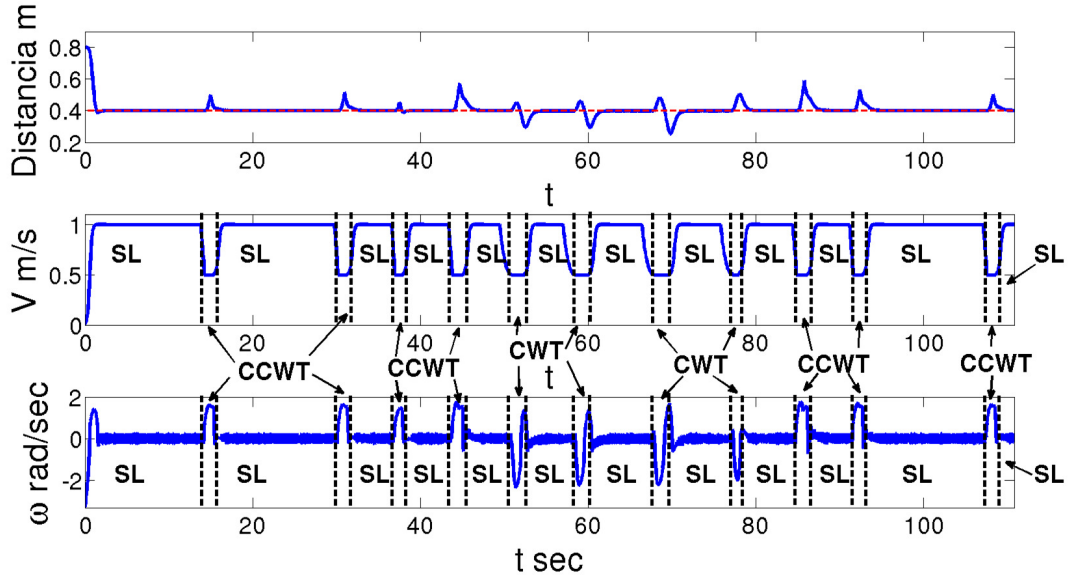


Figura 8.22: Gráficas de distancia más corta y velocidades lineal y angular

8.5.2. Experimentos en un Robot Real donde el Controlador Reduce la Velocidad Lineal

Para validar el esquema presentado en la Sección 8.5, el método es implementado en un robot Pioneer P3-DX. La distancia deseada d_d entre el centro del robot y la frontera del entorno se mantiene en 0.4 m. El radio d_t del circulo que define la trayectoria de arco de círculo del robot en la esquina cóncava se fija a 0.6 m. En la velocidad lineal se presentan valores entre $v_n = 0.5$ y $v_{corner} = 0.25$.

Los experimentos fueron realizados en una oficina del CIMAT, la Figura 8.23 muestra dicho entorno. El objetivo es probar el desempeño del autómata para el seguimiento de pared, el cual es operado por un controlador *Super-Twisting* y la velocidad lineal del robot se reduce al aproximarse a las esquinas. De este modo, es posible aumentar la velocidad durante el seguimiento de pared en las líneas rectas y así reducir el tiempo de navegación por vuelta en el entorno.

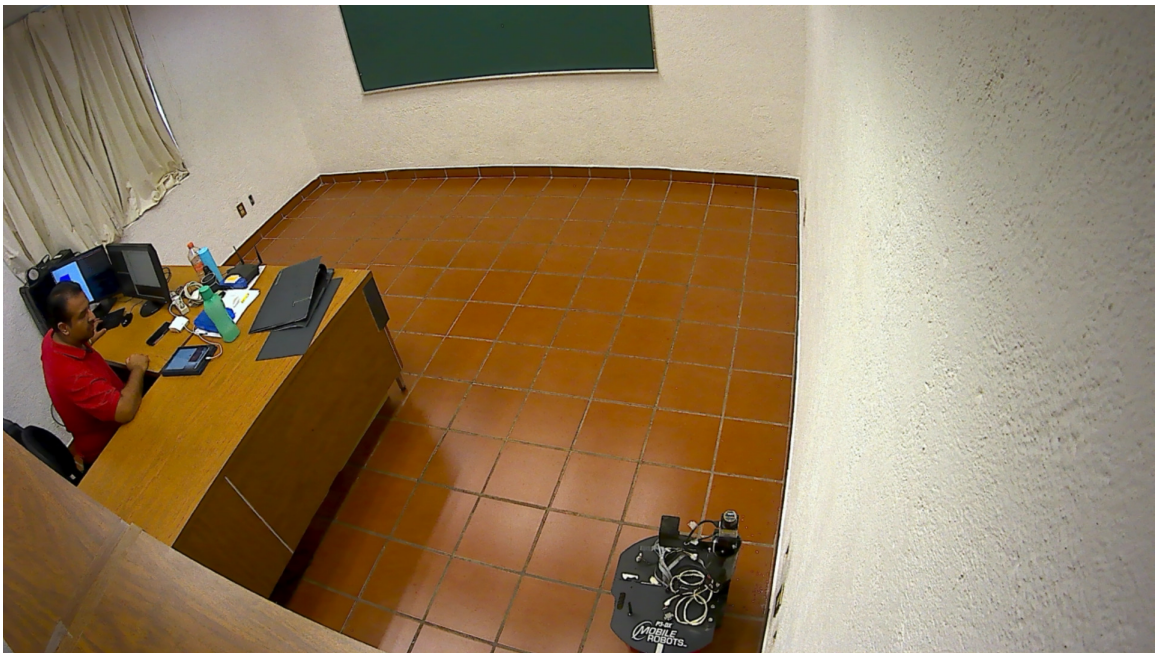


Figura 8.23: Entorno de Oficina utilizado para los experimentos.

Los resultados son presentados en la Tabla 8.4 donde se muestra las estadísticas para 3 vueltas y se hace una comparación con respecto al esquema de control donde el autómata es operado por el controlador *Super-Twisting* y mantiene la velocidad lineal constante. Para este entorno de oficina, se observa que el tiempo promedio por vuelta es de 35.295 s,

obteniendo una reducción de tiempo por vuelta de 7.8143 s respecto al esquema anterior. Nótese que el entorno de oficina tiene un perímetro de 16.04 m.

Experimentos en la oficina del CIMAT					
El robot viaja a velocidad lineal constante					
Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.43	0.06	0.598	0.293	43.102
2	0.444	0.073	0.69	0.286	42.568
3	0.463	0.109	0.818	0.251	43.658
El robot reduce su velocidad lineal al aproximarse a las esquinas					
Número de Vuelta	Promedio de Distancia [m]	Desviación Estándar [m]	Distancia Máxima[m]	Distancia Mínima [m]	Tiempo por Vuelta [seg]
1	0.439	0.114	0.831	0.248	35.342
2	0.417	0.08	0.65	0.248	36.609
3	0.416	0.121	0.757	0.222	33.934

Tabla 8.4: Comparación de las estadísticas obtenidas para la navegación en el entorno de oficina del CIMAT para el autómata operado por el controlador *Super-Twisting* cuando la velocidad lineal se mantiene constante vs cuando la velocidad lineal se reduce al aproximarse a las esquinas.

En la Figura 8.24 se presentan las gráficas de la evolución de las velocidades lineal y angular así como el intervalo de tiempo que dura la ejecución de cada estado del autómata.

Los experimentos realizados permiten validar que el método presentado en la Sección 8.5 el cual permite disminuir el tiempo de navegación al reducir la velocidad lineal del robot al aproximarse a las esquinas y poder aumentar dicha velocidad durante el seguimiento de pared en línea recta.

El siguiente enlace da acceso a un video que muestra todos los experimentos realizados con el robot real implementando todos los esquemas presentados en este Capítulo. En la tercera parte del video se muestra el experimento correspondiente al esquema donde el autómata es operado por el control *Super-Twisting* y se reduce la velocidad lineal del robot al aproximarse a las esquinas.

Click aquí para enlace a video

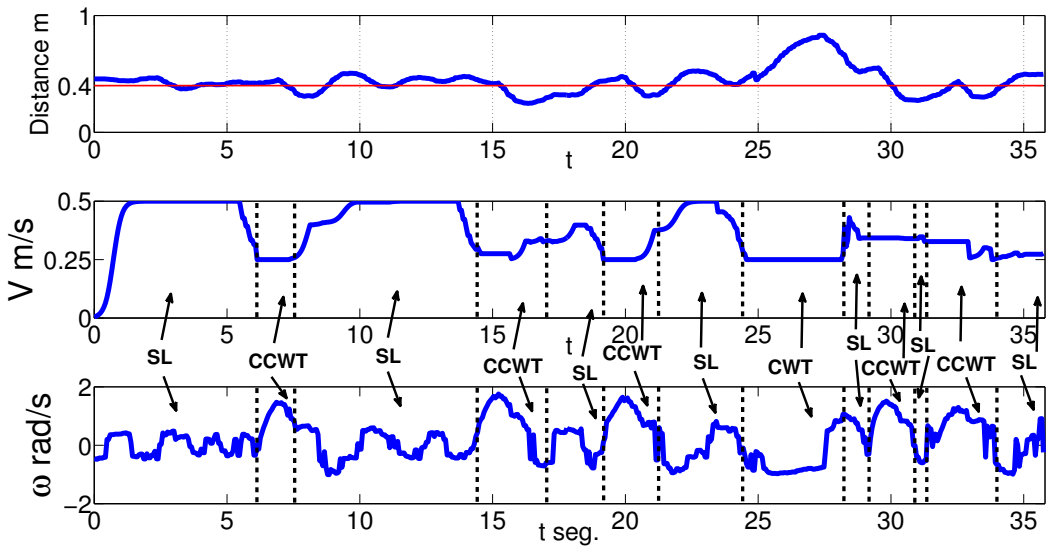


Figura 8.24: Gráficas de distancia a la pared (arriba), velocidad lineal (enmedio) y velocidad angular (abajo).

Capítulo 9

Conclusiones y Trabajo Futuro

Esta tesis se enfoca en el problema de la exploración de entornos desconocidos utilizando un robot de manejo diferencial con forma de disco. Para explorar el ambiente, el robot sigue la frontera del entorno. El robot está equipado con sensores que son típicos en el ámbito de la robótica y la estrategia de exploración propuesta no requiere la localización del robot.

El problema de exploración abordado en esta tesis es más desafiante que el caso donde el robot es un punto debido a que la información de visibilidad no siempre genera trayectorias libres de colisión en el espacio de configuraciones. En este trabajo, se propone una estrategia de exploración la cual es modelada mediante una máquina de Moore, y esta garantiza la exploración de todo el entorno o la mayor región posible de este. El robot es capaz de encontrar un *landmark* o declarar que no existe una estrategia de exploración para cumplir con este objetivo. Se propone una política de movimiento basada en retroalimentación sensorial.

Se propone un esquema de control híbrido que hace frente a la imperfección en los comandos que el robot ejecuta, y así tratar con la dinámica del robot (ej. variaciones en las velocidades). Además, el esquema de control propuesto mantiene la continuidad en las velocidades lineal y angular del robot a pesar de la conmutación entre controladores. La principal diferencia con respecto los trabajos previos sobre seguimiento de pared es que en este enfoque, la FSM restringe las posibles transiciones de estados filtrando las observaciones espurias debidas al ruido en las lecturas del sensado. Se enfatiza que en este enfoque, la etapa de planeación corresponde al diseño de la FSM la cual se realiza antes de la ejecución, una vez que se ha diseñado, el método es reactivo, para cualquier instancia de la ejecución y para cualquier entorno diferente, el método relaciona de manera directa las observaciones y los controles.

Se presenta también una estrategia alternativa de navegación para seguir la frontera del ambiente, donde un solo controlador es usado para controlar la velocidad lineal y angular, en el cual solo los valores de referencia de control cambian con el estado del autómata. También se sugiere un controlador de modos deslizantes que tiene la propiedad de alcanzar la consigna de control en tiempo finito. Esta nueva propuesta de control permite no detener al robot cuando encuentra un obstáculo.

Todos los algoritmos propuestos han sido implementados y tanto las simulaciones como los experimentos son presentados para validar este enfoque. Los resultados experimentales en un robot real han coincidido con el modelado propuesto.

Como trabajo futuro, se buscará extender el enfoque actual para la ejecución basada en retroalimentación para cualquier tipo de trayectorias y no solo para el seguimiento de pared.

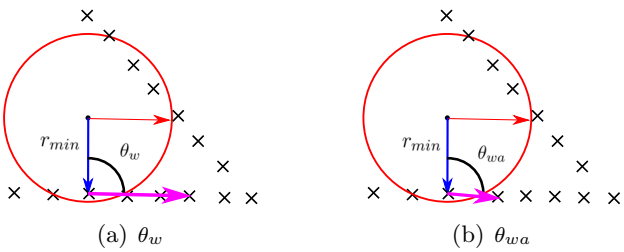
Los principales resultados de esta tesis se reportan en [106]. Además, la estrategia de navegación presentada en el Capítulo 8 de esta tesis se someterá próximamente a una revista internacional indizada [107].

Apéndice A

Algoritmos de Ajuste de Líneas

A.1. Rotación en sitio

En el Algoritmo de ajuste de líneas, el ángulo θ_w es utilizado para caracterizar las lecturas del sensor (puntos láser) que pertenecen a la pared que el robot está siguiendo. El ángulo θ_w es el ángulo entre el rayo r_{min} y la línea que pasa sobre el punto sensado al que r_{min} está apuntando y un punto a la región de la frontera del obstáculo, el cual es posterior en orden angular (en sentido contrario a las manecillas del reloj) al punto que r_{min} está apuntado, ver Figura A.1(a). El segmento de línea que el robot está siguiendo es determinado por el punto al que apunta el rayo r_{min} y el siguiente punto en la lectura del sensor en sentido contrario de las manecillas del reloj. El ángulo θ_{wa} es el ángulo entre entre el rayo r_{min} y la línea que pasa sobre el punto sensado al que r_{min} está apuntando y el siguiente punto sensado, en sentido contrario a las manecillas del reloj, de la región de la frontera del obstáculo (alternativamente en lugar del siguiente punto, se puede usar el promedio de los ángulos de los siguientes n puntos sensados), ver Figura A.1(b). Los puntos que pertenecen a la pared que el robot está siguiendo deben tener un ángulo $\theta_w \in (\theta_{wa} - \epsilon_0, \theta_{wa} + \epsilon_0)$.



(a) θ_w (b) θ_{wa}

Figura A.1: Cálculo del ángulos θ_w y θ_{wa} usando los puntos láser

Tanto las esquinas convexas como las esquinas cóncavas pueden ser ubicadas utilizando el ángulo θ_w. Si hay dos puntos consecutivos en orden angular, el primero (en sentido contrario a las manecillas del reloj) teniendo un ángulo asociado de $\theta_w \in (\theta_{wa} - \epsilon_0, \theta_{wa} + \epsilon_0)$ y el segundo teniendo un ángulo $\theta_w > \theta_{wa} + \epsilon_0$, entonces hay una esquina convexa. Si hay una esquina cóncava uniendo dos segmentos, es posible detectarlo utilizando el ángulo θ_w. Si hay dos puntos sensados y estos son consecutivos en orden angular, el primero (en sentido contrario a las manecillas del reloj) teniendo un ángulo asociado $\theta_w \in (\theta_{wa} - \epsilon_0, \theta_{wa} + \epsilon_0)$ y el segundo teniendo un ángulo $\theta_w < \theta_{wa} - \epsilon_0$, entonces se dice que una esquina cóncava está presente.

El Algoritmo 2 encuentra el último segmento en sentido contrario a las manecillas del reloj en la región de la frontera del obstáculo para alinear el *heading* del robot con este. Este algoritmo es también utilizado para encontrar una esquina convexa para iniciar el movimiento en arco de círculo. El Algoritmo 2 funciona mediante la eliminación interactiva de puntos láser que pertenecen a los segmentos de línea de la región de la frontera del obstáculo que ya ha sido procesada. El Algoritmo 2 considera que el robot (y por lo tanto también el sensor omnidireccional) está dentro del entorno poligonal y que este no tiene acceso al mapa completo, pero usa solo los puntos láser los cuales son visibles, esto es equivalente a razonar sobre el polígono de visibilidad. Ver Figura 5.7. El Algoritmo 2 es capaz de detectar esquinas convexas, esquinas cóncavas y los segmentos que los unen. Se enfatiza que el Algoritmo 2 toma ventaja del orden angular en el cual los puntos del láser son almacenados, esto permite un mejor desempeño en el procesamiento.

A.2. Rotación con respecto a una esquina convexa

El ángulo θ_{cw} es utilizado para detectar puntos de colisión potenciales que no pertenecen al segmento de línea posterior a la esquina convexa. El ángulo θ_{cw} es el ángulo entre el rayo r_{corner} y la línea que pasa sobre la esquina que r_{corner} está apuntando y un punto sensado de la región de la frontera del obstáculo, la cual es posterior en orden angular (en sentido contrario a las manecillas del reloj) a la esquina que r_{corner} está apuntando, ver Figura A.2(a). Ángulo θ_{cwa} es el ángulo medido entre el rayo r_{corner} y la línea que pasa sobre la esquina convexa el siguiente punto sensado en, sentido contrario a las manecillas del reloj (alternativamente en lugar del siguiente punto, se puede usar el promedio de los ángulos de los siguientes n puntos sensados), ver Figura A.2(b). Para una esquina convexa, todos los puntos pertenecientes al segmento posterior a la esquina tienen un ángulo $\theta_{cw} \in (\theta_{cwa} - \epsilon_0, \theta_{cwa} + \epsilon_0)$.

Algoritmo 2 Rotation in place

Entrada: $SensorReadings$.
Salida: θ_2.
1. Considerar el ultimo segmento circular del robot que intersecta la región del obstáculo;

while $SensorReadings \neq empty$ **do**
2. Calcular la distancia más corta a la región del obstáculo;
3. Medir θ_2, el ángulo entre la dirección del rayo r_{min} y la línea $rp - rp'$;
4. Rotar el robot en un ángulo θ_2 en sentido contrario a las manecillas del reloj:
5. $true \leftarrow ConvexCorner(SensorReadings)$ esta función determina si el rayo r_{min} está o no apuntando a la esquina convexa;
if $true = 1$ **then**
6. $Out \leftarrow NxtPtsC(SensorReadings)$;
if $Out = Empty$ **then**
if punto rp' está tocando una esquina convexa **then**
return Éxito;
else
return Falla, rp no toca la esquina convexa;
end if
else
7. $SensorReadings = Out$;
end if
else
8. $Out \leftarrow NxtPtsS(SensorReadings)$;
9. $Aligned \leftarrow NxtPtsS(SensorReadings)$
if $Out = Empty$ **then**
if $Aligned = 1$ **then**
return Alineado;
else
return Falla, no alineado;
end if
else
10. $SensorReadings = Out$;
end if
end if
end while

Subroutine 3 $ConvexCorner(SensorReadings)$

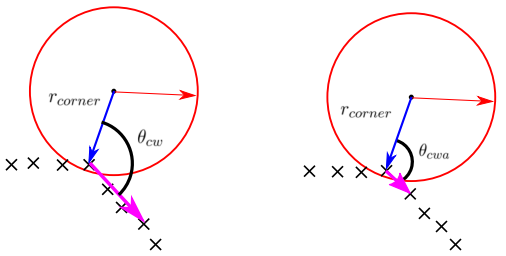
Entrada: $SensorReadings$ es un conjunto de lecturas del sensor láser.
Salida: $CvxC$ Este bit indica si el rayo r_{min} está o no apuntando a una esquina convexa.
1. Calcular una línea desde el punto al que r_{min} está apuntando hasta el punto láser previo (en sentido contrario de las manecillas del reloj), esta línea es llamada l_a;
2. Calcular una línea desde el punto al que r_{min} está apuntando hasta el punto láser siguiente (en sentido contrario de las manecillas del reloj), esta línea es llamada l_b;
3. Medir el ángulo desde l_a hasta l_b–ángulo interno utilizado para definir un vértice reflejo–, este ángulo es llamado θ_{CvxC};
if Ángulo $\theta_{CvxC} > \pi + \epsilon_0$ **then**
  4. $CvxC = 1$;
else
  5. $CvxC = 0$;
end if
Return $CvxC$;

Subroutine 4 $NxtPtsC(SensorReadings)$

Entrada: $SensorReadings$ es un conjunto de lecturas del sensor láser.
Salida:Un nuevo conjunto modificadode puntos láser llamado $NewSet$.
1. Usar el ángulo $\theta_w \in (\theta_{wa} - \epsilon_0, \theta_{wa} + \epsilon_0)$ para caracterizar los puntos láser que pertenecen a la línea posterior (en sentido contrario a las manecillas del reloj) a la esquina que el rayo r_{min} está apuntando;
2. Encontrar el primer punto láser en sentido contrario a las manecillas del reloj que no pertenezca a este segmento, este punto es llamado $first$;
if punto $first$ es más lejano que la distancia d_d desde el centro del robot **then**
  3. $NewSet = Empty$;
else
  4. Desde el punto $first$ recolectar todos los puntos en sentido contrario a las manecillas del reloj que son más cercanos al centro del robot que una distancia d_d y almacenar estos puntos en $NewSet$;
end if
Return $NewSet$;

Subroutine 5 $NxtPtsS(SensorReadings)$

Entrada: $SensorReadings$ es un conjunto de lecturas láser.
Salida: Un conjunto de puntos láser modificado llamado $NewSet$.
Salida: El bit $Aligned$ que indica si el robot está o no alineado con un segmento.
1. Usar el ángulo $\theta_w \in (\theta_{wa} - \epsilon_0, \theta_{wa} + \epsilon_0)$ para caracterizar los puntos láser que pertenecen al segmento del obstáculo al cual el rayo r_{min} está apuntando;
2. Encontrar el primer punto láser en sentido contrario a las manecillas del reloj que no pertenezca a este segmento, este punto es llamado $first$;
if el punto $first$ es más lejano que una distancia d_d desde el centro del robot **then**
  3. $NewSet = Empty$;
  if Ángulo $|\theta_w - \frac{\pi}{2}| < \epsilon_2$ **then**
    4. $Aligned = 1$;
  else
    5. $Aligned = 0$;
  end if
else
  6. A partir del punto $first$, recolectar todos los puntos en sentido contrario a las manecillas del reloj que son más cercanos que la distancia d_d y almacenar estos puntos en el conjunto $NewSet$;
end if
Return $NewSet$;
Return $Aligned$;



(a) θ_{cw} (b) θ_{cwa}

Figura A.2: Cálculo del ángulos θ_{cw} y θ_{cwa} usando los puntos láser

El Algoritmo 6 es utilizado para detectar puntos de colisión potenciales. Este algoritmo utiliza la distancia a un punto sensado dado, la distancia d_{corner} y la ley de cosenos para calcular la distancia entre la esquina convexa y un punto sensado dado. Análogamente al algoritmo de rotación en sitio (Algoritmo 2), el Algoritmo 6 toma ventaja del orden angular en el cual los puntos del láser son almacenados para ejecutar el procesamiento. El Algoritmo 6 no tiene acceso al mapa completo, solamente usa los puntos láser los cuales son visibles. Este puede detectar esquinas convexas, esquinas concavas y segmentos de líneas.

Entonces, si existen puntos de colisión potencial (es decir, el conjunto $PtColPtsSet$ no está vacio) cercanos al robot que la distancia d_s, entonces el ángulo auxiliar θ_4 es medido para determinar el ángulo que el robot debe rotar.

Algorithm 6 PtCollisionPts

Entrada: $SensorReadings$ es un conjunto de lecturas láser.
Salida: Un conjunto de puntos láser modificado llamado $PtColPtsSet$.
1. Usar el ángulo θ_w para caracterizar los puntos láser que pertenecen al segmento del obstáculo posterior al punto al que el rayo r_{corner} está apuntando;
2. Encontrar el primer punto láser en sentido contrario a las manecillas del reloj con un ángulo $\theta_{cw} < \theta_{cwa} - \epsilon_0$ y que es más cercano al centro del robot que la distancia $d_{corner} + d_d$, este punto es llamado $first$;
if punto $first$ está vacío **then**
  3. $PtColPtsSet = Empty$;
else
  4. Desde el punto $first$ recolectar todos los puntos en sentido contrario a las manecillas del reloj que son más cercanos s ls esquina convexa que la distancia $d_{corner} + d_d$ y almacenarlos. those points in $PtColPtsSet$;
  Return $PtColPtsSet$;
end if

Apéndice B

Valores de Implementación

La implementación se llevó a cabo en un robot de manejo diferencial Pioneer P3-DX. En la Tablas B.1, B.2 y B.3 se muestran los parámetros utilizados en cada uno los esquemas presentados en esta tesis, algunos de estos parámetros son adaptables como se indica en la descripción de cada Tabla.

B.1. Parámetros de Implementación para el Esquema donde la Máquina de Moore Conmuta entre Diferentes Controladores

B.1 Parámetros de Implementación para el Esquema donde la Máquina de Moore Conmuta entre Diferentes Controladores

Esquema donde la máquina de estados conmuta entre diferentes controladores (Capítulo 6)		
Parámetro	Valor de Implementación	Descripción
b	0.2 m	Radio del disco que representa al robot real
d_d	0.4 m	Radio de robot virtual (círculo rojo)
d_s	0.8 m	Radio de círculo para detectar un obstáculo cercano y reducir la velocidad lineal del robot
V_d	0.33 m/s	Velocidad lineal deseada
ϵ_1	0.1	Tolerancia para definir rp'
ϵ_2	10°	Tolerancia para determinar la alineación con la pared de acuerdo al la medición de los ángulos θ_1, θ_2, θ_3 y θ_4
α	3.8293	Factor de escala en ecuación del controlador para línea recta (SLI y SLW)
β	0.6	Desplazamiento en ecuación del controlador para línea recta (SLI y SLW)
k_1	–	Ganancia adaptable para controlador de velocidad lineal aplicada al error en distancia en $SLID$
k_2	0.2	Ganancia para controlador de velocidad angular aplicada al error en distancia en SLW y $SLWD$
k_3	-0.5	Ganancia para controlador de velocidad angular aplicada al error en θ_1 en SLW y $SLWD$
k_4	–	Ganancia adaptable para controlador de velocidad lineal aplicada al error en distancia respecto al obstáculo (esquina convexa o cóncava) en $SLWD$
k_5	-4	Ganancia para controlador de velocidad angular aplicado al error angular en RP
k_6	0.01	Ganancia para controlador de velocidad angular aplicado al error en distancia respecto a la esquina convexa en AC
k_7	0.01	Ganancia para controlador de velocidad angular aplicado a la derivada del error en distancia respecto a la esquina convexa en AC
k_t	–	Ganancia adaptable para el control de velocidad lineal en AC
τ_1	1 seg.	Parámetro del tiempo para ejecutar el perfil en $SLWD$
τ_2	1 seg.	Parámetro de tiempo en el perfil que permite el movimiento en RP
τ_3	1 seg.	Parámetro de tiempo en el perfil que permite alcanzar una ganancia máxima de k_t en AC

Figura B.1: Parámetros de Implementación para el Robot Pioneer P3-DX bajo el esquema planteado en el Capítulo 6.

B.2. Parámetros de Implementación para el Esquema que utiliza un solo Controlador *Super-Twisting*, la Máquina de Estados Conmuta entre los Valores de Referencia y la Velocidad Lineal es Constante

Esquema donde se usa un solo controlador *Super-Twisting* y la máquina de estados conmuta entre los valores de referencia (Capítulo 8)		
Parámetro	Valor de Implementación	Descripción
b	0.2 m	Radio del disco que representa al robot real
d_d	0.4 m	Radio de robot virtual (círculo rojo)
d_t	0.7 m	Radio de círculo que define la trayectoria de arco de círculo a ejecutar sobre esquina cóncava
V_d	0.3 m/s	Velocidad lineal deseada
α	3.8293	Factor de escala en ecuación del controlador de velocidad lineal
β	0.6	Desplazamiento en ecuación del controlador de velocidad lineal (SLI y SLW)
ϵ_1	0.1	Tolerancia para definir rp'
ϵ_2	10°	Tolerancia para determinar la alineación con la pared de acuerdo al la medición de los ángulos θ_1, θ_2, θ_3 y θ_4
k_1	2.8	Ganancia aplicada directamente al error en distancia (este termino forma parte de la superficie deslizante s)
k_2	-2.1	Ganan aplicada directamente al error angular (este termino forma parte de la superficie deslizante s)
k_3	0.04	Ganancia aplicada en $\dot{\sigma}$ del controlador *Super-Twisting*
k_4	0.8	Ganancia aplicada en la velocidad angular ω_s del controlador *Super-Twisting*
τ	1 seg.	Parámetro del tiempo para ejecutar el perfil que permite la transición suave entre los valores de referencia

Figura B.2: Parámetros de Implementación para el Robot Pioneer P3-DX bajo el esquema planteado en el Capítulo 8.

B.3. Parámetros de Implementación para el Esquema que utiliza un solo Controlador *Super-Twisting*, la Máquina de Estados Conmuta entre los Valores de Referencia y la Velocidad Lineal se Reduce al Aproximarse a las Esquinas

Esquema donde se usa un solo controlador *Super-Twisting*, la máquina de estados conmuta entre los valores de referencia y se reduce la velocidad lineal (Sección 8.5)		
Parámetro	Valor de Implementación	Descripción
b	0.2 m	Radio del disco que representa al robot real
d_d	0.4 m	Radio de robot virtual (círculo rojo)
d_s	0.96 m	Radio de círculo para detectar un obstáculo cercano y reducir la velocidad lineal del robot
d_t	0.6 m	Radio de círculo que define la trayectoria de arco de círculo a ejecutar sobre esquina cóncava
v_n	0.5 m/s	Velocidad nominal
v_{corner}	0.25	Velocidad lineal durante la ejecución de arcos de círculo en las esquinas
α	3.8293	Factor de escala en ecuación del controlador de velocidad lineal
β	0.6	Desplazamiento en ecuación del controlador de velocidad lineal (SLI y SLW)
ϵ_1	0.1	Tolerancia para definir rp'
ϵ_2	10°	Tolerancia para determinar la alineación con la pared de acuerdo al la medición de los ángulos θ_1, θ_2, θ_3 y θ_4
k_1	2	Ganancia aplicada directamente al error en distancia (este termino forma parte de la superficie deslizante s)
k_2	-2.8	Ganan aplicada directamente al error angular (este termino forma parte de la superficie deslizante s)
k_3	0.06	Ganancia aplicada en $\dot{\sigma}$ del controlador *Super-Twisting*
k_4	0.8	Ganancia aplicada en la velocidad angular ω_s del controlador *Super-Twisting*
k_5	–	Ganancia adaptable para controlador de velocidad lineal aplicada al error en distancia respecto al obstáculo detectado (esquina convexa o cóncava)
τ	1 seg.	Parámetro del tiempo para ejecutar el perfil que permite la transición suave entre los valores de referencia

Figura B.3: Parámetros de Implementación para el Robot Pioneer P3-DX bajo el esquema planteado en la Sección 8.5.

Bibliografía

[1] G. Laguna, R. Murrieta-Cid, H.M. Becerra, R. Lopez-Padilla, and S.M. LaValle. Exploration of an unknown environment with a differential drive disc robot. In *Proc of IEEE Int. Conf. on Robotics and Automation*, pages 2527–2533, 2014.

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

[3] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

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

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

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

[7] J.-B.. Hayet, H. Carlos, C. Esteves, and R. Murrieta-Cid. Motion planning for maintaining landmarks visibility with a differential drive robot. *Robotics and Autonomous Systems*, 4(62):456–473, 2014.

[8] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8(1-2):47–63, 1991.

[9] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, Monterey, CA, USA, 1997.

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

[11] 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.

[12] M. Juliá, A. Gil, and O. Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.

[13] 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, 2007.

[14] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments. *Advanced Robotics*, 23(12-13):1533–1560, 2009.

[15] M. Katsev, A. Yershova, B. Tovar, R. Ghrist, and S. M. LaValle. Mapping and pursuit-evasion strategies for a simple wall-following robot. *IEEE Transactions on Robotics*, 27(1):113–128, 2011.

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

[17] L. Murphy and P. Newman. Using incomplete online metric maps for topological exploration with the gap navigation tree. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 2792–2797, 2008.

[18] C.J. Taylor and D. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. *IEEE Transactions on Robotics and Automation*, 14(3):417–426, 1998.

[19] A. Elfes. Sonar-based real world mapping and navigation. *IEEE Transactions on Robotics and Automation*, 3(3):249–264, 1987.

[20] 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 1986*, pages 1664–1669, San Francisco, California, USA, 1986.

[21] R. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proceedings of the International Conference on Robotics and Automation, ICRA 1990*, pages 566–571, Cincinnati, Ohio, USA, 1990.

[22] Maher Khatib. Sensor-based motion control for mobile robots. *Maher Khatib Sensor-based motion control for mobile robots PHD thesis, LAAS-CNRS December*, 1996.

[23] L. Montano and J. R. Asensio. Real-time robot navigation in unstructured environments using a 3d laser rangefinder. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 1997*, pages 526–532, Grenoble, France, 1997.

[24] K. Azarm and G. Schmidt. Integrated mobile robot motion planning and execution in changing indoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 1994*, pages 298–305, Munich, Germany, 1994.

[25] S. A. Masoud A. A. Masoud and M. M. Bayoumi. Robot navigation using a pressure generated mechanical stress field, the biharmonic potential approach. In *Proceedings of the International Conference on Robotics and Automation, ICRA 1994*, pages 124–130, Los Alamitos, CA, USA, 1994.

[26] R. Bauer W. Feiten and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. In *Proceedings of the International Conference on Robotics and Automation, ICRA 1994*, pages 2412–2417, Los Alamitos, CA, USA, 1994.

[27] J. Borenstein and Y. Koren. The vector field histogram: Fast obstacle avoidance for mobile robots. *IEEE Trans. Robotics and Automation*, 7(3):278–288, 1991.

[28] I. Ulrich and J. Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 1998*, pages 1572–1577, Leuven, Belgium, 1998.

[29] C. E. Thorpe M. Hebert and A. Stentz. Intelligent unmanned ground vehicles: Autonomous navigation research at carnegie mellon, 1996.

[30] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 1996*, pages 3375–3382, Minneapolis, Minnesota, USA, 1996.

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

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

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

[34] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation, ICRA 1993*, page 802–807, Atlanta, GE,USA, 1993.

[35] O. Brock and O. Khatib. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *n In IEEE Int. Conf. on Robotics and Automation, ICRA 2000*, pages 550–555, 2000.

[36] D. Sun V. Isler and S. Sastry. Roadmap based pursuit-evasion and collision avoidance. *Robotics: Science and Systems*, page 257–264, 2005.

[37] Philipp Michelt, Christian Scheurertt, James Kuffnert, Nikolaus Vahrenkampt, and Rudiger Dillmannt. Planning for robust execution of humanoid motions using future perceptive capability. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007*, pages 3223–3228, 2007.

[38] P. Souères and J. Laumond. Shortest paths synthesis for a car-like robot. *IEEE Transaction on Automatic Control*, 41(5):672–688, 1996.

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

[40] 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.

[41] Anthony Lazanas and Jean claude Latombe. Landmark-based robot navigation. In *Algorithmica*, pages 816–822, 1992.

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

[43] Rafael Murrieta-Cid, Carlos Parra, and Michel Devy. Visual navigation in natural environments: from range and color data to a landmark-based model. *Autonomous Robots*, 13(2):143–168, 2002.

[44] Parthasarathy Ranganathan, Jean-Bernard Hayet, Michel Devy, Seth Hutchinson, and Frédéric Lerasle. Topological navigation and qualitative localization for indoor environment using multi-sensory perception. *Robotics and Autonomous Systems*, 41(2-3):137–144, 2002.

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

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

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

[48] 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.

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

[50] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 1998*, pages 1060–1065, 1998.

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

[52] Sourabh Bhattacharya, Rafael Murrieta-Cid, and Seth 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.

[53] Gonzalo López-Nicolás, Nicholas R Gans, Sourabh Bhattacharya, Carlos Sagues, Josechu J Guerrero, and Seth 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.

[54] Jean-Bernard Hayet, Claudia Esteves, and Rafael Murrieta-Cid. A motion planner for maintaining landmark visibility with a differential drive robot. In *Algorithmic Foundation of Robotics VIII*, pages 333–347. Springer, 2009.

[55] S. M. LaValle. Sensing and filtering: A fresh perspective based on preimages and information spaces. In *Foundations and Trends in Robotics Series*. Now Publishers, Delft, The Netherlands, 2012.

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

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

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

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

[60] R. Lopez-Padilla, R. Murrieta-Cid, and S.M. LaValle. Optimal gap navigation for a disc robot. In E. Frazzoli et al., editor, *Proc. of the Tenth Workshop on the Algorithmic Foundations of Robotics: Springer Tracts in Advanced Robotics*, pages 123–138. Springer, 2013.

[61] R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005*, pages 661–666, Barcelona, Spain, 2005.

[62] H. Feder, J. Leonard, and C. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, 18(7):650–668, 1999.

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

[64] Y.-A.. Gidhar and G. Dudek. Modeling curiosity in a mobile robot for long-term autonomous exploration and monitoring. *Autonomous Robots*, 40(7):1267–1278, 2016.

[65] R. Sim and G. Dudek. Effective exploration strategies for the construction of visual maps. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2003*, pages 3224–3231, Las Vegas, Nevada, USA, 2003.

[66] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso. The srt method: randomized strategies for exploration. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2004*, pages 4688–4694 Vol.5, New Orleans, LA, USA, 2004.

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

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

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

[70] F. Amigoni, S. Gasparini, and M. Gini. Building segment-based maps without pose information. *Proceedings of the IEEE*, 94(7):1340–1359, 2006.

[71] A. Kolling and S. Carpin. Extracting surveillance graphs from robot maps. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 11–19, 2008.

[72] J. Neira J. Castellanos, J. Montiel and J. Tard' s. A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–953, 1999.

[73] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *Transactions on Robotics and Automation*, 2001.

[74] D. Koller M. Montemerlo, S. Thrun and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *In Proc. of the National Conference on Artificial Intelligence, AAAI 2002*, 2002.

[75] M. Self R. Smith and P. Cheeseman. Estimating uncertain spatial relationships inrobotics. *Autonomous Robot Vehicles*, 1997.

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

[77] C.J. Taylor and D. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. *IEEE Transactions on Robotics and Automation*, 14(3):417–426, 1998.

[78] L. Murphy and P. Newman. Using incomplete online metric maps for topological exploration with the gap navigation tree. In *In Proc. IEEE Int. Conf. Robotics and Automation, ICRA 2008*, pages 2792–2797, 2008.

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

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

[81] Benjamín Tovar, Lourdes Muñoz-Gómez, Rafael Murrieta-Cid, Moisés Alencastre-Miranda, Raúl Monroy, and Seth Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Journal on Robotics and Autonomous Systems*, 54(4):314–331, 2006.

[82] Héctor González-Baños, Eric Mao, Jean-Claude Latombe, TM Murali, and Alon Efrat. Planning robot motion strategies for efficient model construction. In *In Robotics Research - The 9th International Symposium*, page 345–352, 1999.

[83] 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.

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

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

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

[87] L. Iocchi D. Calisi, A. Farinelli and D. Nardi. Multi-objective exploration and search for autonomous rescue robots. *J. Field Robot*, 24(8-9):763–777, 2007.

[88] E. Bicho. Detecting, representing and following walls based on low-level distance sensors. In *Proc. of the Int. Symposium on Neural Computation*, Berlin, Germany, 2000.

[89] M. Toibero, F. Roberti, and R. Carelli. Stable contour-following control of wheeled mobile robots. *Robotica*, 27(1):1–12, 2009.

[90] M. Toibero, F. Roberti, R. Carelli, and P. Fiorini. Switching control approach for stable navigation of mobile robots in unknown environments. *Robotics and Computer-Integrated Manufacturing*, 27(2):558–568, 2011.

[91] Avik De and Daniel E Koditschek. Toward dynamical sensor management for reactive wall-following. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2400–2406. IEEE, 2013.

[92] A. G. Lamperski, O. Y. Loh, B. L. Kutscher, and N. J. Cowan. Dynamical wall following for a wheeled robot using a passive tactile sensor. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005*, pages 3838–3843, Barcelona, Spain, 2005.

[93] Kamilah Taylor and Steven M LaValle. Intensity-based navigation with global guarantees. *Autonomous Robots*, 36(4):349–364, 2014.

[94] Tzu-Chao Lin, Chao-Chun Chen, and Cheng-Jian Lin. Wall-following and navigation control of mobile robot using reinforcement learning based on dynamic group artificial bee colony. *Journal of Intelligent & Robotic Systems*, pages 1–15, 2017.

[95] Jyun-Yu Jhang, Cheng-Jian Lin, Chin-Teng Lin, and Kuu-Young Young. Navigation control of mobile robots using an interval type-2 fuzzy controller based on dynamic-group particle swarm optimization. *International Journal of Control, Automation and Systems*, pages 1–12, 2018.

[96] R. Lopez-Padilla, R. Murrieta-Cid, I. Becerra, G. Laguna, and S. M. LaValle. Optimal navigation for a differential drive disc robot: A game against the polygonal environ-

ment. *Fully accepted to J. of Intelligent Robotic Systems, Available at the Journal Web page. DOI https://doi.org/10.1007/s10846-016-0433-1*, pages 1–40, 2016.

[97] L Fridman and A Levant. Higher order sliding modes. sliding mode control in engineering, sliding mode control in engineering, jp barbot, w. perruquetti, 2002.

[98] Jaime A Moreno and Marisol Osorio. Strict lyapunov functions for the super-twisting algorithm. *IEEE transactions on automatic control*, 57(4):1035–1040, 2012.

[99] Vadim Utkin. On convergence time and disturbance rejection of super-twisting control. *IEEE Transactions on Automatic Control*, 58(8), 2013.

[100] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Pearson Education, 2000.

[101] W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1994.

[102] F. Lamiraux J. Minguez and J. P. Laumond. Motion planning and obstacle avoidance. *In Siciliano B and Khatib O (eds.), Springer Handbook of Robotics*, pages 827–852, 2008.

[103] T. Fraichard and J. Kuffner Jr. Guaranteeing motion safety for robots. *Auton. Robots*, pages 173–175, 2012.

[104] Andrea Cherubini and François Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. *The International Journal of Robotics Research*, 32(2):189–205, 2013.

[105] R. Reyes and R. Murrieta-Cid. An approach integrating planning and image based visual servo control for road following and moving obstacles avoidance. *Fully accepted to International Journal of Control*, 2018.

[106] Edgar Martinez, Guillermo Laguna, Rafael Murrieta-Cid, Hector M. Becerra, Rigoberto Lopez-Padilla, and Steven M. LaValle. A motion strategy for exploration driven by an automaton activating feedback-based controllers. *Fully accepted to Autonomous Robots*, 2019.

[107] Edgar Martinez, Rafael Murrieta-Cid, and Hector M. Becerra. An automaton and super-twisting sliding-mode control for wall following. *to be submited to a Journal*, 2019.