Benemérita
Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"Planificación de movimientos en ambientes dinámicos utilizando un enfoque lazy PRM reactivo"

TESIS
PROFESIONAL

QUE PARA OBTENER EL GRADO DE:

MA**ESTRO EN CIENCIAS DE LA COMPUTACIÓN**

PRESENTA

JOSÉ RODRIGO CUAUTLE PARODI

ASESOR

Dr. ABRAHAM SÁNCHEZ LÓPEZ

PUEBLA PUE. 2005 OCTUBRE 2005

Índice general

Introducción

La planificación de movimientos dentro de un medio completamente estructurado puede ser considerada como un problema resuelto (dentro de ciertos ejemplos voluntariamente simplificados de robots que evolucionan dentro de ambientes perfectamente controlados), o como un problema académico.

Sólo los casos de ambientes no estructurados presentan entonces un interés realista dentro del marco de la robótica móvil. De igual manera, el dinamismo del ambiente es una hipótesis que se debe tomar en cuenta.

Estas hipótesis de débil estructuración y de dinamismo han sido establecidas anteriormente. Resulta que el factor a optimizar dentro de la realización de un robot es la reactividad de cara a los eventos imprevistos.

De esta forma el **objetivo general** del presente trabajo es aplicar la teoría de las acciones reflejas en el marco de trabajo de los roadmaps probabilistas *(PRM)* [1], [2].

Así también se plantean los siguientes **objetivos específicos:**

1. Desarrollar un método local basado en la *zona virtual deformable* [7], [8].

2. Desarrollar un ambiente de trabajo (software) para probar los diferentes algoritmos desarrollados.

El Capítulo 1 está dedicado a los antecedentes del proyecto, el Capítulo 2 estudia el método de planificación probabilista *lazy PRM* [3] , el Capítulo 3 estudia el método reactivo de la *zona virtual deformable (ZVD)*, el Capítulo 4 contiene la propuesta realizada, en el Capítulo 5 se presentan los resultados obtenidos y por último el Capítulo 6 se dedica a las conclusiones y al trabajo futuro.

Capítulo 1

Antecedentes

1.1. Introducción

Los algoritmos para generar movimiento en los robots han sido históricamente divididos en algoritmos de planificación y algoritmos de control reactivo.

Los algoritmos de planificación son considerados para generar caminos en forma global con respecto al ambiente donde se desenvuelve el robot , mientras que los algoritmos de control sólo usan información local para determinar los controles de movimiento. Además, los primeros calculan el camino antes de la ejecución del movimiento, mientras los segundos determinan los comandos de movimiento mediante retroalimentación durante la ejecución del mismo.

1.2. Métodos de planificación de movimientos

Los enfoques actuales de planificación de movimientos son principalmente de tres tipos: métodos de roadmaps [1], [2], métodos de descomposición en celdas [9] y métodos de campos potenciales [10].

La planificación de movimientos generalmente no se lleva a cabo en el espacio de trabajo del robot, sino en el espacio de configuraciones (CS) en donde el robot se representa por un punto en un espacio n-dimensional , donde n corresponde al número de grados de libertad del robot.

1.2.1. Campos Potenciales

En este método el robot representado como un punto en el espacio de configuraciones es una partícula cuyo movimiento está influenciado por una fuerza

producida por la configuración final y los obstáculos. La configuración final genera un potencial de atracción que lleva al robot hacia la meta y los obstáculos producen un potencial de repulsión que aleja al robot de ellos.

El método es eficiente sin embargo tiene la desventaja de que la función potencial puede caer en un mínimo local, es decir la fuerza de atracción y repulsión en algún punto es de igual magnitud provocando que el robot no se mueva.

1.2.2. Descomposición en celdas

Este método es uno de los más estudiados hasta ahora, consiste en descomponer el espacio libre en regiones simples llamadas celdas, un grafo no dirigido representa la relación de adyacencia entre las celdas y se le llama grafo de conectividad. Los nodos son celdas extraídas del espacio libre y están conectadas por un arista si y solo si las dos celdas correspondientes son adyacentes. El resultado de la búsqueda en el grafo es una secuencia de celdas llamada *canal*, un camino libre de colisiónes se calcula a partir de estas secuencias.

El método consta de dos enfoques, el exacto y el aproximado. El método exacto divide el espacio libre en celdas cuya unión corresponde exactamente al espacio libre, como se muestra en la figura 1.1.

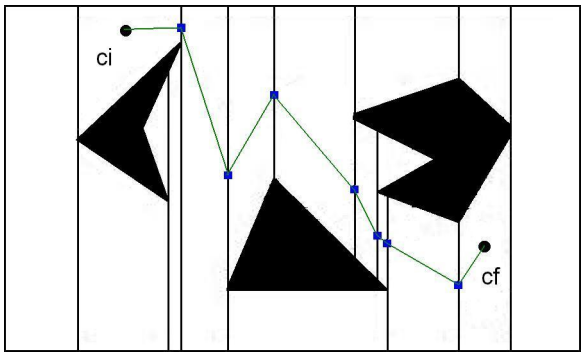

Figura 1.1: El camino encontrado se indica con una línea que toca los puntos medios de cada una de las aristas de las celdas comprendidas dentro del espacio libre.

El enfoque aproximado produce celdas de una forma predefinida, generalmente rectángulos, cuya unión no corresponde estrictamente al espacio libre, tal como se muestra en la figura 1.2. El rectángulo que limita el espacio de trabajo es recursivamente descompuesto en rectángulos más pequeños.

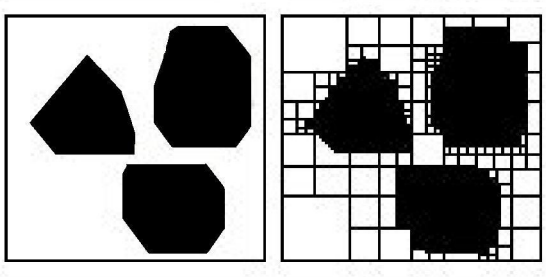

Figura 1.2: Método aproximado de descomposición de celdas (quadtree).

Las celdas que se encuentran completamente en el espacio libre se usan para construir el grafo de conectividad, si la búsqueda en el grafo encuentra una secuencia de celdas que representen un camino en el espacio libre el método reportará éxito, en caso contrario significará que la resolución es insuficiente o bien que no existe una solución al problema.

1.2.3. Roadmaps

En este método se utiliza un grafo denominado roadmap en el cuál los nodos son puntos que pertenecen al espacio de configuraciones libre (CS_{libre}), y que se conectan entre sí por medio de algún método local, cada camino local también pertenece a CS_{libre}. Una vez construído el roadmap, se intentan conectar las configuraciones inicial y final, para finalmente buscar un camino entre estos dos puntos.

Se han propuesto muchas variantes, es decir la construcción de roadmaps con distintas características, una de las más destacadas es el método de roadmaps probabilístico *PRM* (Figura 1.3), en el cuál los nodos que forman el roadmap son generados en forma aleatoria. El método PRM también presenta variantes, una de estas se denomina *Lazy PRM* [17], y se distingue porque cuando se genera una nueva configuración aleatoria para un nodo del roadmap no se comprueba si pertenece al espacio libre CS_{libre} hasta que forma parte de un probable camino solución.

1.3. Métodos de control reactivo

Los métodos de planificación de movimiento descritos anteriormente asumen que el ambiente es conocido y estático. El control reactivo integra información sensorial al proceso de generación de movimientos para poder reaccionar ante eventos imprevistos.

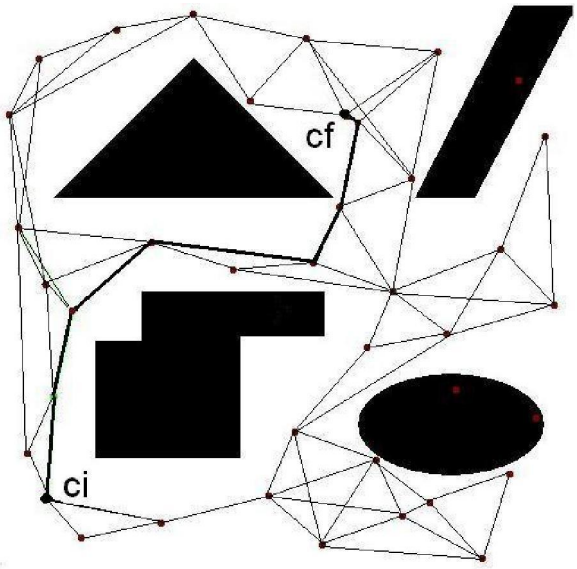

Figura 1.3: Roadmap construido mediante *PRM*. Las líneas gruesas muestran el camino que lleva al robot móvil de la configuración inicial c_i a la final c_f.

1.3.1. Campos potenciales

Descrito como método de planificación de movimientos en la sección anterior, puede también dotar de reactividad al robot si se genera el potencial de repulsión ocasionado por los obstáculos a través de información sensorial. El potencial atractivo es generado por el vector distancia hacia la configuración objetivo. El problema de mínimos locales permanece.

1.3.2. Seguimiento de frontera

Este esquema de control reactivo es inmune a mínimos locales. El robot se mueve en línea recta hacia su objetivo, si un obstáculo bloquea el camino el robot sigue el contorno de este, recordando el punto más cercano a su objetivo [11]. A partir de este punto el movimiento se continúa nuevamente en línea recta (Figura 1.4).

1.3.3. Zona virtual deformable

Bajo este método [7], [8] se asume que el robot está envuelto por una zona virtual, cuya geometría depende de la cinemática del robot, esta zona virtual puede sufrir deformaciones debidas a la intrusiones ocasionadas por obstáculos

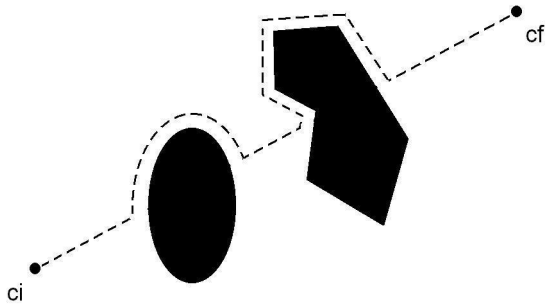

Figura 1.4: Seguimiento de frontera.

próximos. Este método consiste en minimizar las deformaciones que se presenten al modificar el vector de control del robot (aceleración, orientación,...). Por lo tanto este método de control puede ser visto como un juego entre dos contrincantes: el ambiente trata de maximizar las deformaciones y el módulo de control del robot trata de minimizarlas (Figura 1.5). En el capítulo 3, se hará una descripción más detallada de la zona virtual deformable.

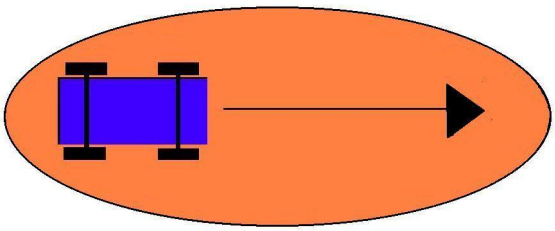

Figura 1.5: La zona virtual envuelve al robot para protegerlo contra colisiones.

1.4. Integrando planificación y ejecución

La integración de planificación y ejecución de movimientos ha sido aplicada para mejorar o extender el desempeño de los algoritmos de generación de movimientos. Brock presenta varias arquitecturas de sistemas para la generación de movimientos del robot [6]. Estas se ilustran en las figuras 1.6, 1.7, 1.8, y 1.9.

El esquema de un método puramente reactivo se muestra en la figura 1.6. El control usa información obtenida a través de sensores. No existe información de conectividad global acerca del espacio libre, causando que el enfoque sea sus-

ceptible a mínimos locales.

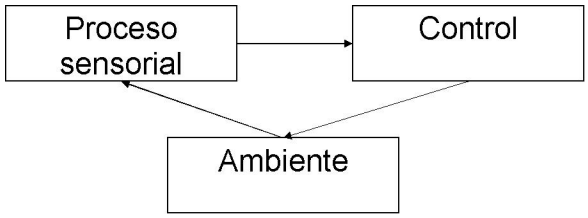

Figura 1.6: Arquitectura para una ejecución de movimientos puramente reactiva.

Por otra parte los métodos de planificación (Figura 1.7), eliminan los problemas de mínimos locales, al usar un modelo del ambiente. Se asume que el ambiente permanece sin modificaciones o que estas son conocidas, garantizando por lo tanto que el camino calculado lleve sin problemas a la configuración objetivo. Sin embargo estas condiciones no son reales en la mayoría de los casos.

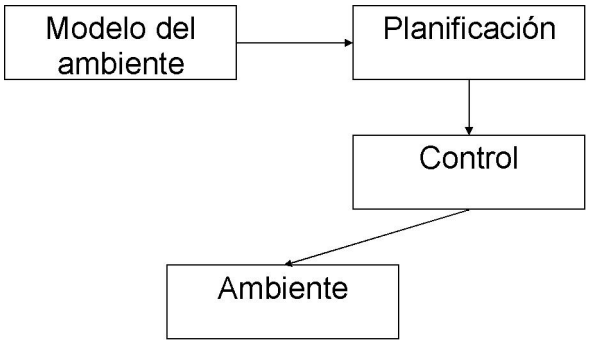

Figura 1.7: Arquitectura de sólo planificación de movimientos.

Los esquemas presentados pueden ser combinados, el resultado se ilustra en la figura 1.8. Aquí una vez que se ha planificado se ejecuta el camino obtenido en forma reactiva. Sin embargo, aunque los mínimos locales presentes al momento de planificar pueden ser evitados, no será el caso de aquellos que se pudieran crear por cambios en el ambiente durante la ejecución, ya que la información sensorial no actualiza el modelo del ambiente para generar un nuevo plan.

Por lo tanto la arquitectura ideal para el sistema se compone de un lazo que comprende planificación, ejecución de movimiento, adquisición de información por sensores, y actualización del modelo del ambiente. Esta idea se ilustra en la figura 1.9.

Si bien en teoría esta es la arquitectura ideal, no es aplicable en la mayoría de

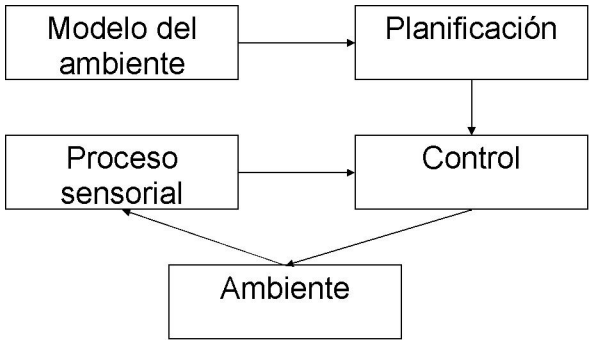

Figura 1.8: Arquitectura de planificación de movimientos con ejecución reactiva.

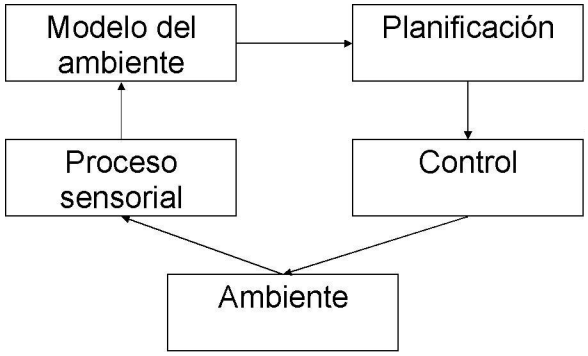

Figura 1.9: Arquitectura ideal, pero de alto costo computacional.

los problemas prácticos. Desafortunadamente la planificación es una operación computacionalmente cara, el reto es por lo tanto lograr métodos que integren las ventajas de la planificación y del control reactivo de movimientos del robot, para que éste sea capaz de desenvolverse en ambientes reales.

En los últimos años se vienen presentando diversas propuestas que atacan este problema: planificadores basados en métodos PRM [13], técnicas para replanificación en tiempo real usando el método de la banda elástica [14], y técnicas explorando el espacio tiempo×configuración para obstáculos móviles a lo largo de caminos conocidos [15].

La propuesta realizada en esta tesis comprende integrar el método lazy PRM de planificación de movimientos y el de control reactivo de la zona virtual deformable, en un planificador de movimientos para robots móviles no holonómicos (tipo carro), que se desenvuelvan en ambientes dinámicos. Esta integración de métodos será implementada mediante software para poder evaluar los resultados obtenidos.

Capítulo 2

Lazy PRM

2.1. Introducción

Lazy PRM [17] es similar al método básico PRM en el hecho de generar las configuraciones que constituyen los nodos del roadmap en forma aleatoria, sin embargo en contraste con el método clásico, lazy PRM no construye un roadmap de caminos factibles, sino que asume que lo son y sólo comprobará su factibilidad en caso de ser requeridos. Es decir, al momento de realizar la búsqueda en el grafo. En [3], [16] y [18] se propuso por primera vez una adaptación del algoritmo lazy PRM a la planificación no holonómica. El algoritmo original se propuso para resolver sólo problemas holonómicos.

Sean c_i y c_f las configuraciones inicial y final respectivamente, y un número de configuraciones uniformemente distribuidas que constituyen los nodos del roadmap. Cada par de nodos que estén suficientemente cerca se conectan. Lazy PRM encuentra el camino factible más corto en el roadmap, al revisar y eliminar repetidamente los nodos y arcos entre nodos que estén en colisión dentro de cada probable camino solución que se genera en cada iteración. Cada vez que una colisión ocurre , el nodo o arco correspondiente es eliminado y lazy PRM busca un nuevo camino más corto.

El procedimiento puede terminar de dos formas. Si existe un camino factible en el roadmap generado entre c_i y c_f, este será encontrado. De otra forma se reportará fallo, o si aún queda tiempo es posible agregar más nodos al roadmap y buscar nuevamente. Una descripción de alto nivel se muestra en la figura 2.1.

El punto a favor del método lazy, es que sólo se comprobará la factibilidad de los nodos y arcos necesarios, y no del roadmap completo, esto es una ventaja porque nunca se realizará más trabajo en cuanto a revisión de colisiones de lo que PRM clásico haría.

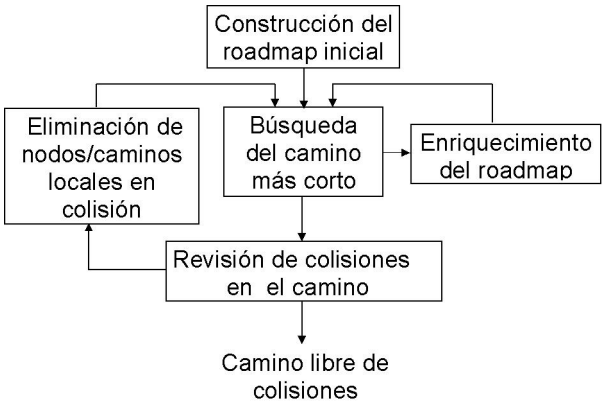

Figura 2.1: Descripción de alto nivel del algoritmo lazy PRM.

2.2. Construcción del roadmap inicial

El primer paso en el algoritmo es construir un roadmap G en el espacio de configuraciones CS. Existen dos parámetros que determinan el tamaño de G: el número de nodos N, y el número de vecinos V_c conectado a cada nodo.

Inicialmente se distribuyen N puntos uniformemente en forma aleatoria en el espacio de configuraciones CS. Estos puntos, junto con $c_i \in CS_{libre}$, y $C_f \in CS_{libre}$, constituyen los nodos de G. El procedimiento es simple. Se contruye el grafo $G=(V,E)$, al generar repetidamente una configuración c en forma aleatoria, se añade dicha configuración a V, calculando un conjunto $V_c \in V$ (k-vecinos), y añadiendo un arco (c,n) a E para todo $n \in V_c$,cuyo método local puede conectar desde c. El procedimiento puede escribirse de la siguiente manera:

$V \leftarrow c_i, c_f$
$E \leftarrow \emptyset$
mientras $n_{nodos} < N$ *hacer*
  $c \leftarrow configuración\ aleatoria$
  si $c \in CS$ *entonces*
    $V \leftarrow V \bigcup \{c\}$
    $V_c \leftarrow$ *un conjunto de vecinos de c seleccionados de* V
    $\forall v \in V_c$, *en orden ascendente de* $d(v,c)$ *hacer*
      si$\neg$*conectado* $(c,v) \wedge L(c,v) \subset CS$ *entonces*
        $E \leftarrow E \bigcup \{(c,v)\}$
fin

2.3. Búsqueda del camino más corto

El segundo procedimiento en el algoritmo es encontrar el camino más corto en G entre c_i y c_f, o determinar que no existe alguno. Para este propósito se puede utilizar el algoritmo A* [4]. Si la búsqueda tiene éxito, se revisa entonces que se encuentre libre de colisiones. Si no existe camino alguno entre c_i y c_f, se reporta fallo o bien se llama a un procedimiento de enriquecimiento del roadmap, el cuál agrega nuevos nodos, teniendo en cuenta el tiempo permitido para resolver el problema.

2.4. Revisión del camino encontrado

Para asegurar que el camino calculado por el algoritmo A* entre c_i y c_f es factible, se revisa que los nodos y arcos que lo forman estén libres de colisión (Figura 2.2). Ya que la eliminación de un nodo implica también la de sus arcos, es razonable comprobar primero la factibilidad de estos antes que la de los arcos del camino.

Revisión de nodos. Se inicia revisando a los nodos extremos del camino calculado, para luego avanzar alternativamente hacia el centro. Tan pronto como se detecte un nodo en colisión, este se elimina con todo y sus arcos, y se busca nuevamente un camino más corto.

Revisión de arcos. Si todos los nodos a lo largo del camino están en CS_{libre}, se revisa entonces la factibilidad de cada camino local entre nodos. Sin embargo, para minimizar el número de llamadas al detector de colisiones, primero se revisan todos los arcos del camino con poca resolución, afinando luego, en varias pasadas la resolución utilizada hasta llegar a la deseada. Como en el caso de los nodos, si una trayectoria local está en colisión, se elimina el arco correspondiente y se busca nuevamente el camino más corto. Si ninguna colisión es detectada a lo largo del camino, el algoritmo termina y retorna como resultado al camino libre de colisiones.

Prueba de colisión para un robot tipo carro

Dado un camino local, debemos probar si este está en colisión. Esta operación no puede efectuarse de manera analítica sobre el camino. La estrategia utilizada consiste en probar la no colisión de un número finito de configuraciones sobre el camino, asegurando que el intervalo entre cada punto esta igualmente sin colisión.

La prueba de colisión se basa en las llamadas repetidas al detector de colisiones, según un paso de discretización del camino no uniforme [16]. La figura 2.3 describe este mecanismo de validación del camino. Dado un camino admisible λ entre dos configuraciones c_i y c_f, se coloca el objeto móvil en su primera

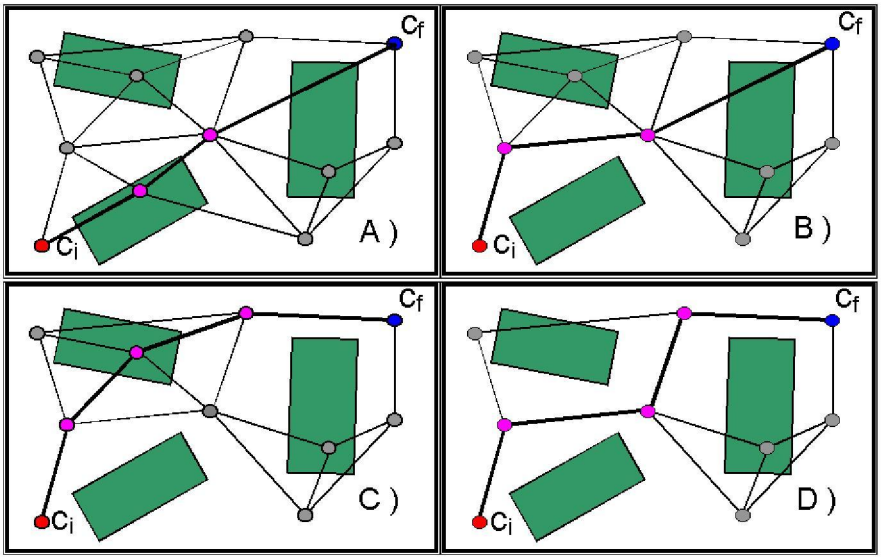

Figura 2.2: Lazy PRM revisa colisiones sólo en caminos candidatos a solución. En la parte A) se detecta un nodo en colisión, el cuál se elimina con todo y arcos. En la figura B) no existen nodos en colisión, pero si un camino local. En C) se tiene un nuevo camino candidato a solución, pero presenta un nodo en colisión; finalmente en D) se obtiene un camino completo libre de colisiones.

configuración actual $c_c = c_i$ y se calcula a que distancia d_{min} se encuentra el obstáculo más próximo (con la ayuda de las cajas englobantes). A partir de la estructura del objeto móvil, se puede calcular una cota superior de la distancia recorrida por sus puntos sobre un incremento del camino d_c. Se puede entonces calcular para d_{min}, que incremento d_c se puede avanzar sobre λ estando seguro de no chocar con el obstáculo. Se coloca el objeto móvil en la configuración $c_c = c_c + d_c$ y se repite el mismo procedimiento. De esta manera, se va a recorrer λ con un mínimo de llamadas al detector de colisiones.

Es claro que a medida que el objeto se acerque a un obstáculo durante el recorrido de su camino, el paso d_c será más pequeño.

Cuando un obstáculo se encuentre sobre el camino, o bien este próximo de este, el objeto móvil va a recorrer el camino hasta que su caja englobante este en contacto con la del obstáculo. El paso de discretización será entonces uniforme y pequeño, hasta que el objeto esté en colisión con el obstáculo, o bien termina por alejarse (Figura 2.4).

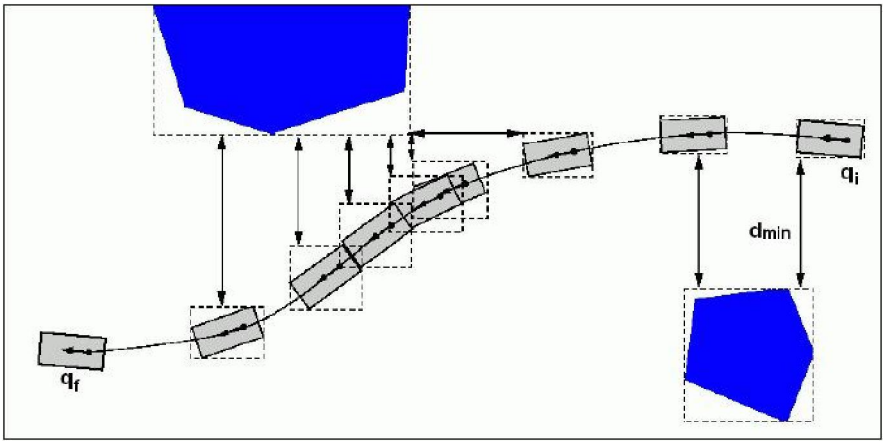

Figura 2.3: Validación de un camino libre de obstáculos.

2.5. Enriquecimiento del roadmap

Si el procedimiento de búsqueda falla, entonces no existe camino factible entre c_i y c_f, y es necesario agregar más nodos al roadmap.

Es posible que durante los pasos anteriores se llegue a un roadmap disconexo, o bien que durante la construcción inicial del mismo hayan quedado porciones aisladas, pues bien, al enriquecer el roadmap estas partes aisladas tienen posibilidad de integrarse y obtener tal vez entonces un camino solución, como se ejemplifica en la figura 2.5. En este trabajo no se discuten con detalle estas técnicas, en [1] y [2] se presentan algunas técnicas para este propósito.

2.6. Método local

Cada arco entre nodos corresponde a un camino local entre ellos. Las posibilidades en la forma de cada camino local están condicionadas por las restricciones cinemáticas del robot (Figura 2.6); para el caso de un robot tipo carro se emplea a menudo el método Reeds & Shepp [5].

La curvas de Reeds&Shepp proporcionan una familia de 48 caminos elementales para un robot tipo carro que puede moverse hacia el frente y en reversa. Los caminos óptimos están constituidos por una secuencia finita de a lo más cinco caminos elementales, los cuáles pueden ser segmentos de línea recta o bien arcos de circunferencia. La tabla 2.1, muestra la familia de curvas Reeds & Shepp.

Las figuras 2.7 y 2.8 muestran ejemplos de curvas Reeds&Shepp.

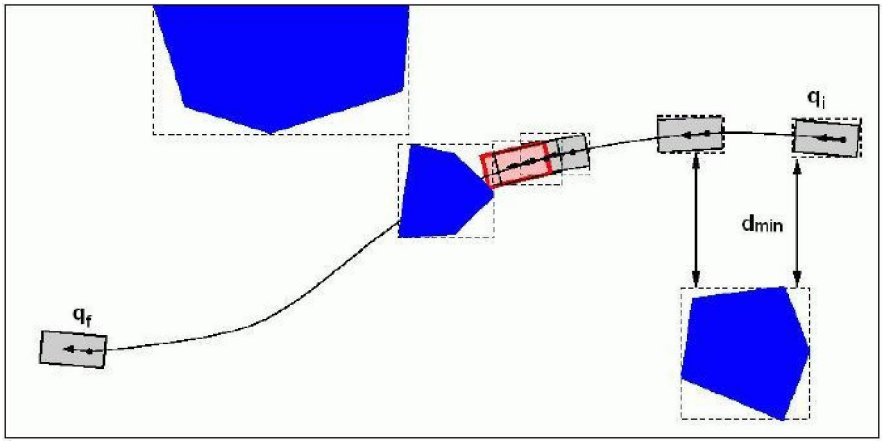

Figura 2.4: Validación de un camino en presencia de un obstáculo.

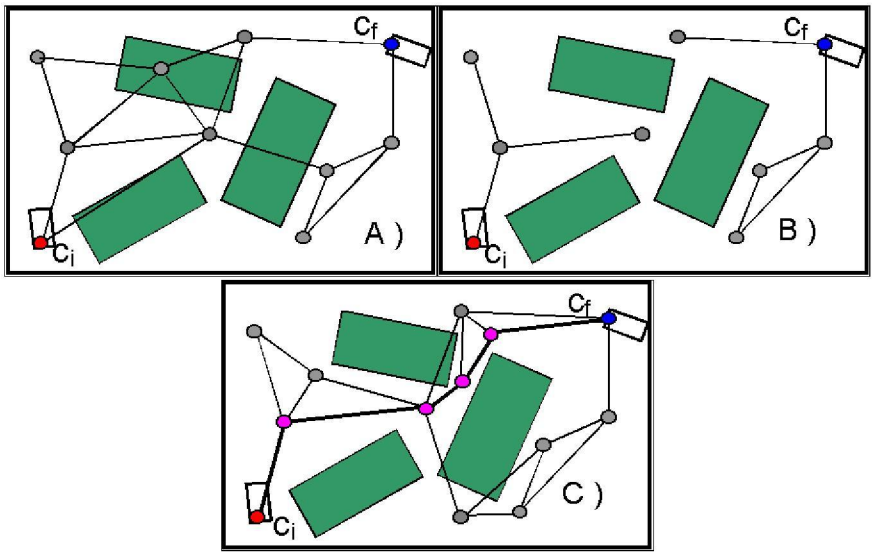

Figura 2.5: En A) se ilustra un roadmap generado, en el cuál al buscar un camino solución se llega a la situación mostrada en B), donde se observa que no existe camino alguno que lleve de c_i a c_f. Al llamar a un proceso de enriquecimiento es posible que el algoritmo concluya con éxito, como se muestra en C).

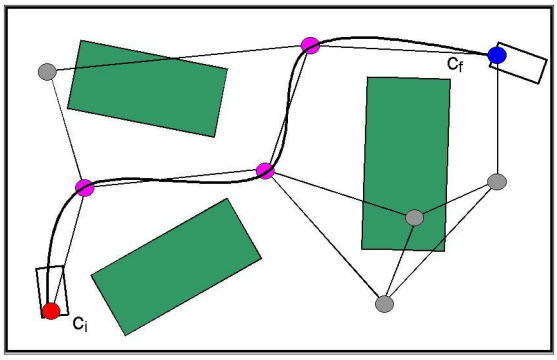

Figura 2.6: Camino factible para un robot tipo carro.

Palabra base	Secuencias de primitivas de movimiento
$C\|C\|C$	$(L^+R^-L^+)(L^-R^+L^-)(R^+L^-R^+)(R^-L^+R^-)$
$CC\|C$	$(L^+R^+L^-)(L^-R^-L^+)(R^+L^+R^-)(R^-L^-R^+)$
$C\|CC$	$(L^+R^-L^-)(L^-R^+L^+)(R^+L^+R^-)(R^-L^+R^+)$
CSC	$(L^+S^+L^+)(L^-S^-L^-)(R^+S^-R^+)(R^-S^-R^-)$
$CC_\beta\|C_\beta C$	$(L^+R_\beta^+L_\beta^-R^-)(L^-R_\beta^-L_\beta^+R^+)(R^+L_\beta^+R_\beta^-L^-)(R^-L_\beta^-R_\beta^+L^+)$
$C\|C_\beta C_\beta\|C$	$(L^+R_\beta^-L_\beta^-R^+)(L^-R_\beta^+L_\beta^+R^-)(R^+L_\beta^-R_\beta^-L^+)(R^-L_\beta^+R_\beta^+L^-)$
$C\|C_{\pi/2}SC$	$(L^+R_{\pi/2}^-S^-R^-)(L^-R_{\pi/2}^+S^+R^+)(R^+L_{\pi/2}^-S^-L^-)(R^-L_{\pi/2}^+S^+L^+)$ $(L^+R_{\pi/2}^-S^-L^-)(L^-R_{\pi/2}^+S^+L^+)(R^+L_{\pi/2}^-S^-R^-)(R^-L_{\pi/2}^+S^+R^+)$
$CSC_{\pi/2}\|C$	$(L^+S^+L_{\pi/2}^+R^-)(L^-S^-L_{\pi/2}^-R^+)(R^+S^+R_{\pi/2}^+L^-)(R^-S^-R_{\pi/2}^-L^+)$ $(R^+S^+L_{\pi/2}^+R^-)(R^-S^-L_{\pi/2}^-R^+)(L^+S^+R_{\pi/2}^+L^-)(L^-S^-R_{\pi/2}^-L^+)$
$C\|C_{\pi/2}SC_{\pi/2}\|C$	$(L^+R_{\pi/2}^-S^-L_{\pi/2}^-R^+)(L^-R_{\pi/2}^+S^+L_{\pi/2}^+R^-)(R^+L_{\pi/2}^-S^-R_{\pi/2}^-L^+)$ $(R^-L_{\pi/2}^+S^+R_{\pi/2}^+L^-)(L^+R_{\pi/2}^-S^-R_{\pi/2}^-R^+)(L^-R_{\pi/2}^+S^+R_{\pi/2}^+R^-)$ $(R^+L_{\pi/2}^-S^-L_{\pi/2}^-L^+)(R^-L_{\pi/2}^+S^+L_{\pi/2}^+L^-)$

Tabla 2.1: Familia de curvas Reeds&Shepp.

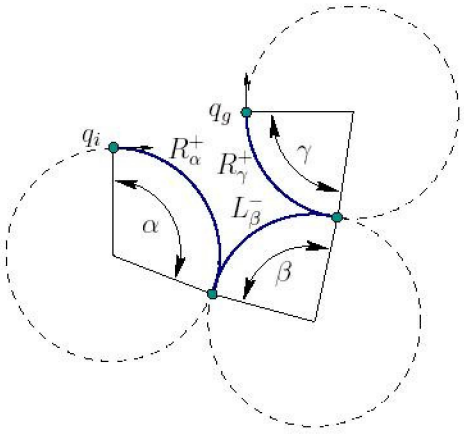

Figura 2.7: Ejemplo de curva Reeds and Shepp (C|C|C).

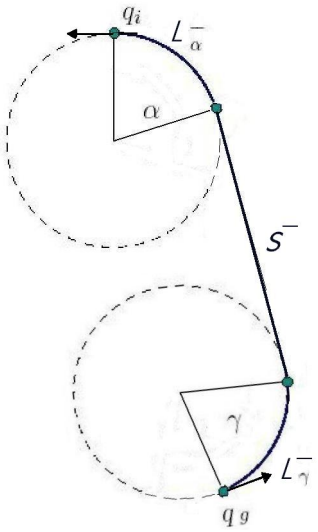

Figura 2.8: Ejemplo de curva Reeds and Shepp (C|S|C).

Capítulo 3

Zona virtual deformable

3.1. Introducción

La reactividad de un robot que navega en ambientes no estructurados y/o dinámicos es fundamental para la ejecución exitosa de sus tareas encomendadas. Pocos métodos han sido desarrollados e implementados en mecanismos móviles con capacidad reactiva. Uno de los más recientes es precisamente el método de control reactivo de la zona virtual deformable (ZVD) [7], [8], [12] y [19].

El control por ZVD está basado en una interacción robot / ambiente sin ninguna modelación de los obstáculos, dicha interacción se da por medio de una forma geométrica, llamada **zona virtual deformable** (Ξ), que rodea al robot móvil. Esta zona puede ser comparada con una acción refleja y enviarse al piloto en el nivel más bajo de una arquitectura de control.

El área y el perfil de esta zona dependen de dos tipos de parámetros:

- Las magnitudes que modelan la cinemática del vehículo.
- Las medidas proximétricas del ambiente proporcionadas por los sensores embarcados en el robot.

Por medio de las magnitudes del control del vehículo, podemos conocer el estado cinemático del robot. Esta acción arrastra una modificación de la forma de la ZVD. El ambiente, en cuanto a éste, actúa directamente sobre los parámetros de la ZVD. Entonces tenemos dos tipos de deformaciones de Ξ (Figura 3.1):

- Las deformaciones controladas debidas a los controles.
- Las deformaciones no controladas debidas a la evolución relativa del ambiente alrededor del vehículo.

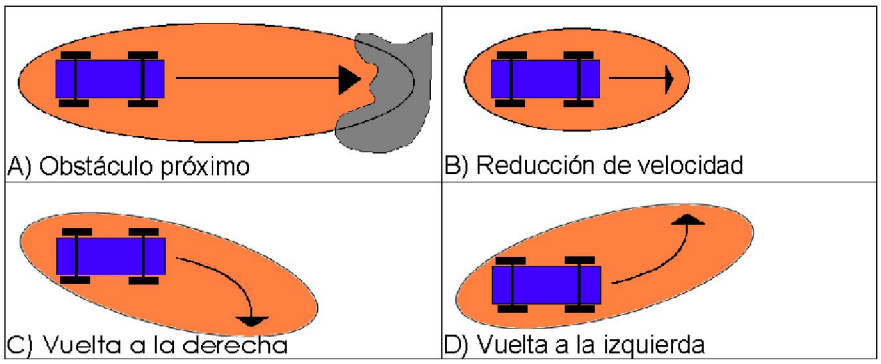

Figura 3.1: En A) la deformación de la ZVD es causada por la proximidad de un obstáculo; en B),C) y D) las deformaciones son causadas por comandos del robot.

El control por ZVD consiste en minimizar las deformaciones no controladas debidas al ambiente actuando sobre los controles del vehículo con la finalidad de obtener si es posible una zona de la cuál su forma dependa únicamente del estado del vehículo.

3.2. Ejemplo simplificado de acción refleja de paro

Un robot controlado en aceleración se desplaza sobre un riel recto, su estado cinemático es representado por su velocidad lineal. Un sensor proximétrico proporciona la distancia del primer obstáculo que se encuentra sobre la vía (Figura 3.2).

La zona virtual deformable Ξ toma la forma de un segmento de recta donde en una de las extremidades se encuentra el robot, y la otra extremidad se encuentra sobre la vía adelante del vehículo. La longitud de este segmento está dada por el mínimo de las dos distancias:

$$\Xi = Min(k * V^2 + d_{seguridad}, d_{obstáculo}) \tag{3.1}$$

donde:

- $d_{obstáculo}$ es la distancia al primer obstáculo,
- y $k*V^2+d_{seguridad}$ es la función que corresponde a la distancia de frenado necesaria para que el vehículo se detenga.

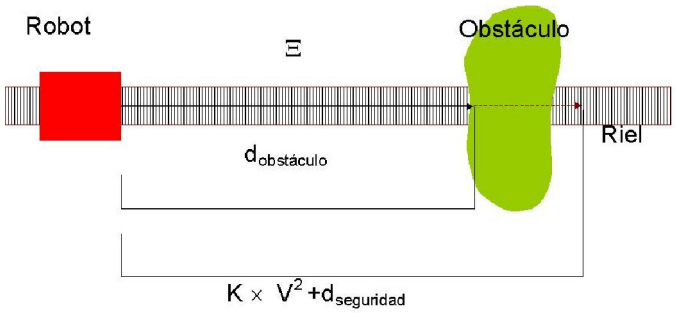

Figura 3.2: Forma de Ξ para un robot que se desplaza sobre un riel.

Cuando la zona virtual deformable Ξ es deformada por el ambiente ($k*V^2 + d_{seguridad} > d_{obstáculo}$, como en la figura 3.2), es necesario considerar la disminución de la aceleración del vehículo con la finalidad de obtener una ZVD que dependa únicamente del estado del robot ($k * V^2 + d_{seguridad} < d_{obstáculo}$).

3.3. Formalismo del método

El estado dinámico σ de un robot móvil representa sus actitudes y sus velocidades de desplazamiento. La zona virtual deformable Ξ alrededor del robot representa la componente de interacción entre el robot y su medio de evolución. El estado intrínseco χ de un robot está representado por estos dos componentes:

$$\chi = \begin{bmatrix} \Xi \\ \sigma \end{bmatrix}, con\ \bar{\Xi} = \begin{bmatrix} \Xi_1 \\ \Xi_2 \\ \vdots \\ \Xi_c \end{bmatrix} \tag{3.2}$$

La zona virtual deformable puede ser definida por un vector como el mostrado en la ecuación 3.2 donde cada componente Ξ_i es la norma de un vector correspondiente a la distancia de la frontera de la ZVD dentro de un sistema de referencia polar localizado en el centro del robot. Estos vectores pertenecen a las rectas vectoriales que a su vez corresponden a las direcciones principales de los c sensores de proximetría c_i.

Podemos escribir el control de un robot móvil bajo la forma siguiente:

$$\Phi = \dot{\pi} \tag{3.3}$$

con

$$\pi = f(\chi) \tag{3.4}$$

El vector de control Φ de un robot móvil es la derivada de ciertas variables cinemáticas reagrupadas dentro del vector π (ec. 3.3). La ecuación 3.4 unida por la función f representa el estado intrínseco χ del robot en ciertas de estas variables cinemáticas contenidas dentro del vector π.

3.4. Zona virtual deformable no deformada

Dentro del espacio de estado intrínseco E, denotamos como H al conjunto de los estados intrínsecos χ_h intactos, es decir, la zona donde Ξ no sufre ninguna deformación debida al ambiente. Podemos entonces encontrar una aplicación ρ que asocia a cada vector π un estado intrínseco χ_h donde la zona virtual Ξ_h no está deformada por el ambiente:

$$\rho : \Pi \to H, \quad \pi \to \rho(\pi) = \chi_h \quad con \quad \begin{cases} \Xi_h = \rho_\Xi(\pi) \\ \sigma = \rho_\sigma(\pi) \end{cases} \tag{3.5}$$

La composición de las dos funciones permite obtener para cada estado intrínseco χ el estado intrínseco intacto χ_h que debería tener el vehículo si su ZVD no estuviera deformada por el ambiente:

$$E \xrightarrow{f} \Pi \xrightarrow{\rho} H, \quad \chi \to \pi = f(\chi) \to \chi_h = \rho(\pi) \tag{3.6}$$

H es una invariante de la transformación $\rho o f$, es decir:

$$\forall \chi_h \in H, \rho o f(\chi_h) = \chi_h \in H \tag{3.7}$$

Controlar un robot por ZVD (Figura 3.3) consiste en aplicar un vector de control Φ que permita conservar el estado intrínseco χ del robot dentro del conjunto H. Esta zona virtual no deformada es entonces llamada zona virtual intacta.

Si en el curso del movimiento, la zona no está deformada por el ambiente, el estado intrínseco χ se quedará dentro de H. La evolución del estado intrínseco intacto χ_h a través del tiempo se obtiene en la ecuación 3.8, al derivar la ecuación 3.5 y utilizar la ecuación 3.3.

$$\dot{\chi_h} = \rho^{'}(\pi)\Phi \quad donde \quad \dot{\Xi_h} = \rho^{'}_{\Xi}(\pi)\Phi \tag{3.8}$$

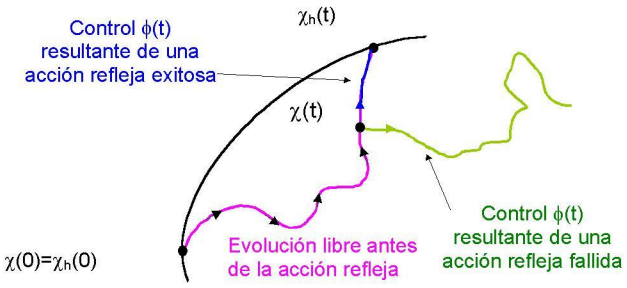

Figura 3.3: Evolución del estado intrínseco χ dentro de E.

3.5. Intrusiones y deformaciones debidas al ambiente

Sea I el **vector de intrusión** (ec. 3.9) de dimensión c, donde cada componente I_i expresa la profundidad de una intrusión siguiendo la recta vectorial D_i; esta información es resultado del dato aportado por el sensor c_i. El vector de intrusión I_i es evaluado a partir de la medida de distancia d_i al obstáculo, proporcionada por el sensor c_i, teniendo un alcance d_{imax} (Figuras 3.4 y 3.5), es decir:

$$I = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_i \\ \vdots \\ I_c \end{pmatrix} \tag{3.9}$$

$$I_i = d_{imax} - d_i \tag{3.10}$$

Cuando no hay ningún obstáculo, el sensor proporciona la medida $d_i = d_{imax}$

Definimos el **vector de deformación** Δ (ec. 3.11), de dimensión c, que nos proporciona la profundidad de la deformación de la zona virtual deformable Ξ

debida al ambiente. Cada componente Δ_i de este vector representa la longitud de deformación sobre la recta vectorial D_i.

$$\Delta = \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_i \\ \vdots \\ \Delta_c \end{pmatrix} \tag{3.11}$$

Cada elemento Δ_i está definido gracias a la función α mostrada en la ecuación 3.12 y que se ilustra con las figuras 3.4 y 3.5.

$$\Delta_i = \alpha(d_{hi}, I_i) = \begin{cases} 0 & si\ d_i > d_{hi} \\ d_{hi} - d_i & si\ d_i \leq d_{hi} \end{cases} \tag{3.12}$$

Donde d_{hi} es un elemento de la ZVD intacta Ξ_h

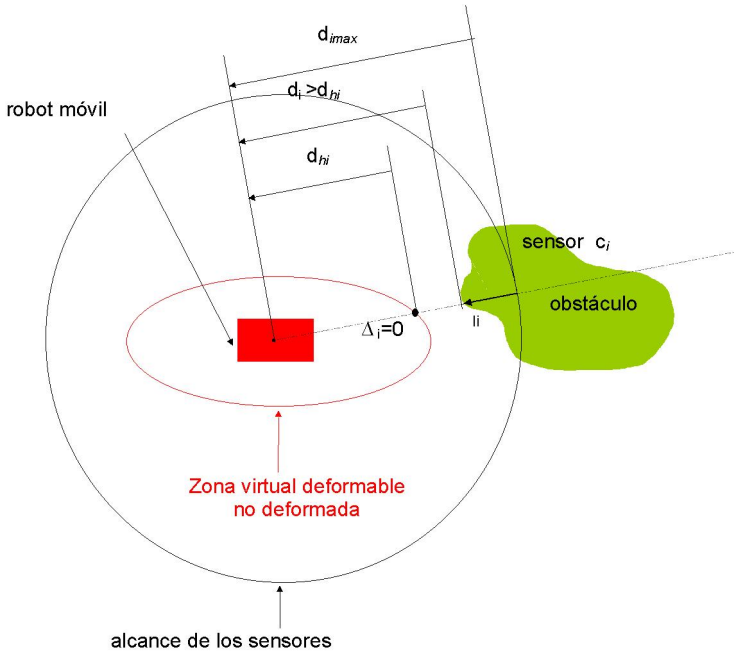

Figura 3.4: El vector de deformación Δ_i=0.

Es posible dividir el vector de representación de la zona virtual deformable Ξ en dos partes: un vector que representa la zona virtual deformable Ξ_h debida

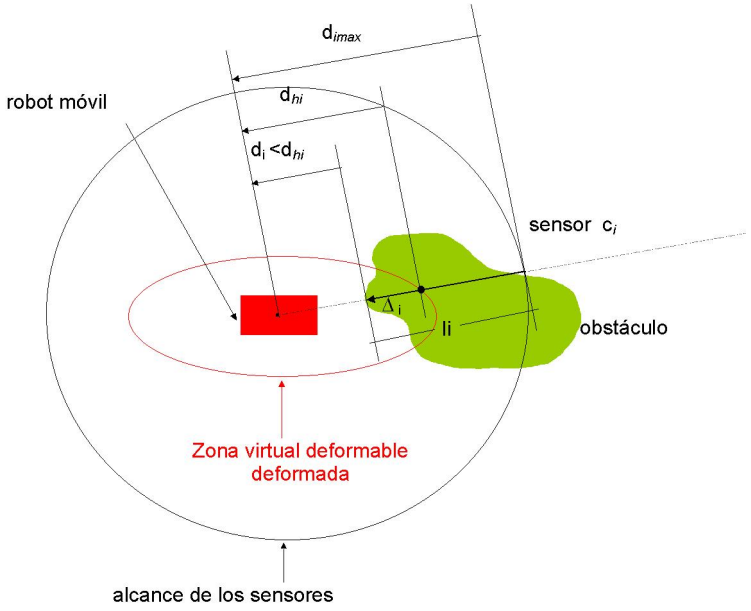

Figura 3.5: El obstáculo deforma la ZVD.

únicamente a Φ, y un vector Δ que representa la deformación generada por el ambiente (ec. 3.13).

$$\Xi = \Xi_h - \Delta \tag{3.13}$$

3.6. Ecuación de estado

De la ecuación 3.12, se puede obtener la ecuación de evolución de Δ:

$$\dot{\Delta} = \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I)\dot{\Xi}_h + \frac{\partial \alpha}{\partial I}(\Xi_h, I)\dot{I} \tag{3.14}$$

Derivando 3.13 y utilizando las ecuaciones 3.14 y 3.8

$$\dot{\Xi} = \rho_{\Xi}^{'}(\pi)\Phi - \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I)\rho_{\Xi}^{'}(\pi)\Phi - \frac{\partial \alpha}{\partial I}(\Xi_h, I)\dot{I} \tag{3.15}$$

Si de manera similar en el control (ec. 3.3), decimos que $\Psi = \dot{I}$, obtenemos las ecuaciones de estado de control por ZVD:

$$\dot{\chi} = \begin{cases} \dot{\Xi} = \Big(Id_{c\times c} - \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I)\Big)\rho_{\Xi}^{'}(\pi)\Phi - \frac{\partial \alpha}{\partial I}(\Xi_h, I)\Psi \\ \dot{\sigma} = \rho_{\sigma}^{'}(\pi)\Phi \end{cases} \tag{3.16}$$

$$con \begin{cases} \Phi = \dot{\pi} \\ \Psi = \dot{I} \\ \dot{\Xi}_h = \rho_{\Xi}^{'}(\pi)\Phi \end{cases} \tag{3.17}$$

3.7. Generación de comandos reflejos

Sean las matrices **A** con dim(A)=[c×n] (c sensores, n variables de control), y **B** con dim(B)=[c×c] de la siguiente manera:

$$\begin{array}{l} A = \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I)\rho_{\Xi}^{'}(\pi) \\ B = \frac{\partial \alpha}{\partial I}(\Xi_h, I) \end{array} \tag{3.18}$$

Susituyendo la ecuación 3.17 y la ecuación 3.18 en la ecuación 3.14, la evolución de la deformación es modelada por:

$$\dot{\Delta} = A\Phi + B\Psi \tag{3.19}$$

El algoritmo de control ZVD consiste en escoger una evolución deseada para la deformación $\dot{\Delta}_{des}$ de tal forma que se regrese a la zona virtual no deformada-da; para tal efecto es necesario aplicar un vector de control óptimo Φ_{opt}, el cuál se obtiene al despejarlo en una ecuación con la forma 3.19, como se muestra en la ecuación 3.20.

$$\Phi_{opt} = A^{+}(\dot{\Delta}_{des} - B\Psi) \tag{3.20}$$

donde A^{+} es la pseudoinversa de A.

Una simple y eficiente ley de control consiste en escoger la deformación deseada de forma proporcional a la deformación real y a su derivada:

$$\dot{\Delta}_{des} = K_p\Delta + K_d\dot{\Delta} \tag{3.21}$$

Donde las dos matrices K_p y K_d son respectivamente las ganancias proporcional y derivativa, seleccionadas para evitar colisiones.

3.8. Aplicaciones

Desarrollada originalmente para asegurar la evasión refleja de un robot móvil, el control por ZVD se ha aplicado en:

- El pilotaje completo de robots con ruedas.
- En la estabilización dinámica de robots con ruedas.

- En el equilibrio dinámico de robots con patas.
- En la protección refleja de robots manipuladores.

Las acciones reflejas por ZVD pueden ser implantadas en conjunto, funcionando de manera paralela. Es suficiente entonces proporcionar prioridades a las diferentes zonas implantadas.

Capítulo 4

Un enfoque Lazy-PRM reactivo

4.1. Introducción

En este capítulo se describe la propuesta realizada en esta tesis, la cuál integra los métodos de planificación Lazy PRM y de control reactivo por ZVD de la siguiente manera: en forma previa a la ejecución se calcula un camino factible y libre de colisiones para un robot tipo carro con el método Lazy PRM, una vez obtenido el camino, se inicia su ejecución estando el robot bajo la protección permanente de la ZVD, mientras ningún obstáculo dinámico irrumpa en ella, no se requieren comandos reflejos y el control corre por cuenta de Lazy PRM. En caso contrario, el método reactivo asume el control y genera comandos que alejan al robot de los obstáculos intrusos para devolver su ZVD al estado intacto. Ya en este punto el robot ha extraviado el camino y es necesario buscar un camino de reconexión con el camino pendiente por recorrer, el cuál consistirá en una curva Reeds&Shepp libre de colisiones de una sola palabra. Si el intento de reconexión tiene éxito, el robot ejecuta el nuevo camino hacia su objetivo, sino se ejecuta el método Lazy PRM, el cuál planificará un camino alterno al anterior, a partir de la configuración actual del robot. Durante los procesos de reconexión y de replanificación el robot permanece fijo, pero en caso de presentarse una deformación se interrumpen por acciones reflejas.

El algoritmo puede terminar de tres formas: el robot ejecuta su camino con éxito, la acción refleja no es suficiente y se produce una colisión, o bien, el robot no encuentra un camino alterno para concluir su tarea. Lo anterior se puede ilustrar en la figura 4.1. El resto del capítulo explica cada bloque con detalle.

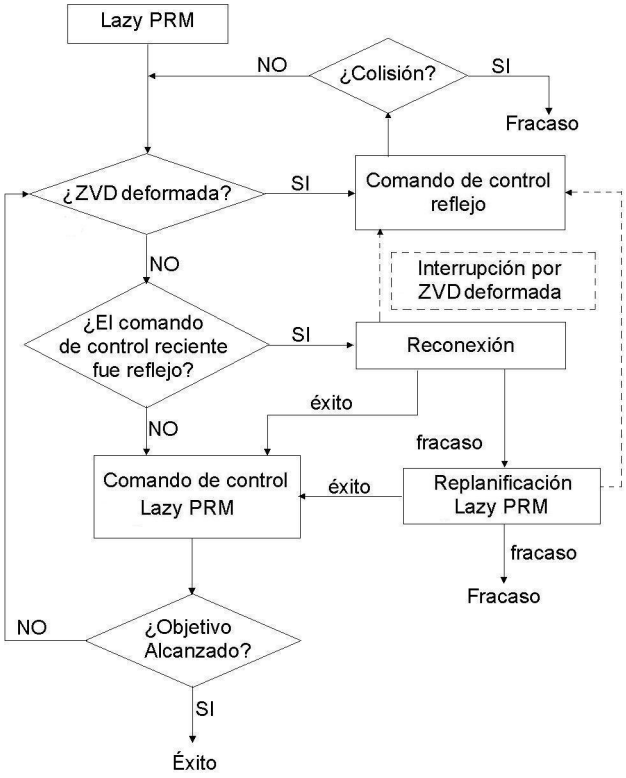

Figura 4.1: Descripción de alto nivel del algoritmo propuesto.

4.2. Planificador Lazy PRM

La descripción de alto nivel del algoritmo empleado, corresponde a la mostrada en la figura 2.1, sin embargo el funcionamiento interno de algunos bloques de dicha figura es algo diferente a la descrita en el capítulo 2 , como a continuación se detalla.

4.2.1. Construcción del roadmap inicial

Como primer paso se inicializa al roadmap con las configuraciones inicial y final, y el arco entre éstas, luego inicia un ciclo en el que se selecciona cada vez una configuración al azar, en x y y, sin considerar aún el ángulo de orientación. La configuración generada se añade al roadmap y se seleccionan sus k-vecinos más próximos, con cada uno de los cuáles se establece un arco. El procedimiento se describe a continuación:

$V \leftarrow c_i, c_f$
$E \leftarrow (c_i, c_f)$
$mientras\ n_{nodos} < N\ hacer$
$\quad c \leftarrow configuración\ aleatoria$
$\quad V \leftarrow V \bigcup \{c\}$
$\quad \forall\ v \in V, en\ orden\ ascendente\ de\ d(v,c)\ hacer$
$\quad\quad si\ d(c,v) < dmax \wedge tamaño(V_c) < k$
$\quad\quad\quad E \leftarrow E \bigcup \{(c,v)\}$
fin

4.2.2. Búsqueda del camino más corto

Para tal fin se emplea el algoritmo A*. Una vez obtenido el camino se asignan valores a los ángulos de las configuraciones que constituyen los nodos internos del camino, como se ilustra en la figura 4.2.

4.2.3. Revisión de colisiones

Como primer paso se revisan los nodos del camino de la configuración inicial a la final, si hay nodos en colisión se eliminan así como también sus arcos y se busca nuevamente el camino más corto, si no hay nodos en colisión se llama al método local Reeds&Shepp para los arcos del camino, revisándose sólo aquellas configuraciones que corresponden a los extremos iniciales de los componentes elementales de cada camino local obtenido. Si hay caminos en colisión se eliminan los arcos correspondientes y se busca nuevamente el camino más corto, en caso contrario se discretizan las curvas Reeds&Shepp a una resolución muy fina entre configuraciones, para cada una de las cuáles se llama al detector de colisiones.

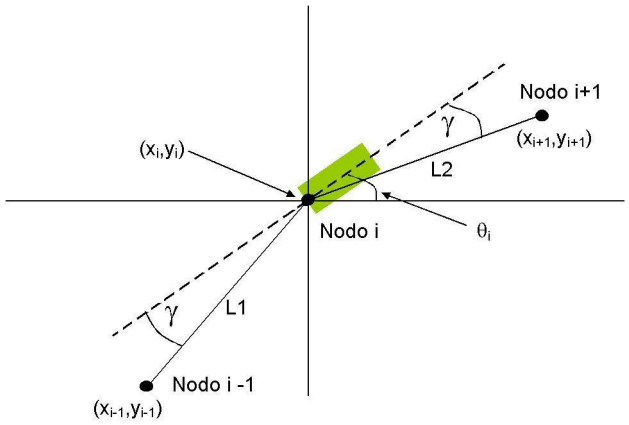

Figura 4.2: Obtención de θ para la configuración del nodo i-ésimo del camino más corto.

Detector de colisiones

Se utilizan dos detectores de colisiones, uno de ellos es completo y el otro no. El detector completo revisa cada pixel que el robot (que tiene forma rectangular) ocupa en el ambiente y se utiliza sólo para validar las configuraciones inicial y final. El otro detector se utiliza para validar el camino calculado (Figura 4.4) y solo revisa las esquinas del robot, así como el punto que corresponde a la configuración en curso (Figura 4.3). Este número reducido de puntos de revisión lo hace más rápido que al detector completo y aunque es posible que un obstáculo pequeño quede dentro del camino por recorrer, los sensores empleados por el método de la ZVD los detectan y evitan (en algunos casos no es posible, esto se aclara en las conclusiones de la tesis, en el capítulo 6.)

Figura 4.3: Puntos que se revisan en el detector de colisiones incompleto.

4.2.4. Proceso de enriquecimiento

Varias llamadas al proceso de enriquecimiento del roadmap pueden ocasionar una cantidad excesiva de nodos en el mismo, para evitar esto, cada vez que fracasa el algoritmo A*, se revisa si el número de nodos es menor al límite definido, si este es el caso se enriquece el roadmap como se inidica en la sección 2.5, si no, se destruye el roadmap y se genera uno nuevo.

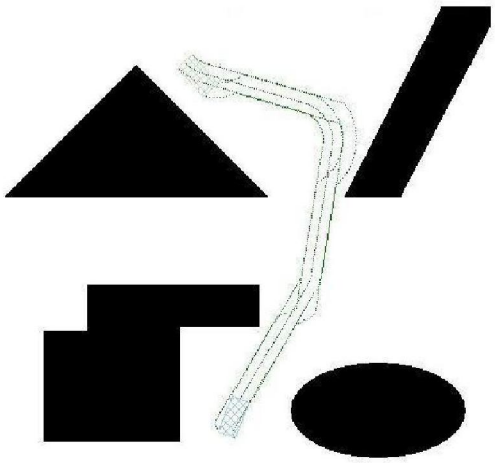

Figura 4.4: Pixeles revisados del camino solución por el detector de colisiones incompleto.

4.3. Control reactivo por ZVD

Con base en las ecuaciones introducidas en el Capítulo 3 y con la ecuación siguiente se asigna la forma de la ZVD utilizada en este trabajo:

$$d_{hi} = K_1 V_1^2 cos^2(\beta_i + K_2\dot{\theta}) + d_i^{sec} \tag{4.1}$$

Donde K_1 y K_2 son constantes, V_1 y $\dot{\theta}$ son velocidades del robot (ec. 4.6), β_i es el angulo del sensor c_i con respecto al eje transversal del robot, y d_i^{sec} es una distancia de seguridad en la dirección del sensor ci. La figura 4.5 muestra un ejemplo de la forma de la ZVD intacta así obtenida para algunos valores específicos.

Para el primer caso de la ecuación 3.12 ($d_i > d_{hi}$), la ZVD no está deformada por el ambiente, no hay acciones reflejas y se ejecuta el camino planificado por Lazy PRM. Sin embargo cuando ($d_i < d_{hi}$), una acción refleja es necesaria, el camino Lazy PRM se suspende y el control del robot es asumido por el método de la ZVD.

Generación de comandos reflejos

Cuando el método de la ZVD asume el control, tiene la tarea de llevar al robot a un estado libre de deformaciones, indicando las actitudes cinemáticas que deberá ir presentando el robot continuamente. Estas actitudes constituyen al vector π descrito en la ecuación 3.4, y que en este caso adoptará la forma

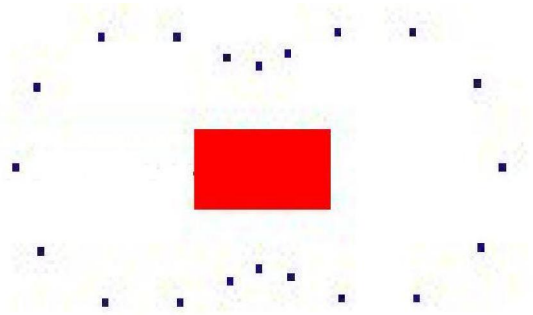

Figura 4.5: Forma obtenida para una ZVD de 20 sensores, según la ecuación 4.1, cuando se utilizan los siguientes valores: V_1=3, $\dot{\theta}$=0, K_1=5, $K_2 = 10$ y d_i^{sec}=25.

$$\pi = \begin{bmatrix} V_1 \\ \dot{\theta} \end{bmatrix} \tag{4.2}$$

En este trabajo, en lugar de implementar el control como indica la ecuación 3.20, se utiliza lo siguiente:

Sea $f_i[n]$ un vector en la dirección del sensor c_i en el muestreo n-ésimo definido como:

$$f_i[n] = \begin{cases} \Delta_i[n] - \Delta_i[n-1] & si\ \Delta_i[n] - \Delta_i[n-1] > 0 \\ 0 & si\ \Delta_i[n] - \Delta_i[n-1] \leq 0 \end{cases} \tag{4.3}$$

y sea $F[n]$ el vector suma de los vectores $f_i[n]$

$$F[n] = \sum_{i=1}^{c} f_i[n] \tag{4.4}$$

entonces, el vector $\pi[n]$ estará dado por:

$$\pi[n] = \begin{cases} V_1[n] = V_1[n-1] + Kv * \|F[n]\| * signo(cos(\hat{F}[n])) \\ \dot{\theta}[n] = \dot{\theta}[n-1] + Kt * sen(\hat{F}[n])) \end{cases} \tag{4.5}$$

4.4. Reconexión

Cuando el robot móvil después de una acción refleja exitosa, recupera el estado intacto de su ZVD, ya ha extraviado el camino planificado (Figura 4.6-B), sin embargo bajo el esquema Lazy PRM propuesto, es indispensable que el móvil posea un camino que lo lleve a su objetivo, por lo que en este punto es

necesario proporcionarle un camino para tal fin.

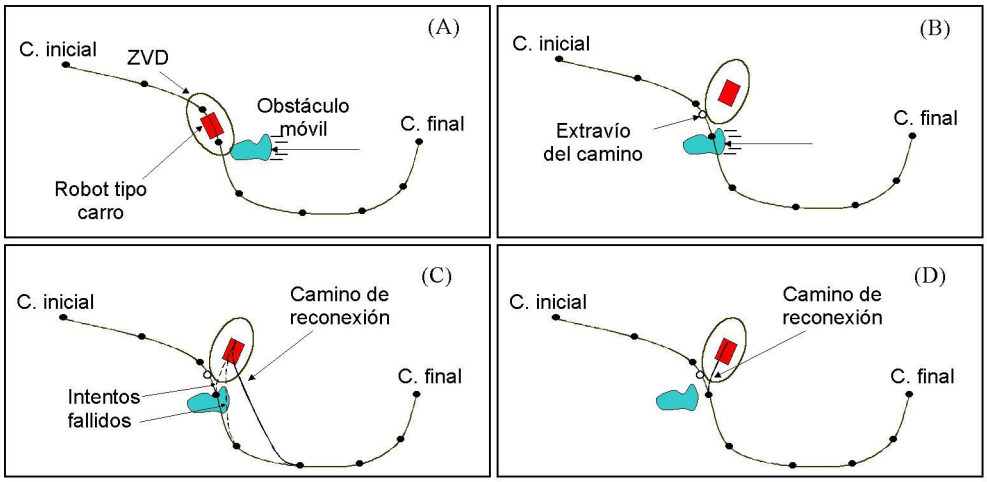

Figura 4.6: En (A) es necesaria una acción refleja para evitar el obstáculo móvil, en (B) se muestra la situación del robot después de la acción refleja, en (C) se muestra el camino de reconexión al camino planificado. En el caso de que todos los intentos de (C) hubieran estado en colisión, el proceso se repite después de algún tiempo, y es posible entonces obtener éxito, como sucede en (D).

Ya que el costo computacional de una replanificación completa es alto, se evita en lo posible, recurriendo a un proceso que consiste en reconectar con el camino planificado mediante un camino de una sola curva Reeds&Shepp que se encuentre libre de colisiones, como se ilustra en la figura 4.6-C.

Inicialmente se intenta un camino de reconexión con el camino local siguiente al interrumpido por la acción refleja, si esto no es posible se continúa con los siguientes hasta encontrar una reconexión libre de colisiones o bien hasta realizar un determinado número de intentos. Si no hay éxito, es probable que los posibles caminos estuvieran bloqueados por objetos que un tiempo adelante ya no interferirán, por tanto el robot permanecerá inmóvil un tiempo determinado, transcurrido el cuál se realizará un nuevo intento (Figura 4.6-D), este proceso se puede repetir varias veces.

Si durante el proceso anterior la ZVD es deformada por una nueva intrusión, el proceso de reconexión se abandona para ejecutar comandos reflejos.

4.5. Replanificación

Si los diferentes intentos de reconexión fallan, es porque las posibles caminos de reconexión están bloqueados por un tránsito elevado de objetos móviles, o porque un objeto antes móvil se ha detenido obstruyendo el camino planificado. Solo en este caso se recurrirá a una replanificación completa, es decir se llama nuevamente al método Lazy PRM, pasándole como configuración inicial, la configuración actual del robot. Se puede llamar varias veces al planificador Lazy PRM hasta que devuelva un camino libre de colisiones; si después de ciertos intentos no se obtiene un camino, se reporta fracaso.

4.6. Simulación de la cinemática del robot

Tanto el camino calculado por Lazy PRM como las acciones reflejas, indican al robot las actitudes cinemáticas que deberá asumir constantemente. Estas actitudes son V_1 y $\dot{\theta}$, como se muestra en la ecuación 4.6 y conforman el vector π de la ecuación 3.4. Estas actitudes se indican a intervalos Δt, siendo constantes durante cada uno de ellos.

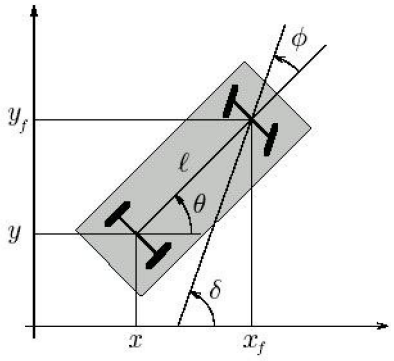

Figura 4.7: Diagrama del robot tipo carro.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos\ \theta \\ sen\ \theta \\ (tan\ \phi)/l \\ 0 \end{bmatrix} V_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} V_2 \tag{4.6}$$

El hecho de que el cambio en x,y y θ, sea constante durante cada intervalo, significa que el ángulo de conducción ϕ de las ruedas delanteras permanece fijo durante todo el intervalo ($V_2 = 0$), describiendo un camino circular durante el

mismo.

La observación anterior es útil para evitar la operación de integración, requerida de otra forma para determinar los valores de x,y, y θ, para el siguiente intervalo de tiempo. En lugar de ello basta con utilizar geometría analítica en una circunferencia. La ecuación 4.7 con ayuda de la figura 4.8, muestra como se obtiene la nueva configuración para el robot después de la aplicación de un impulso determinado.

$$\begin{aligned} x_2 &= x_1 + r(cos\gamma_2 - cos\gamma_1) \\ y_2 &= y_1 + r(sen\gamma_2 - sen\gamma_1) \\ \theta_2 &= \theta_1 + q * (V_1/r) * \Delta t \\ & \\ &donde: \\ & \\ r &= MAX(abs(V_1/\dot{\theta}), R_{min}) \\ q &= signo(V_1/\dot{\theta}) \\ \gamma_1 &= \theta_1 - q(\pi/2) \\ \gamma_2 &= \theta_2 - q(\pi/2) \end{aligned} \tag{4.7}$$

R_{min} es el radio de la trayectoria que describe el robot tipo carro cuando sus ruedas delanteras están en su ángulo máximo de conducción.

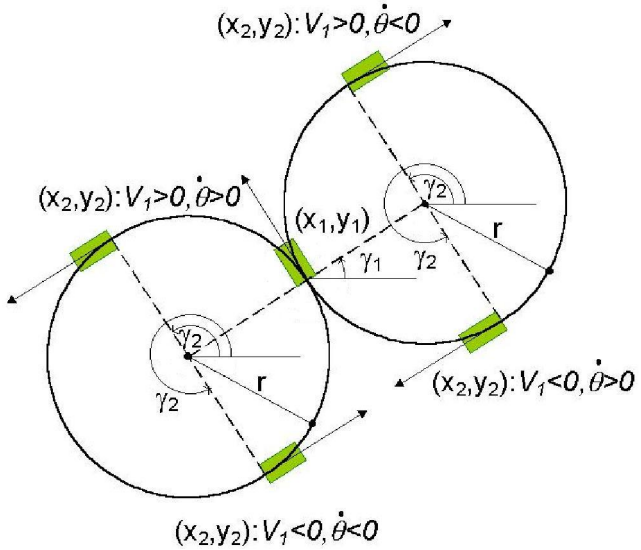

Figura 4.8: La figura muestra la relación entre las variables de la ecuación 4.7.

4.7. Distinción entre obstáculos móviles y fijos

En el caso de que el robot móvil se desenvuelva en un ambiente estático (o regionalmente estático), el camino planificado basta para asegurar que no habrá colisión, bajo esta suposición es necesario que cuando un obstáculo fijo irrumpa en la ZVD, no se genere acción refleja alguna.. Sin embargo ya que el método de la ZVD no utiliza ningún modelo del ambiente, no puede distinguir cuando una intrusión se debe a un obstáculo fijo y cuando a uno móvil.

Para resolver este problema se utiliza una imagen auxiliar del ambiente que se actualiza cada vez que se llama al procedimiento de reconexión o de replanificación. Así, cada vez que los sensores embarcados en el robot detectan un obstáculo que deforma a la ZVD, se revisa si en las coordenadas donde se detectó al objeto intruso, ya existía un obstáculo en la imagen auxiliar, en caso afirmativo se asume que se enfrenta a un obstáculo fijo y no se requiere una acción refleja, en caso contrario es totalmente seguro que el obstáculo es móvil.

Capítulo 5

Resultados experimentales

Este capítulo presenta los resultados obtenidos en la evaluación del método Lazy-PRM reactivo. El algoritmo fue implementado con Builder C++ y ejecutado en una computadora personal basada en un procesador AMD K6-II a 550Mhz con 64 Mb en RAM.

Las pruebas evaluaron el desempeño mostrado en la planificación y ejecución de caminos, para un robot tipo carro que se desenvuelve en ambientes dinámicos.

5.1. Evaluación del planificador Lazy PRM

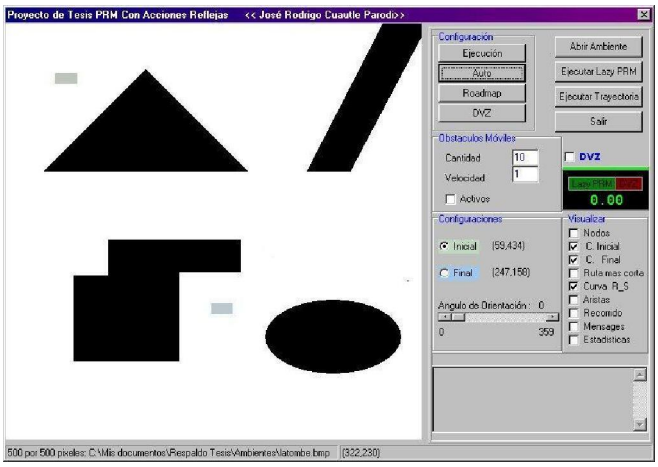

Figura 5.1: Ambiente utilizado para evaluar al método Lazy-PRM reactivo.

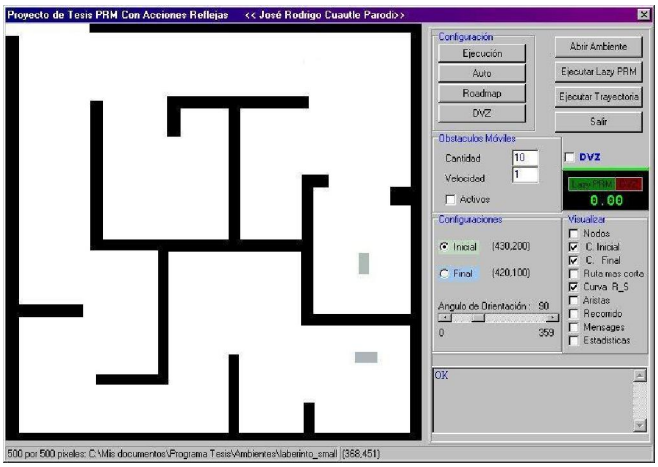

Figura 5.2: Ambiente utilizado para evaluar el desempeño del método en situaciones más complicadas.

El planificador fue probado en los ambientes mostrados en las figuras 5.1 y 5.2 (ambos son bitmaps de 500×500), bajo diferentes condiciones: cantidad inicial de nodos N que conforman el roadmap de 50, 100, 200 y 400, con límites de 100, 200, 300 y 500 respectivamente. Número de vecinos V_c para cada nodo de 5 y 15, distancia máxima d_{max} para poder insertar un arco entre dos nodos del 30 y 50 por ciento de la longitud de un lado del ambiente, y finalmente diferentes restricciones cinemáticas, que corresponden a ángulos de conducción máximos de 45 y 70 grados.

Para todas las pruebas, los siguientes parámetros permanecieron constantes: nodos agregados en cada enriquecimiento: 10, dimensiones del robot móvil: 25 × 13 píxeles, resolución máxima entre configuraciones revisadas por el detector de colisiones: 2 píxeles, distancia entre ejes del robot móvil : 25 píxeles, número máximo de búsquedas permitidas: 500. Para cada combinación posible se realizaron 10 planificaciones.

Las tablas 5.1 y 5.2 muestran los resultados promediados obtenidos para el planificador Lazy PRM en el ambiente de la figura 5.1, con una configuración inicial c_i en (59,434,0°) y una configuración final c_f en (247,185,0°).

La figura 5.3 muestra un ejemplo de camino planificado para el ambiente 5.1.

De igual manera, las tablas 5.3 y 5.4 muestran los resultados obtenidos para el planificador Lazy PRM en un ambiente más complicado (Figura 5.2), con una configuración inicial c_i en (430,200,90°) y una configuración final c_f en (420,100,0°).

Condiciones				**Resultados Obtenidos**							
N	V_c	d_{max}	ϕ	Construcción del roadmap	Iteraciones	Nodos resultantes	Llamadas al detector de colisiones	Tiempo acumulado en las llamadas	Tiempo en A*	Tiempo total L. PRM	Exitos
		(%)	(Grados)	(Seg.)				(Seg.)	(Seg.)	(Seg.)	
50	5	30	45	0.007	13	52	980	0.045	0.005	0.071	10/10
50	5	30	70	0.006	11	50	812	0.036	0.003	0.054	10/10
50	5	50	45	0.007	18	44	1182	0.052	0.007	0.084	10/10
50	5	50	70	0.007	14	52	1540	0.072	0.008	0.099	10/10
50	15	30	45	0.008	16	42	1072	0.047	0.004	0.073	10/10
50	15	30	70	0.008	15	42	1093	0.050	0.005	0.074	10/10
50	15	50	45	0.012	26	40	1985	0.093	0.008	0.136	10/10
50	15	50	70	0.012	27	40	2187	0.107	0.007	0.146	10/10
100	5	30	45	0.016	17	91	1299	0.061	0.016	0.109	10/10
100	5	30	70	0.016	12	84	1011	0.051	0.011	0.088	10/10
100	5	50	45	0.016	16	84	1088	0.054	0.012	0.093	10/10
100	5	50	70	0.017	15	85	1229	0.061	0.012	0.100	10/10
100	15	30	45	0.028	27	82	1600	0.080	0.017	0.148	10/10
100	15	30	70	0.028	24	81	1712	0.085	0.015	0.147	10/10
100	15	50	45	0.035	36	82	2248	0.117	0.024	0.207	10/10
100	15	50	70	0.034	35	81	2527	0.137	0.022	0.218	10/10

Tabla 5.1: Resultados de la planificación de caminos en la figura 5.1 para N=50 y N=100.

Condiciones				**Resultados Obtenidos**							
N	V_c	d_{max}	ϕ	Construcción del roadmap	Iteraciones	Nodos resultantes	Llamadas al detector de colisiones	Tiempo acumulado en las llamadas	Tiempo en A*	Tiempo total L. PRM	Exitos
		(%)	(Grados)	(Seg.)				(Seg.)	(Seg.)	(Seg.)	
200	5	30	45	0.051	16	172	886	0.047	0.031	0.143	10/10
200	5	30	70	0.052	16	174	885	0.048	0.036	0.150	10/10
200	5	50	45	0.053	17	174	1063	0.055	0.038	0.164	10/10
200	5	50	70	0.052	16	174	1095	0.059	0.033	0.158	10/10
200	15	30	45	0.095	37	167	2128	0.121	0.070	0.321	10/10
200	15	30	70	0.099	35	169	2568	0.145	0.065	0.340	10/10
200	15	50	45	0.107	44	169	2811	0.162	0.079	0.392	10/10
200	15	50	70	0.106	49	166	2975	0.178	0.089	0.418	10/10
400	5	30	45	0.187	20	351	1392	0.086	0.169	0.462	10/10
400	5	30	70	0.194	17	351	1019	0.073	0.133	0.416	10/10
400	5	50	45	0.196	20	352	1046	0.077	0.155	0.446	10/10
400	5	50	70	0.200	20	353	1034	0.075	0.166	0.459	10/10
400	15	30	45	0.356	42	343	1899	0.188	0.251	0.836	10/10
400	15	30	70	0.364	46	343	2540	0.225	0.282	0.914	10/10
400	15	50	45	0.373	51	343	2370	0.223	0.304	0.950	10/10
400	15	50	70	0.381	50	340	2878	0.257	0.310	0.995	10/10

Tabla 5.2: Resultados de la planificación de caminos en la figura 5.1 para N=200 y N=400.

Condiciones				**Resultados Obtenidos**							
N	V_c	d_{max}	ϕ	Construcción del roadmap	Iteraciones	Nodos resultantes	Llamadas al detector de colisiones	Tiempo acumulado en las llamadas	Tiempo en A*	Tiempo total L. PRM	Exitos
		(%)	(Grados)	(Seg.)				(Seg.)	(Seg.)	(Seg.)	
50	5	30	45	0.007	453	64	11785	0.522	0.277	1.843	2/10
50	5	30	70	0.006	460	73	15919	0.728	0.280	2.130	4/10
50	5	50	45	0.007	421	53	14795	0.664	0.264	1.900	3/10
50	5	50	70	0.007	384	69	18644	0.820	0.232	1.889	6/10
50	15	30	45	0.009	297	54	18466	0.814	0.185	1.623	6/10
50	15	30	70	0.009	239	66	19589	0.860	0.146	1.483	8/10
50	15	50	45	0.012	283	53	24345	1.089	0.138	1.820	7/10
50	15	50	70	0.012	279	57	30641	1.376	0.152	2.091	8/10
100	5	30	45	0.017	326	117	9539	0.457	0.678	2.058	5/10
100	5	30	70	0.017	264	115	11351	0.538	0.516	1.763	7/10
100	5	50	45	0.018	323	123	11453	0.547	0.647	2.081	7/10
100	5	50	70	0.018	264	136	12319	0.585	0.593	1.845	8/10
100	15	30	45	0.028	128	98	11383	0.520	0.190	0.963	10/10
100	15	30	70	0.029	105	102	13082	0.601	0.155	0.947	10/10
100	15	50	45	0.033	185	68	16582	0.764	0.210	1.340	10/10
100	15	50	70	0.032	171	70	21844	1.071	0.196	1.589	10/10

Tabla 5.3: Resultados de la planificación de caminos en la figura 5.2 para N=50 y N=100.

Condiciones				**Resultados Obtenidos**							
N	V_c	d_{max}	ϕ	Construcción del roadmap	Iteraciones	Nodos resultantes	Llamadas al detector de colisiones	Tiempo acumulado en las llamadas	Tiempo en A*	Tiempo total L. PRM	Exitos
		(%)	(Grados)	(Seg.)				(Seg.)	(Seg.)	(Seg.)	
200	5	30	45	0.054	275	186	10168	0.520	1.404	2.952	8/10
200	5	30	70	0.053	322	186	13065	0.672	1.689	3.713	6/10
200	5	50	45	0.053	176	206	7851	0.392	0.956	1.909	10/10
200	5	50	70	0.055	200	213	10739	0.546	1.017	2.235	10/10
200	15	30	45	0.093	152	143	13666	0.648	0.532	1.557	10/10
200	15	30	70	0.094	151	147	17750	0.835	0.536	1.746	10/10
200	15	50	45	0.101	194	135	17138	0.824	0.629	1.924	10/10
200	15	50	70	0.100	212	138	24888	1.206	0.679	2.400	10/10
400	5	30	45	0.145	138	352	7269	0.386	2.453	3.600	10/10
400	5	30	70	0.193	141	349	8393	0.465	2.404	3.652	10/10
400	5	50	45	0.199	119	359	6193	0.346	2.147	3.086	10/10
400	5	50	70	0.195	142	329	8604	0.464	2.195	3.413	10/10
400	15	30	45	0.354	211	298	15914	0.852	2.841	4.422	10/10
400	15	30	70	0.346	216	295	22171	1.150	2.852	4.745	10/10
400	15	50	45	0.352	267	289	22116	1.132	3.281	5.317	10/10
400	15	50	70	0.360	281	294	28953	1.390	3.507	5.850	9/10

Tabla 5.4: Resultados de la planificación de caminos en la figura 5.2 para N=200 y N=400.

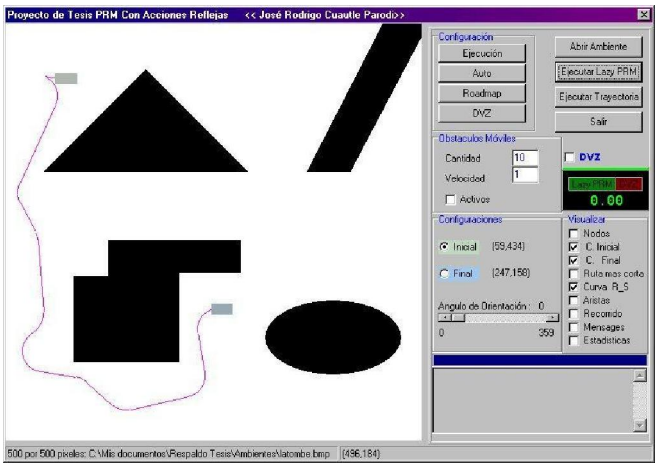

Figura 5.3: Ejemplo de camino planificado para el ambiente de la figura 5.1.

La figura 5.4 muestra un ejemplo de camino planificado para el ambiente de la figura 5.2.

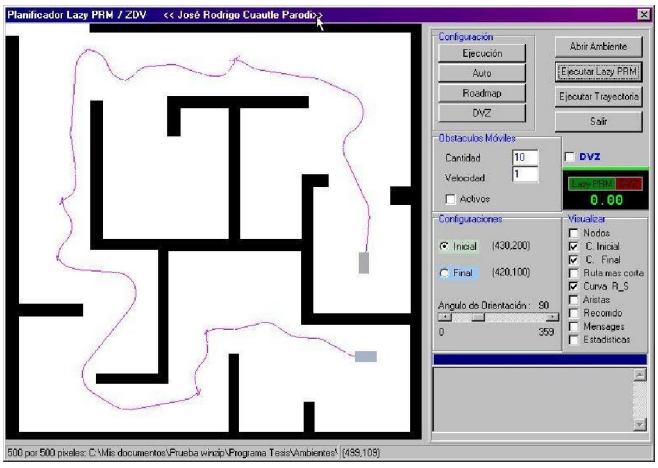

Figura 5.4: Ejemplo de camino planificado para el ambiente de la figura 5.2.

5.2. Evaluación del método Lazy-PRM reactivo en la ejecución de caminos

El dinamismo de los ambientes ya utilizados en las pruebas anteriores, corre por cuenta de obstáculos móviles ubicados aleatoriamente, de forma cuadrada

de 10×10 píxeles que se desplazan en línea recta a velocidad constante; cada vez que colisionan con otro objeto asumen en su movimiento una nueva dirección seleccionada al azar.

Para la ejecución de caminos en el ambiente mostrado en la figura 5.1, con base en las tablas 5.1 y 5.2, se seleccionaron las siguientes condiciones para el planificador Lazy PRM:

- N : 50.
- V_c: 5.
- d_{max}:30 %.
- ϕ: 45°.

Así también se utilizaron los siguientes valores para las acciones reflejas:

- Sensores: 20.
- K_1 : 0.5.
- K_2: 3.
- K_v: 2.
- K_t: 2.
- d^{sec}: 25.
- Intentos de reconexión antes de replanificar: 40.
- Velocidad del robot bajo Lazy PRM: 2.
- Velocidad máxima en acciones reflejas: 5.
- Búsquedas de caminos alternos antes de terminar con fracaso: 3.

Las tablas 5.5 a 5.8 exhiben los resultados obtenidos para diferentes cantidades y velocidades de obstáculos móviles cuando el robot ejecuta caminos planificados como el mostrado en el ambiente de la figura 5.5. Cada tabla muestra los resultados de 10 ejecuciones.

Así también, para la ejecución de caminos en el ambiente mostrado en la figura 5.2, con base en las tablas 5.3 y 5.4, se seleccionaron las siguientes condiciones para el planificador Lazy PRM:

- N : 100.
- V_c: 15.
- d_{max}:30 %.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
43	0.027	0	0			•
12	0.025	0	0			•
3	0.025	0	0			•
0	0.000	0	0			•
0	0.000	0	0			•
65	0.030	0	0			•
12	0.026	0	0			•
16	0.026	0	0			•
84	0.031	0	0			•
97	0.027	0	0			•

Tabla 5.5: Resultados en la ejecución de caminos en el ambiente de la figura 5.1 con 5 obstáculos móviles con velocidad V=1.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
36	0.029	0	0			•
36	0.031	0	0			•
90	0.029	1	0.019			•
81	0.032	1	0.115			•
3	0.027	0	0		•	
0	0.000	0	0			•
9	0.026	0	0			•
5	0.026	0	0		•	
27	0.032	1	0			•
4	0.026	0	0			•

Tabla 5.6: Resultados en la ejecución de caminos en el ambiente de la figura 5.1 con 5 obstáculos móviles con velocidad V=2.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
70	0.037	0	0			•
63	0.026	0	0			•
74	0.034	0	0			•
11	0.027	0	0			•
213	0.030	1	0.079			•
75	0.038	0	0			•
43	0.038	0	0			•
8	0.028	0	0		•	
27	0.031	0	0		•	
44	0.027	0	0			•

Tabla 5.7: Resultados en la ejecución de caminos en el ambiente de la figura 5.1 con 10 obstáculos móviles con velocidad V=1.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
0	0.000	0	0			•
166	0.040	1	0.027			•
25	0.031	0	0			•
29	0.034	0	0			•
64	0.028	0	0			•
19	0.028	0	0		•	
129	0.028	0	0			•
100	0.028	1	0.081			•
2	0.025	0	0		•	
20	0.038	0	0		•	

Tabla 5.8: Resultados en la ejecución de caminos en el ambiente de la figura 5.1 con 10 obstáculos móviles con velocidad V=2.

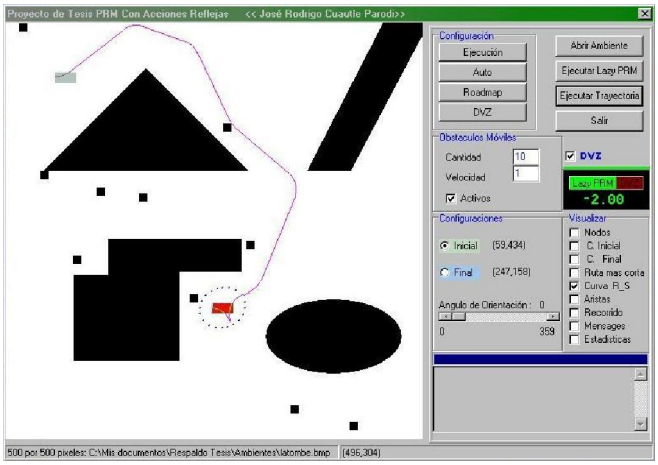

Figura 5.5: Ejecución de caminos en el ambiente de la figura 5.1 con 10 obstáculos dinámicos.

- ϕ: 45°.

Las tablas 5.9 a 5.12 exhiben los resultados obtenidos cuando el robot ejecuta caminos planificados como el mostrado en el ambiente de la figura 5.6.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
17	0.035	0	0		•	
60	0.034	0	0			•
127	0.039	0	0			•
190	0.039	0	0		•	
15	0.027	0	0			•
80	0.028	0	0			•
24	0.030	0	0		•	
9	0.029	0	0			•
177	0.041	0	0			•
20	0.029	0	0			•

Tabla 5.9: Resultados en la ejecución de caminos en el ambiente de la figura 5.2 con 5 obstáculos móviles con velocidad V=1.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
32	0.031	0	0			•
9	0.031	0	0			•
46	0.031	0	0		•	
25	0.034	0	0			•
16	0.029	0	0		•	
78	0.039	1	0.272			•
14	0.028	0	0			•
63	0.031	0	0			•
80	0.054	0	0		•	
14	0.038	0	0		•	

Tabla 5.10: Resultados en la ejecución de caminos en el ambiente de la figura 5.2 con 5 obstáculos móviles con velocidad V=2.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
50	0.035	0	0		•	
79	0.031	0	0			•
35	0.034	0	0			•
89	0.035	0	0			•
247	0.042	2	1.481			•
40	0.031	0	0		•	
209	0.034	1	0.427			•
47	0.034	0	0			•
148	0.039	0	0		•	
100	0.034	0	0			•

Tabla 5.11: Resultados en la ejecución de caminos en el ambiente de la figura 5.2 con 10 obstáculos móviles con velocidad V=1.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
65	0.039	0	0			•
260	0.042	3	0.639			•
93	0.051	1	1.088		•	
49	0.034	0	0		•	
69	0.038	0	1		•	
9	0.041	0	0		•	
74	0.038	0	0		•	
21	0.029	0	0		•	
59	0.041	1	2.445			•
32	0.032	0	0		•	

Tabla 5.12: Resultados en la ejecución de caminos en el ambiente de la figura 5.2 con 10 obstáculos móviles con velocidad V=2.

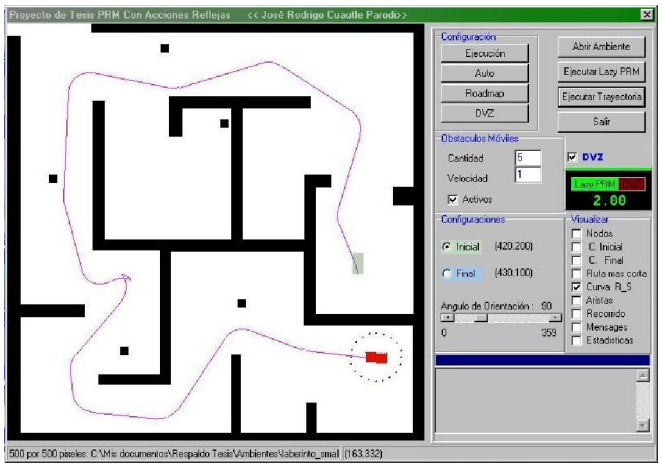

Figura 5.6: Ejecución de caminos en el ambiente de la figura 5.2 con 5 obstáculos dinámicos.

Finalmente la tabla 5.13 muestra el desempeño del robot cuando navega en un ambiente que posee puros obstáculos dinámicos, como se ilustra en la figura 5.7.

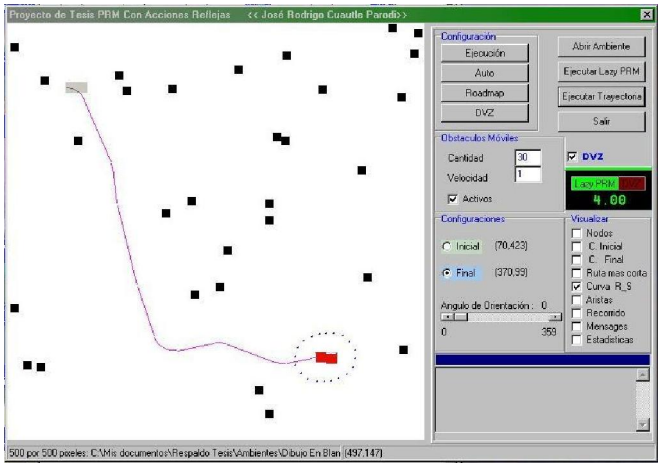

Figura 5.7: Ambiente que solo posee obstáculos dinámicos.

				Resultado		
Reconexiones	Tiempo de reconexión promedio (Seg.)	Replani-ficaciones	Tiempo de replanificación promedio (Seg.)	Camino no encontrado	Colisión	Éxito
96	0.031	1	0.037			•
57	0.034	0	0			•
85	0.027	1	0.009			•
7	0.024	0	0			•
65	0.026	0	0			•
27	0.028	0	0			•
223	0.032	0	0			•
42	0.026	0	0			•
60	0.036	0	0			•
54	0.025	0	0			•

Tabla 5.13: Resultados en la ejecución de caminos en el ambiente de la figura 5.7 con 30 obstáculos móviles con velocidad V=2.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

De acuerdo con el objetivo de este trabajo se tienen las siguientes conclusiones:

Se desarrolló un enfoque reactivo en el marco de trabajo de los roadmaps probabilistas, mediante la integración de un planificador Lazy PRM y de acciones reflejas basadas en el método de la zona virtual deformable.

La ejecución de caminos de robots no holonómicos en ambientes dinámicos es un problema complejo. Los resultados obtenidos en las pruebas aplicadas al método Lazy-PRM reactivo muestran que éste constituye una opción importante como solución a este tipo de problemas. Este trabajo presenta un primer intento de dotar de reactividad a los métodos PRM. No tenemos conocimiento de otros trabajos semejantes.

En efecto, el desempeño del método es satisfactorio: presenta una planificación rápida, acciones reflejas basadas en sensores que no emplean algoritmos costosos, un proceso de reconexión efectivo que emplea sólo algunos milisegundos y un proceso de replanificación poco requerido si los parámetros del método Lazy-PRM reactivo son apropiados.

El tiempo de planificación es reducido debido al empleo del detector de colisiones incompleto, cuya labor es complementada por los sensores del robot durante la ejecución del camino. Por otra parte, la forma de asignar los ángulos de orientación a los nodos que conforman el camino más corto obtenido por el algoritmo A*, produce curvas que permiten omitir el proceso de optimización de las mismas.

Con respecto al proceso de reconexión , ya que los caminos que retorna están conformados por una sola curva Reeds and Shepp, y que se basa también en el detector de colisiones incompleto, el tiempo que requiere es realmente corto y las curvas obtenidas son óptimas.

Además, gracias a que las acciones reflejas las proporciona el método de la ZVD, es posible interrumpir los procesos de reconexión y de replanificación si es necesario, sin implicar mayores problemas.

Así también, si los parámetos de ejecución del método Lazy-PRM reactivo son adecuados, el proceso de replanificación será llamado pocas veces, y si no existen pasajes estrechos, muy rara vez fracasará la replanificación.

Sin embargo existen algunos puntos débiles en el método: la intención de que las acciones reflejas por ZVD siempre se presenten ante obstáculos móviles no se logra debido a que éstos suelen ser confundidos con obstáculos fijos. La causa es que no se utiliza modelo alguno del ambiente, y por tanto no se conoce explícitamente el comportamiento de cada objeto presente. Sólo se suponen objetos móviles aquellos sean detectados por los sensores del robot en una posición que al momento de la última reconexión o replanificación pertenecía al espacio libre.

Otro problema presente es ocasionado por el uso del detector de colisiones incompleto, ya que es posible que el camino planificado pase por un obstáculo estático pequeño, que no ocasionará acción refleja y será embestido por el robot.

6.2. Trabajo futuro

De los resultados obtenidos, se pueden plantear los siguientes puntos como trabajo futuro:

- Probar el método Lazy-PRM reactivo en robots reales.
- Resolver el problema que representan los obstáculos estáticos pequeños.
- Durante las pruebas de evaluación se presentaron casos en los que las acciones reflejas por ZVD no fueron suficientes para evitar colisiones. Estos casos son difíciles porque requieren un comportamiento más inteligente para evitar que el robot quede atrapado. Así por ejemplo, sería útil añadir un bloque que permita calcular las trayectorias de los objetos móviles próximos, para corregir oportunamente el camino en ejecución.

- Un punto de interés actuál en robótica móvil es el caso de los ambientes no estructurados. Por tanto es interesante buscar una extensión del método para aplicarlo en tales casos.

Bibliografía

[1] Kavraki, L. E., Švestka, P., Latombe, J. C., Overmars, M.: "Probabilistic roadmaps for path planning in hig-dimensional configuration spaces", *IEEE Transactions on Robotic and Automation*, Vol. 12, No. 4, pp. 566-580, 1996.

[2] Švestka, P., Overmars, M.: "Motion planning for car-like robots using a probabilistic learning approach", *The International Journal of Robotics Research*, Vol 16, No. 2, pp. 119-143, 1997.

[3] Sánchez, A., Zapata, R.: "A lazy probabilistic roadmap planner for car-like robots", Technical Report LIRMM, 2001.

[4] Latombe, J. C.: "Robot motion planning". Kluwer Academic Publishers, 1991.

[5] Reeds, J. A., Shepp, R. A.: "Optimal paths for a car that goes both forward and backwards", *Pacific Journal of Mathematics*, Vol. 145, No. 2, pp. 367-393, 1990.

[6] Brock, O.: "Generating robot motion: The integration of planning and execution", PhD thesis, Stanford University, 1999.

[7] Zapata, R., Lépinay, P., Thompson, P.: "Reactive behaviors of fast mobile robots", *Journal of Robotic Systems*, Vol. 11, No. 1, pp. 13-20, 1994.

[8] Sánchez, A.: "Zonas virtuales deformables (DVZ)", Technical Report LIRMM, 2001.

[9] Russell, S., Norving, P.: "Inteligencia Artificial Un enfoque moderno", Pearson Education, pp. 839-852, 1996.

[10] Khatib, O.: "Real time obstacle avoidance for manipulators and mobile robots", *IEEE Journal of Robotics and Automation*, Vol. 2, pp. 90-98, 1986.

[11] Lumelsky, V., Stepanov, A.: "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape", *Algorithmica 2*, pp. 403-430, 1987.

[12] Cacitti, A., Zapata, R.: "Reactive behaviours of mobile manipulator based on the DVZ method", International Conference on Robotics and Automation, pp. 680-685, 2001.

[13] Jaillet, L., Siméon, T.: " A PRM-based motion planner for dynamically changing environments", *IEEE International Conference on Intelligent Robots and Systems*, pp. 1606-1611, 2004.

[14] Brock, O., Khatib, O.: "Elastic strips: A framework for integrated planning and execution", *Proceedings of the 1999 International Symposium on Experimental Robotics*, preprints, pp. 245-254, 1999.

[15] Hsu, D., Kindel, R., Latombe, J. C., Rock, S.: "Randomized kinodynamic motion planning with moving obstacles", *The International Journal of Robotics Research*, Vol. 21, No. 3, pp. 233-255, 2002.

[16] Sánchez,A.: "Contribution á la planification de mouvements en robotique: Approches probabilistas et approches déterministes" PhD Thesis, Université Montpellier II, 2003.

[17] Bohlin, R., Kavraki. L.: "Path planning using lazy PRM", *IEEE International Conference on Robotics and Automation*, pp. 521-528, 2000.

[18] Lanzoni, C., Sánchez, A., Zapata, R.: "A single-query motion planner for car-like nonholonomic mobile robots", *ISRA*, 2002.

[19] Cacitti, A., Zapata, R.: "Reactive behaviours of mobile manipulators based on the DVZ approach", *IEEE International Conference on Robotics and Automation*, pp. 680-685, 2001.

[20] Cuautle, R., Sánchez, A., Zapata, R.: "A reactive lazy PRM approach for nonholonomic motion planning", sometido al *International Conference on Robotics and Automation*, 2006.