



Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

**“Estrategias reactivas para la planificación
dinámica de movimientos en robótica móvil”**

Tesis de Licenciatura

Para obtener el título de:
Ingeniero en Ciencias de la Computación

**Presenta
Gilberto Sarabia Cortés**

**Asesor
Dr. Abraham Sánchez López**

Puebla, Pue.

Primavera 2009

Gracias a todos los que hicieron posible este trabajo

Resumen

La planificación de movimientos en robótica es un área de investigación que ha sido ampliamente estudiada estos últimos 30 años. El objetivo de estos trabajos es el de desarrollar métodos algorítmicos que tengan un buen desempeño y a la vez sean efectivos en el cálculo automático de trayectorias para sistemas mecánicos. A pesar de la explosión combinatoria de estos problemas, en los últimos 15 años se han propuesto los métodos probabilistas de planificación de movimientos.

Aún cuando las técnicas de planificación de movimientos han tenido éxitos importantes en la solución de muchos problemas prácticos en robótica en los últimos años, su adaptación a escenas que contengan obstáculos estáticos y obstáculos móviles es realmente muy limitada (en este trabajo de tesis estamos particularmente interesados en problemas en un contexto reactivo). Una de las razones es el costo asociado a la actualización de las estructuras de datos precalculadas con el fin de capturar la conectividad del espacio libre. Nuestra contribución principal es la comparación de dos propuestas de solución ya presentadas por la comunidad de robótica (el método propuesto por Cuautle y Sánchez basado en el algoritmo Lazy PRM y la DVZ y el propuesto por Jaillet-Siméon que esta basado en la combinación de los métodos PRM con los métodos RRT). Esta comparación ha sido interesante pues nos ha llevado a proponer un nuevo algoritmo basado en los árboles aleatorios de exploración rápida y la zona virtual deformable. A este nuevo algoritmo lo hemos llamado RRT-Reactivo.

Índice General

1. Introducción.....	7
1.1. Planificación y consideración del tiempo.....	7
1.2. Planificación y ejecución.....	9
1.3. Planificación y escenas dinámicas.....	11
2. Métodos de planificación probabilista.....	14
2.1. Árboles aleatorios de exploración rápida.....	14
2.1.1. Formulación del problema.....	14
2.1.2. Planificadores RRT simples.....	18
2.1.3. Planificadores bidireccionales.....	20
2.1.4. Planteamientos adicionales con RRT.....	21
2.1.5. Aplicación de los RRTs en robots tipo carro.....	22
2.1.5.1. Modelo cinemático no-lineal para un robot tipo carro.....	22
2.2. Lazy PRM.....	24
2.2.1. Construcción del roadmap inicial.....	25
2.2.2. Búsqueda del camino más corto.....	26
2.2.3. Revisión del camino encontrado.....	26
2.2.3.1 Revisión de nodos.....	27
2.2.3.2 Revisión de arcos.....	27
2.2.4. Enriquecimiento del roadmap.....	28
2.2.5. Prueba de Colisión para un robot tipo carro.....	29
2.2.6. Curvas de Reeds & Shepp.....	30
3. Método reactivo de planificación basado en Lazy PRM y la DVZ.....	34
3.1. Zona Virtual Deformable.....	34
3.1.1. Ejemplo simplificado de acción refleja de paro.....	35
3.1.2. Formalismo del método.....	36
3.1.3. Zona virtual deformable no deformada.....	37

3.1.4. Intrusiones y deformaciones debidas al ambiente.....	38
3.1.5. Ecuación de estado.....	40
3.1.6. Generación de comandos reflejos.....	41
3.1.7. Aplicaciones.....	42
3.2. Lazy PRM reactivo.....	43
3.2.1. Planificador Lazy PRM.....	43
3.2.2. Construcción del roadmap inicial.....	44
3.2.3. Búsqueda del camino más corto.....	45
3.2.4. Revisión de colisiones.....	45
3.3. Control Reactivo por ZVD.....	46
3.4. Reconexión.....	48
3.5. Replanificación.....	48
3.6. Simulación de la cinemática del robot.....	49
3.7. Distinción entre obstáculos móviles y fijos.....	51
4. Otros métodos reactivos.....	52
4.1. Ejemplo Introductivo.....	52
4.2. Principio del enfoque.....	54
4.2.1. Red Inicial.....	55
4.2.2. Consulta de planificación.....	57
4.3. Actualización de la red en las consultas de planificación.....	58
4.3.1. Conexiones a la red.....	58
4.3.2. Búsqueda de un camino.....	61
4.3.3. Reconexiones locales.....	63
4.3.4. Enriquecimiento de la red.....	64
4.3.5. Etiquetado de los arcos.....	65
4.4. RRT reactivo.....	67
4.4.1. Revisión de colisiones.....	68
4.4.2. Replanificación.....	68
5. Resultados.....	69

5.1 Evaluación del planificador Lazy PRM.....	69
5.2 Evaluación del planificador RRT.....	73
5.3 Evaluación de Lazy PRM Reactivo.....	75
5.4 Evaluación de RRT Reactivo.....	77
5.4.1 Reconexión simple.....	78
5.4.2 Reconexión RRTDual.....	80
6. Conclusiones y trabajo futuro.....	82
Bibliografía.....	85

Capítulo 1

Introducción

La planificación de movimientos es un problema clásico de la robótica que ha suscitado numerosos trabajos después de los trabajos pioneros del MIT [Lozano-Perez 83]. La consideración de las escenas dinámicas que es objeto de esta tesis corresponde a problemas más difíciles que el caso de escenas estáticas consideradas en la mayoría de los trabajos.

Los algoritmos de planificación de movimientos consideran en la mayor parte de los casos, ambientes estáticos perfectamente conocidos. En muchas situaciones, las modificaciones de la escena necesitan por lo tanto considerarse en el cálculo del movimiento. Se habla de manera general de problemas en ambientes dinámicos. Los trabajos dirigidos a este tipo de situaciones se orientan particularmente alrededor de tres grandes problemáticas.

La primera considera las situaciones en las cuales, la evolución en el tiempo de la posición de los obstáculos móviles se supone conocida. En este caso, la consideración de los obstáculos se puede hacer a partir de la fase de planificación. La segunda problemática se refiere a la navegación de los robots, que necesitan conjugar métodos de planificación con técnicas de ejecución de trayectorias. La última problemática es justamente el objetivo de la tesis. El objetivo es desarrollar métodos de planificación más poderosos para considerar los diferentes cambios susceptibles de presentarse en el ambiente.

1.1 Planificación y consideración del tiempo

En el caso de un ambiente 3D que contiene muchos obstáculos móviles que siguen las trayectorias conocidas, se sabe que el problema es PSPACE-difícil si la velocidad máxima está limitada, y NP-difícil en el caso contrario [Reif 85]. El planificador debe entonces calcular una trayectoria parametrizada en tiempo, en lugar de un simple camino geométrico.

En [Kant 86], se propuso un enfoque de dos fases. Inicialmente, los obstáculos móviles se ignoran y se calcula un camino válido considerando los obstáculos estáticos. En una segunda etapa, la velocidad del robot a lo largo del camino se controla para evitar las colisiones con los obstáculos. El planificador proporciona buenos resultados cuando el número de obstáculos móviles es bajo y/o el ambiente no contiene muchos obstáculos. Para aumentar las potencialidades del planificador, se propuso en [Fujimua 95] extender el método generando una red de caminos.

La noción del espacio de las velocidades denotada como V_{obst} se introdujo en [Shiller96]. De un punto de vista general, V_{obst} es el espacio de las velocidades como el C_{obst} es el espacio de configuraciones: este designa el conjunto de las velocidades instantáneas del robot que entran en colisión en un tiempo acotado que se denomina horizonte de tiempo. Este concepto se utiliza para generar, para el robot, las trayectorias que tienen un perfil de velocidades que no pertenecen a V_{obst} en el caso donde los obstáculos se desplazan en línea recta y a velocidad constante. Recientemente, [Large 03], presentó una expresión analítica de los límites de V_{obst} , que permite extender el método a cualquier tipo de trayectorias de los obstáculos y remplazar la información booleana de colisión o no-colisión por el tiempo de colisión asociado a cada velocidad.

La noción del espacio de configuraciones $\times$ tiempo se introdujo en [Erdmann 87]. Este espacio corresponde al espacio de configuraciones C aumentado en una dimensión que representa el tiempo. Una extensión a restricciones kinodinámicas [Fraichard 95] introduce la noción del espacio de estados $\times$ tiempo, donde el estado de un robot está definido por la unión de sus parámetros de configuración y de sus derivadas (las primeras y eventualmente las segundas) en relación con el tiempo. El uso de los estados $\times$ tiempo ha sido igualmente retomado en [Hsu 02]. El planificador difunde en este espacio un árbol de búsqueda en vías de conectar los puntos estado-tiempo, dados como entradas del problema. Un análisis algorítmico muestra la convergencia en probabilidad del método bajo ciertas hipótesis relacionadas con la geometría del espacio de estados-tiempo. El planificador se utiliza para replanificar en línea la trayectoria de un robot circular rodeado de muchos obstáculos móviles.

Otro enfoque propuesto en [van den Berg 04] utiliza un algoritmo de dos etapas. En la primera, una red (roadmap) que contiene ciclos se calcula sin tomar en cuenta los obstáculos dinámicos. Enseguida, a partir de esta red, una trayectoria valida se calcula para una consulta dada. La trayectoria se determina difundiendo las restricciones temporales a lo largo de la red y buscando un camino más corto sobre una rejilla en el espacio de estados-tiempo. Una extensión a problemas multi-robot se propuso también en [van den Berg 05].

1.2 Planificación y ejecución

En muchos trabajos aplicados a la navegación de robots móviles, los métodos de planificación se combinan con métodos de evitamiento de obstáculos, como el método de potenciales [Khatib 86]. Estos métodos, generalmente basados en un razonamiento local, tienen como objetivo asegurar la seguridad del robot a pesar de los riesgos ligados a la ejecución del movimiento. Aunque muy reactivos, estos poseen sin embargo un inconveniente más importante: el problema de los mínimos locales y pueden conducir a situaciones de bloqueo en las cuales el robot es incapaz de alcanzar su posición objetivo. Muchos métodos de evitamiento de obstáculos existen en la literatura, citamos los más importantes: histograma de campos vectoriales [Borenstein 91, Ulrich 98, Ulrich 00], las técnicas de velocidad de curvatura [Simmons 96, Ko 98], la ventana dinámica [Fox 97, Brock 99], y los diagramas de proximidad [Minguez 00].

Otro enfoque establece la liga entre planificación y ejecución. Se trata de la técnica de la banda elástica [Quinlan 93]. El principio consiste en mantener en permanencia un pasaje sin colisión (llamado banda) entre el robot y su objetivo. Este pasaje se representa con un conjunto de discos (2D) o de bolas (3D) del espacio libre, llamadas burbujas. El diámetro de estas burbujas puede crecer o reducirse para cubrir adecuadamente el espacio libre, en función de los cambios del ambiente. Su número puede igualmente variar para evitar toda interrupción de la banda. Dos tipos de fuerzas entran en acción para mantener su coherencia: fuerzas atractivas entre burbujas consecutivas con la finalidad de mantener la banda “tendida” y las fuerzas repulsivas con los obstáculos para mantener la banda alejada de estas últimas.

El equilibrio de estas fuerzas tiende hacia un camino seguro hasta el objetivo. La actualización de las fuerzas asegura la adaptación de este camino en tiempo real en función de los obstáculos. El enfoque, inicialmente desarrollado para planificar caminos geométricos en tiempo real [Quinlan 93], se ha extendido en [Khatib 97] a robots no-holonómicos utilizando un método local de tipo Reeds & Shepp. Para administrar las “rupturas” de la banda, necesarias en el caso de un obstáculo que pasa ortogonalmente a la trayectoria y para evitar que la banda se reduzca, una extensión se propuso bajo el término de tira elástica [Brock 00]. La idea es de aflojar temporalmente la restricción entre dos burbujas adyacentes para dejar pasar un obstáculo móvil. Además, para limitar los procesos de replanificación completa, se conserva un conjunto de bandas de seguridad, las cuales sirven de soluciones alternativas. En [Lamiriaux 04], se propuso un método genérico de deformación reactiva de trayectorias para robots no-holonómicos, este se basa en un mecanismo de perturbación de los controles de entrada del sistema robótico utilizado.

Los métodos híbridos, combinan en el mismo algoritmo estrategias de planificación de trayectorias y métodos de evitamiento de obstáculos. Algunos de estos métodos corresponden a una extensión de los métodos de evitamiento de obstáculos en una versión global. Así en [Brock 99], se propuso generalizar la técnica de las ventanas dinámicas, acoplándola con los métodos estándar de planificación. De la misma manera, la extensión de los diagramas de proximidad en método global se propuso en [Minguez 02]. En [Large 04], se utiliza la representación de las trayectorias de los obstáculos en el espacio de velocidades instantáneas del robot, para evitar los obstáculos dinámicos. Esta estrategia enseguida se combina con técnicas de expansión de grafos y de técnicas de predicción de movimientos de los obstáculos, permitiendo al final construir un planificador global incremental. Otros métodos se basan en las técnicas de planificación y las adaptan para controlar la ejecución de una trayectoria. Es el caso por ejemplo de los trabajos presentados en [Hsu 02, Bruce 02]. En [Petti 05], se busca construir movimientos seguros, planificando de manera iterativa asegurando a cada intervalo de tiempo que no se alcanzan estados para los cuales una colisión sería inevitable.

Otro método [Brock 01] intenta explotar la información de conexidad ligada al espacio de trabajo W para guiar la búsqueda en el espacio de configuraciones C . El algoritmo se descompone en dos etapas. La primera consiste en calcular un volumen libre de W que contenga al menos un camino solución. Este cálculo se hace con la ayuda de un algoritmo de propagación por olas de bolas libres de W a partir de la posición inicial. En una segunda etapa, un camino solución se calcula con la ayuda de técnicas de campos de potencial luego una función de navegación que se apoya en el túnel anteriormente calculado adapta la trayectoria en función de los obstáculos móviles.

En general, las estrategias que integran planificación y ejecución funcionan adecuadamente para los problemas donde un camino solución suficientemente alejado de los obstáculos puede fácilmente determinarse. Estos suponen además, una cierta semejanza entre la conexidad de W y la de C . Por estas razones, estos no están adaptados a sistemas mecánicos complejos, ni a los ambientes con muchos obstáculos. Este trabajo presentado en la tesis es el contrario de estos métodos, donde la dinámica es medianamente considerada, pero se consideran sistemas más complejos.

1.3 Planificación y escenas dinámicas

Para tomar en cuenta los cambios de posición de los objetos en las escenas dinámicas, los métodos de planificación basados en muestreo (PRM o RRT) deben considerar una fase de actualización durante la cual la validez de los elementos del grafo sean probados por colisión. Tomando en cuenta las dificultades temporales generalmente fuertes para los problemas dinámicos, estos métodos deben ser especialmente eficaces. Una primera idea es utilizar las técnicas de mapeo entre el ambiente W y el espacio de configuraciones C . Estas técnicas construyen una estructura de datos que hace corresponder las células elementales del ambiente (píxeles en 2D, voxels en 3D) a los diferentes nodos y arcos del grafo. Esta construcción se hace escaneando el espacio de trabajo a partir de un obstáculo elemental del tamaño de las células y probando para cada posición tomada por este obstáculo en el ambiente, cuales son las porciones de la red afectadas.

En la actualización del grafo, las células del ambiente invalidadas permiten determinar sin tener que realizar nuevas pruebas de colisión, cuales son los nodos y arcos afectados. Un método “primitivo” de mapeo se presentó en [Lean 96a]. Este construye una estructura que memoriza las porciones del grafo invalidadas para las diferentes posiciones potenciales de un obstáculo que puede aparecer o desaparecer en la escena.

Leven [Leven 00] propone la primera representación del espacio a través de una estructura de datos, la cual la aplica a robots manipuladores planos. Una limitación del método es que no necesita un tamaño de memoria importante para almacenar la información de mapeo. Para intentar reducir el tamaño de memoria necesario, se propuso un método de compresión de datos basado en la coherencia espacial propuesta en [Leven 02]. Una limitación importante es la descomposición del espacio en células que hacen que el método sea completo para una resolución dada.

Otra idea consiste en utilizar el principio de redes perezosas [Bohlin 00, Sánchez 03] igualmente utilizadas para los problemas de ambientes estáticos. Para estas redes, la prueba de validez de los arcos y eventualmente de los nodos no se realiza más que en el momento de la consulta. En ambientes dinámicos, estas pruebas se hacen directamente al contexto actual. Así esta estrategia permite evitar las actualizaciones inútiles para contextos diferentes del contexto de consulta. Además, el método utilizado se basa en una actualización de la red solamente parcial, lo que permite de hacer al enfoque más interesante en los problemas de escenas dinámicas donde las restricciones de eficiencia son fuertes. Estos métodos nos distinguen la escena estática de los obstáculos móviles, y no explotan el carácter invariante de la parte estática.

El trabajo de Lean [Lean 96] propone utilizar una método de reparación de las porciones invalidas, después de una etapa de actualización. Este método repara los caminos del grafo inicial proyectándolos fuera de los obstáculos de las configuraciones invalidadas con la ayuda de un movimiento browniano sesgado hacia las regiones exteriores. Sin embargo, este método “ad hoc” utilizado para resolver un problema específico en ambientes industriales se limita a los robots rígidos simples.

La estructura de los grafos utilizados en los problemas de planificación en ambientes dinámicos es igualmente muy importante. En la literatura, esta pregunta es por lo tanto muy poco considerada. Las redes, generalmente utilizadas [Leven00, Bohlin 00, Sánchez 03, Lean 96] están construidas a partir de las estrategias tradicionales de conexión (k -nodos los más cercanos, distancia máxima) utilizados para construir redes en ambientes estáticos. La estructura final es generalmente densa y muy redundante, lo que permite capturar los diferentes caminos del espacio libre pero genera costos de actualización más elevados. El criterio de ϵ -robustez [Leven 00] se ha propuesto para estimar la robustez de las redes respecto a los obstáculos móviles: una red es ϵ -robusta si ningún obstáculo de radio igual o inferior a ϵ no puede romper la conexidad. En la práctica, la evaluación de este parámetro es muy difícil.

Capítulo 2

Métodos de planificación probabilista

2.1 Árboles aleatorios de exploración rápida

En esta sección, presentaremos los progresos actuales en el diseño y análisis de algoritmos de planificación de caminos basados en árboles aleatorios de exploración rápida (RRT), técnica desarrollada por Steven M. LaValle y su grupo de colaboradores en la universidad de Illinois, EUA, [LaValle 98, LaValle 00, LaValle 01a, LaValle 01b, LaValle 01c]. La base de estos métodos es la construcción incremental de árboles de búsqueda que intentan explorar rápida y uniformemente el espacio de estados, ofreciendo beneficios similares a los obtenidos por otros métodos exitosos de planificación aleatoria, como los métodos de roadmap probabilísticos (PRM) [Kavraski 96, Overmars 95]. Además, los RRTs son, particularmente, convenientes para problemas que involucran restricciones diferenciales. Expondremos las propiedades teóricas básicas para planificadores que usan RRTs. La discusión y comparación de varios planificadores que usan este enfoque será expuesta en el capítulo de resultados experimentales.

2.1.1 Formulación del problema

El tipo de problemas considerados por el enfoque RRT están formulados en términos de seis componentes:

1. **Espacio de estados:** Un espacio topológico, X .
2. **Valores limite:** $x_{init} \in X$ y $X_{goal} \subset X$

3. **Detector de colisión:** Una función, $D : X \rightarrow \{\text{true}, \text{false}\}$, que determina si las restricciones globales son satisfechas desde el estado x . Esta podría ser una función binaria o real.
4. **Entradas:** Un conjunto, U , que especifica el conjunto de controles o acciones que pueden afectar al estado.
5. **Simulador incremental:** Dado el actual estado, $x(t)$, y las entradas aplicadas sobre un intervalo de tiempo, $\{u(t') | t \leq t' \leq t + \Delta t\}$, calcular $x(t + \Delta t)$.
6. **Métrica:** Una función real, $\rho : X \times X \rightarrow [0, \infty)$, la cual especifica la distancia entre pares de puntos en X .

La planificación de caminos generalmente es vista como una búsqueda en el espacio de estados, X , para un camino continuo desde un estado inicial, x_{init} a una región meta $X_{meta} \subset X$ o un estado meta $x_{meta} \in X$. Se asume que se tiene un conjunto complicado de restricciones diferenciales sobre X y cualquier camino solución debe mantener al estado dentro de este conjunto. Un detector de colisión reporta si un estado dado x , satisface las restricciones globales. Generalmente, se utiliza la notación, X_{free} para referirse al conjunto de estados que satisfacen las restricciones globales. Se asignan restricciones locales y diferenciales a través de un conjunto de entradas (o controles) y de un simulador incremental. Estos dos componentes especifican los posibles cambios en el estado. El simulador incremental puede definirse por integración numérica de una *ecuación de transición de estado*. Finalmente, se define una métrica para indicar la cercanía de pares de puntos en el espacio de estados.

El árbol aleatorio de exploración rápida (RRT, del inglés, Rapidly-Exploring Random Tree) fue presentado en [LaValle 98] como un algoritmo de planificación para búsqueda rápida en espacios de altas dimensiones que tienen tanto restricciones algebraicas (provenientes de los obstáculos) como restricciones diferenciales (originadas por la no-holonomía y la dinámica). La idea clave es dirigir la exploración hacia regiones no exploradas del espacio tomando puntos en el espacio de estados e incrementalmente “jalar”

el árbol hacia ellos. Por lo menos, se han propuesto otras dos técnicas aleatorias de planificación de caminos (planificación holonómica): el algoritmo de Hilo de Ariadna [Mazer 98] y los planificadores en [Yu 00]. Intuitivamente, estos planificadores intentan “empujar” la búsqueda lejos de los vértices previamente construidos, en contraste, con el RRT el cual usa el espacio circunvecino para “jalar” la búsqueda. Hasta donde llega nuestro conocimiento, no se había propuesto un método aleatorio con árboles de búsqueda para planificación no-holonómica o kinodynamic. Quizá, los métodos más aproximados son [Sekhavat 98, Svestka 95], en donde, el método roadmap probabilístico se combina con técnicas de conducción no holonómica para planificar caminos con robots móviles rodantes.

```
CONSTRUIR_RRT( $x_{init}$ )
1   $T.init(x_{init});$ 
2  para  $k = 1$  a  $K$ 
3     $x_{rand} \leftarrow ESTADO\_ALEATORIO();$ 
4     $EXTENDER(T, x_{rand});$ 
5  Regresa  $T$ 
```

```
EXTENDER( $T, x$ )
1   $x_{near} \leftarrow VECINO\_MAS\_PROXIMO(x, T);$ 
2  si  $NUEVO\_ESTADO(x, x_{near}, x_{new}, u_{new})$  entonces
3     $T.agrega.vertice(x_{new});$ 
4     $T.agrega.arista(x_{near}, x_{new}, u_{new});$ 
5    si  $x_{new} = x$  entonces
6      Regresa Alcanzado;
7    sino
8      Regresa Avanzado;
9  Regresa Atrapado;
```

Figura 2.1. Algoritmo básico de construcción del RRT

El algoritmo básico de construcción de RRT, se muestra en la figura 2.1. En cada iteración se intenta extender el árbol agregando un nuevo vértice en dirección a un estado seleccionado aleatoriamente. La función EXTENDER, ilustrada en la figura 2.2, selecciona del árbol el vértice más cercano a un estado dado. Este vértice se elige de acuerdo a una

métrica, p . La función NUEVO_ESTADO hace un movimiento hacia x aplicando una entrada $u \in U$ para algún incremento Δt . Esta entrada puede seleccionarse aleatoriamente o probando todas las posibles entradas eligiendo aquella que produzca un nuevo estado tan próximo como sea posible a x (si U es infinito puede usarse una aproximación o una técnica analítica). NUEVO_ESTADO utiliza implícitamente una función de detección de colisiones para determinar si el nuevo estado (y todos los estados intermedios) satisfacen las restricciones globales. Si NUEVO_ESTADO se cumple, el nuevo estado junto con la entrada, se representan por medio de x_{new} y u_{new} , respectivamente. Pueden ocurrir tres situaciones: *Alcanzado*, el nuevo vértice alcanza al estado muestreado x , (para la planificación no-holonómica, tendríamos un umbral, $\|x_{new} - x\| < \varepsilon$ para un ε pequeño, $\varepsilon > 0$); *Avanzado*, un nuevo vértice $x_{new} \neq x$ es agregado al RRT. *Atrapado*, NUEVO_ESTADO falla en producir un nuevo estado que se encuentre en X_{free} .

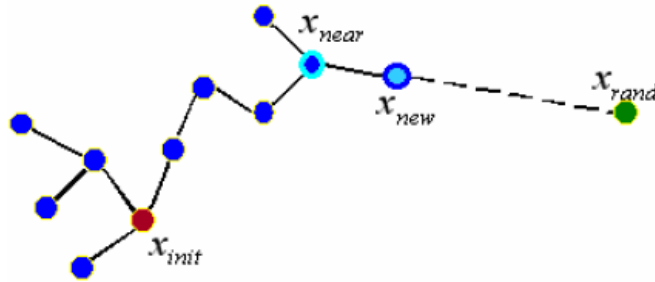


Figura 2.2. Operación de la función EXTENDER

Generalmente consideramos al RRT como un bloque en la construcción de un planificador eficiente. Por ejemplo, podríamos usar un RRT para escapar de un mínimo local en un planificador de caminos aleatorio con campos de potencial. En [Vallejo 00], un RRT fue usado como planificador local para el planificador roadmap probabilístico. Presentamos varias alternativas de planificadores que utilizan un RRT. La elección recomendada depende de varios factores, tal como, si existieran restricciones diferenciales, el tipo de algoritmo para la detección de colisiones, o la eficiencia en el cómputo del vecino más próximo.

2.1.2 Planificadores RRT simples

En principio, el **RRT básico** puede usarse como un planificador de caminos por que sus vértices, eventualmente, cubrirán un componente conectado de X_{free} , llegando arbitrariamente cerca de cualquier x_{goal} especificado. El problema es que sin ningún sesgo hacia el objetivo, la convergencia es lenta. Un planificador mejorado, llamado **RRT_GoalBias**, se obtiene reemplazando la función ESTADO_ALEATORIO en la figura 2.1 con otra que elija, con cierta probabilidad, entre x_{goal} o un estado tomado de cualquier parte del espacio de estados. Incluso con una probabilidad pequeña (p.ej. 0.05) de elegir x_{goal} , RRT_GoalBias usualmente converge al objetivo mucho más rápido que el RRT básico. Pero, si se designa un sesgo muy grande hacia el objetivo entonces el planificador empieza a comportarse como un planificador aleatorio de campos de potencial, ya que pueden presentarse mínimos locales. Una mejora es, **RRT_GoalZoom** el cual reemplaza la función ESTADO_ALEATORIO con una decisión, basada en cierta probabilidad, de optar por un estado aleatorio dentro de una región circunvecina al objetivo o por uno elegido del espacio de estados completo. El tamaño de la región alrededor del objetivo lo controla el vértice del árbol más cercano al objetivo. El efecto es que el foco de muestreo converge al objetivo como el RRT se acerca a él. Este planificador funciona bastante bien en la práctica, sin embargo, su rendimiento aún puede degradarse debido a mínimos locales. En general, parece mejor sustituir la función ESTADO_ALEATORIO con un esquema de muestreo que extraiga los estados desde una función de densidad de probabilidad no-uniforme con un sesgo gradual hacia el objetivo. La figura 2.3 muestra un ejemplo de un RRT construido al muestrear los estados usando una función de densidad de probabilidad que asigna una probabilidad equitativa a anillos concéntricos. Hay muchas investigaciones interesantes con respecto al problema del muestreo.

Un problema más a considerarse, es el tamaño del paso usado en la construcción del RRT. Este podría elegirse dinámicamente durante la ejecución, acorde con la función que calcula la distancia en la detección de colisión. Si los cuerpos están lejos de colisionar, se

puede tomar un paso grande. Además de seguir esta idea, ¿qué tan lejos debe estar x_{new} de x_{near} ?, ¿Trataríamos de conectar x_{near} con x_{rand} ?. En lugar de intentar extender un RRT por un paso incremental, EXTENDER puede ejecutarse iterativamente hasta alcanzar al estado aleatorio o un obstáculo, como se muestra en descripción del algoritmo CONECTAR de la figura 2.4. Al sustituir EXTENDER por CONECTAR el árbol crece muy rápido, si lo permiten las restricciones de detección de colisión y las restricciones diferenciales. Una de las ventajas clave de la función CONECTAR es que construimos un camino tan largo como sea posible con una sola llamada al algoritmo del vecino más próximo. Esta ventaja motiva la elección de un algoritmo voraz.

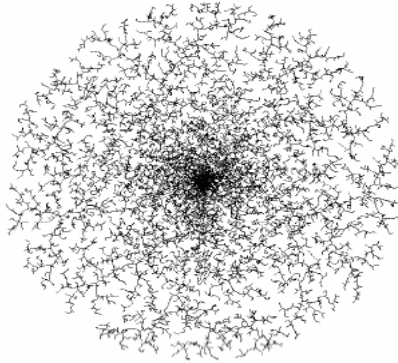


Figura 2.3. Un RRT en 2D construido usando un muestreo sesgado

```
CONECTAR( $T, x$ )
1  repetir
2     $S \leftarrow \text{EXTENDER}(T, x)$ ;
3  hasta que no ( $S = \text{Avanzado}$ )
4  Regresa  $S$ 
```

Figura 2.4. Función CONECTAR

La experiencia ha demostrado que la función CONECTAR produce mejores resultados en problemas holonómicos y la función EXTENDER es sobresaliente en problemas no-holonómicos. Una razón para esta diferencia es que CONECTAR confía más en la métrica, y los problemas no-holonómicos requieren del diseño de buenas métricas.

2.1.3 Planificadores bidireccionales

Inspirados en las técnicas clásicas de búsqueda bidireccional, parece razonable esperar un mejor desempeño al crecer dos árboles de exploración, uno desde x_{init} y el otro a partir de x_{goal} ; se obtiene una solución cuando los dos árboles se encuentran. Para una búsqueda simple, la implementación es directa, sin embargo, la construcción RRT debe guiarse para asegurar que ambos árboles se encuentren antes de cubrir el espacio entero y permitir una unión eficaz.

```
RRT_BIDIRECCIONAL( $x_{init}, x_{goal}$ )
1   $T_a.inic(x_{init}); T_b.inic(x_{goal});$ 
2  para  $k = 1$  a  $K$ 
3     $x_{rand} \leftarrow ESTADO\_ALEATORIO();$ 
4    si ( $EXTENDER(T_a, x_{rand}) < > Atrapado$ ) entonces
5      si ( $EXTENDER(T_b, x_{new}) = Alcanzado$ ) entonces
6        Regresa CAMINO( $T_a, T_b$ );
7    INTERCAMBIAR( $T_a, T_b$ );
8  Regresa Fallo;
```

Figura 2.5. Planificador bidireccional usando un RRT

La figura 2.5 muestra el algoritmo **RRT_BIDIRECCIONAL**, puede compararse con el algoritmo **CONSTRUIR_RRT** de la figura 2.1. El RRT_BIDIRECCIONAL divide el tiempo de cómputo entre dos procesos: 1) explorar el espacio de estados; 2) intentar crecer los árboles uno hacia el otro. Siempre existen dos árboles T_a y T_b , hasta que se enlazan y se encuentra una solución. En cada iteración un árbol crece, y se intenta conectar el nuevo vértice con aquel más cercano en el otro árbol. Entonces, se invierten los roles intercambiando los árboles. El crecimiento de dos RRTs también fue propuesto en [LaValle 01c] para planificación kinodynamic, en ese enfoque, en cada iteración ambos árboles crecían hacia un estado aleatorio. El algoritmo actual intenta que los árboles crezcan uno hacia el otro en la mitad de tiempo, con lo cual se obtiene un buen rendimiento.

Se consideran algunas variaciones del planificador anterior. Puede reemplazarse cualquier ocurrencia de EXTENDER con CONECTAR en el RRT_BIDIRECCIONAL. Cada reemplazo hace a la operación más agresiva. Si sustituimos EXTENDER por CONECTAR en la línea 4, entonces el planificador explora agresivamente el espacio de estados, con la misma compensación que el RRT básico. Si reemplazamos EXTENDER con CONECTAR en la línea 5, el planificador intenta unir los árboles agresivamente en cada iteración. Esta variante sería muy exitosa para resolver problemas de planificación holonómica, por comodidad la denominamos **RRT_ExtCon** y al algoritmo original como **RRT_ExtExt**. Entre las variantes discutidas hasta el momento, encontramos a RRT_ExtCon más eficiente para problemas holonómicos [LaValle 00] y a RRT_ExtExt ideal para problemas no-holonómicos. El planificador más agresivo que se puede construir es sustituyendo las dos ocurrencias de EXTENDER por CONECTAR, líneas 4 y 5, originando RRT_ConCon.

Concluimos, que el enfoque bidireccional es mucho más eficiente que el RRT básico. Una limitación al utilizar un RRT bidireccional en problemas de planificación no-holonómicos y kinodynamic es que necesitamos conectar varios vértices para unir un RRT al otro. Para un problema de planificación que involucre alcanzar una región meta desde un estado inicial, no se necesita conectar ningún vértice. El espacio entre las dos trayectorias puede cerrarse, en la práctica usando, si es posible, métodos de conducción o métodos clásicos de disparo (shooting), presentes con frecuencia en problemas de valores en la frontera.

2.1.4 Planteamientos adicionales con RRT

Si un enfoque dual ofrece ventajas sobre un árbol simple, entonces es natural preguntar si el crecimiento de tres o más árboles ofrecería mayores ventajas. Estos árboles adicionales, iniciarían en estados aleatorios. Por supuesto, los problemas de conexión en el caso de la planificación no-holonómica serían aún más difíciles de resolver. El tiempo de cómputo debería dividirse entre intentar extender los árboles y tratar de conectarlos unos a

otros. Muchas cuestiones quedan por investigar en este y otros planificadores que usen un RRT.

Es interesante considerar el caso límite, en el cual se construye un nuevo RRT para cada estado aleatorio x_{rand} . Una vez generado un nuevo vértice, puede aplicarse la función CONECTAR, de la figura 2.4, a cada RRT. Para mejorar el rendimiento, podemos considerar los vértices que están a una distancia fija de x_{rand} , de acuerdo con la métrica. Si una conexión tiene éxito entonces los dos árboles se unen en un único grafo. El algoritmo resultante, simula el comportamiento del roadmap probabilístico. Por lo que, el roadmap probabilístico puede considerarse como una versión extrema de un algoritmo que usa RRTs, donde se construye un número máximo de RRTs independientes.

2.1.5 Aplicación de los RRTs en robots tipo carro

El diseño de trayectorias para automóviles es un problema importante tanto en robótica como en el desarrollo de prototipos virtuales. En robótica, se necesitan algoritmos que puedan calcular trayectorias para vehículos autónomos en ambientes complicados. En la industria automovilística, los simuladores se utilizan exhaustivamente para evaluar el rendimiento de los vehículos a través de prototipos virtuales. La mayoría de estas simulaciones involucran estrechamente al operador humano con el sistema de simulación. Los planificadores RRTs podrían utilizarse en el diseño de prototipos virtuales para automóviles, considerándolos como un “conductor virtual de prueba” que intenta lograr las condiciones especificadas, como conducir a través de un camino lleno de obstáculos.

2.1.5.1 Modelo cinemático no-lineal para un robot tipo carro

Las restricciones diferenciales presentadas en esta sección son únicamente cinemáticas. Por lo cual, el espacio de estados se reduce al espacio de configuraciones (conjunto de todas las transformaciones que pueden aplicarse al robot). El modelo es tri-dimensional, ver

la figura 2.6, cualquier estado se representa por medio de (x, y, θ) y su ecuación de transición de estado es:

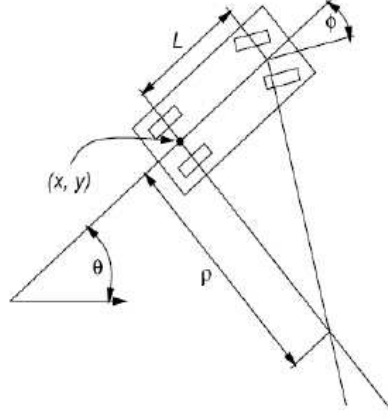


Figura 2.6. Modelo cinemático para un robot tipo carro

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} s \cos \theta \\ s \sin \theta \\ \frac{s}{L} \tan \phi \end{pmatrix} \quad (2.1)$$

Donde, L denota la distancia entre el eje frontal y el trasero, s es la velocidad y ϕ representa al ángulo de conducción. Este último está limitado, $|\phi| \leq \phi_{\max} < \frac{\pi}{2}$. El vector de entrada es (s, ϕ) . Si el carro viaja sólo hacia delante, fijamos a $s = 1$, obteniendo así el carro Dubins [Latombe 91]. Si el carro se mueve hacia atrás y hacia delante, $s = -1$ o $s = 1$, obtenemos un carro Reeds-Shepp.

En el modelo de la ecuación 2.1, el ángulo de conducción ϕ es una entrada, esto implica que podemos mover instantáneamente las llantas frontales. En muchas aplicaciones esta suposición es poco realista. Hay una curvatura discontinua, en el camino trazado por el centro del eje trasero del carro, generada cuando el ángulo de conducción cambia discontinuamente. Podemos obtener un modelo de carro que genere caminos suaves, “modelo smooth”, agregando el ángulo de conducción como variable de estado. La entrada es la velocidad angular, ω , del ángulo de conducción.

El resultado es un espacio de estados de cuatro dimensiones, en el cual cada estado se representa por (x, y, ϕ, θ) , con su consiguiente ecuación de transición:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} s \cos \theta \\ s \sin \theta \\ \omega \\ \frac{s}{L} \tan \phi \end{pmatrix} \quad (2.2)$$

La nueva entrada representa un cambio en el ángulo de conducción.

2.2 Lazy PRM

Lazy PRM [Sánchez 01a] 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 requeridas. Es decir, al momento de realizar la búsqueda en el grafo.

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 aun queda tiempo es posible agregar más nodos al roadmap y buscar nuevamente. Una descripción de alto nivel se muestra en la figura 2.9.

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.

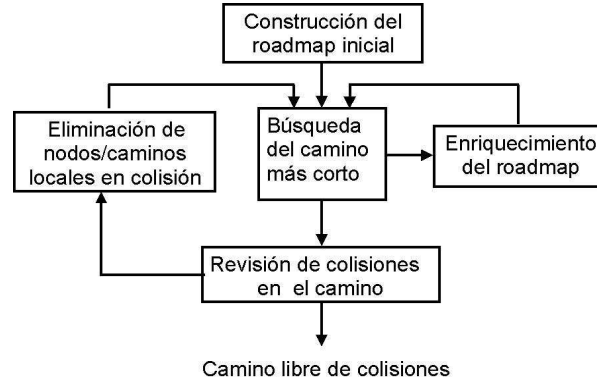


Figura 2.9: Descripción de alto nivel del algoritmo lazy PRM

2.2.1 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 construye 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 ← configuración aleatoria
si c ∈ CS entonces
    V ← V ∪ {c}
    Vc ← un conjunto de vecinos de c seleccionados de V
    ∀ v ∈ Vc, en orden ascendente de d(v, c) hacer
        si ¬conectado (c, v) ∧ L(c, v) ⊂ CS entonces
            E ← E ∪ {(c, v)}
end
```

2.2.2 Búsqueda del camino más corto

El segundo procedimiento en el algoritmo es encontrar el camino mas corto en *G* entre *c_i* y *c_f*, o determinar que no existe alguno. Para este propósito se puede utilizar el algoritmo A* [Latombe 91]. 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 cual agrega nuevos nodos, teniendo en cuenta el tiempo permitido para resolver el problema.

2.2.3 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 2.10). 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.

El propósito general de los pasos de Búsqueda, Revisión y Eliminación del algoritmo (Figura 2.9), es el identificar y remover tanto nodos como arcos en colisión hasta que el camino mas corto entre *c_i* y *c_f* sea factible. Como la eliminación de un nodo implica que

sean eliminados todos sus arcos, resulta lógico verificar primero todos los nodos del camino y después sus arcos.

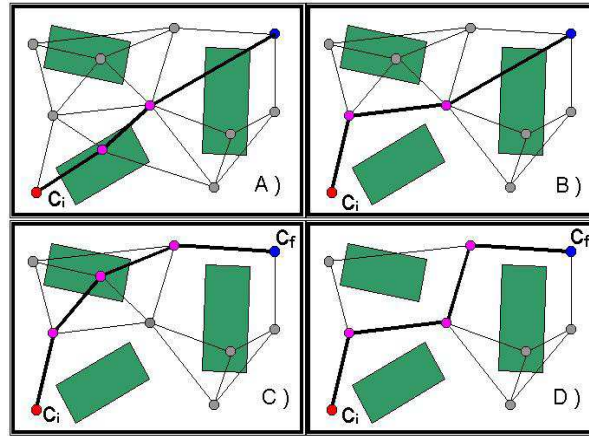


Figura 2.10: Lazy PRM revisa colisiones solo en caminos candidatos a solución
En la parte A) se detecta un nodo en colisión, el cual se elimina con todo y arcos. En la figura B) no existen nodos en colisión, pero si un camino local. En C) se observa un nuevo camino candidato a solución, pero presenta un nodo en colisión; finalmente en D) se obtiene un camino completo libre de colisiones.

2.2.3.1 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 a la ruta mas corta. La razón de verificar los nodos en este orden es que la probabilidad de encontrar el camino mas corto a través de un particular nodo es mejor si ese nodo esta cercano a C_i o C_f .

2.2.3.2 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 especificada. Como en el caso de los nodos, si una trayectoria local esta en colisión, se elimina el arco correspondiente y se busca nuevamente la ruta mas corta. Si ninguna colisión es detectada a lo largo del camino, el algoritmo termina y retorna como resultado al camino libre de colisiones.

2.2.4 Enriquecimiento del roadmap

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

Es posible que durante los pasos anteriores se llegue a un roadmap desconexo, 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 2.11.

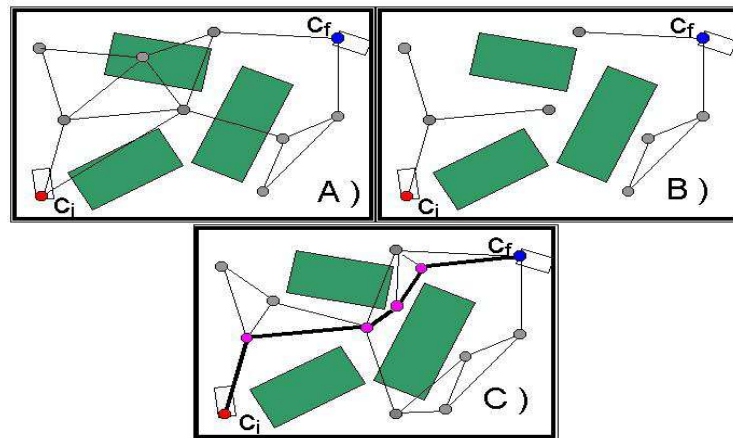


Figura 2.11: Enriquecer el roadmap

En A) se ilustra un roadmap generado, en el cual 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).

2.2.5 Prueba de colisión para un robot tipo carro

Dado un camino local, debemos probar si este esta en colisión. Esta operación no puede efectuarse de manera analítica sobre el camino. La estrategia utilizada consiste en probar la no colisión de un numero 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 [Sánchez 03]. La figura 2.12 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 configuración actual $c_c = c_i$ y se calcula a que distancia d_{min} se encuentra el obstáculo mas 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 minino de llamadas al detector de colisiones.

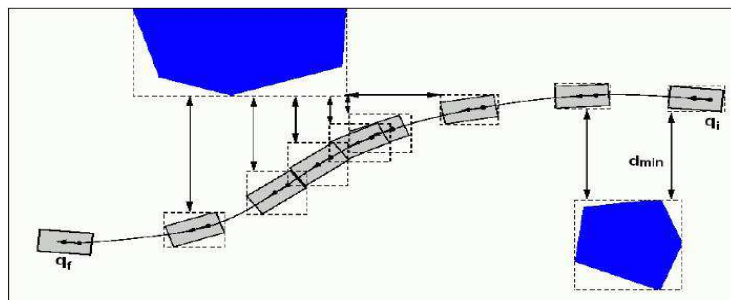


Figura 2.12: Validación de un camino libre de obstáculos

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 este en colisión con el obstáculo, o bien termina por alejarse (Figura 2.13).

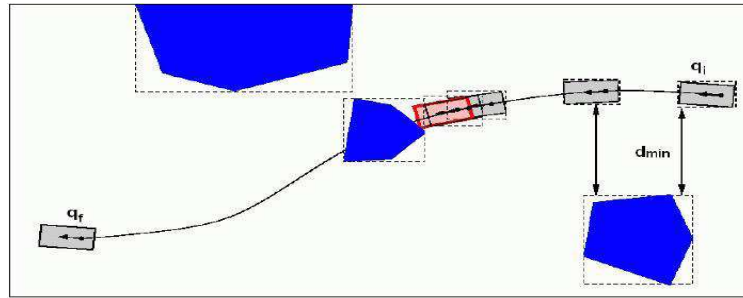


Figura 2.13: Validación de un camino en presencia de un obstáculo

2.2.6 Curvas Reeds & Shepp

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 robot (Figura 2.14); para el caso de un robot tipo carro se emplea a menudo el método Reeds & Shepp [Reeds 90].

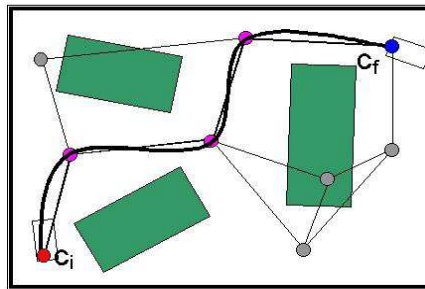


Figura 2.14: Camino factible para un robot tipo carro

Una curva Reeds & Shepp se compone de segmentos de recta y de arcos de círculo.

El problema de la caracterización de los caminos más cortos, para un robot de tipo carro fue propuesto por Reeds y Shepp [Reeds 90]. Los autores determinaron una familia de caminos óptimos. Su demostración no utiliza las técnicas de la teoría de control. El estudio de este problema para la utilización del principio del Máximo de Pontriagine fue abordado en paralelo por Boissonnat, Cerezo y Leblong, y por Sussman y Tang. Después, Soueres y Laumond propusieron una mejora, donde la familia de caminos exhibía propiedades de simetría y construyeron una síntesis de caminos más cortos que cubren el espacio de configuraciones.

Estas palabras reagrupan 48 tipos de caminos, Entre los 4 tipos resumidos por la primer palabra $C|C|C$, se ha establecido que los tipos $l- l+ l-$ y $r- r+ r-$ son equivalentes a los tipos $l+ l- l+$ y $r+ r- r+$ respectivamente. Por lo tanto, solo restan 46 tipos.

Sin embargo, si estos caminos son los mas cortos en longitud, estos exigen, para ser recorridos, que el vehiculo se detenga en cada discontinuidad de la curva, es decir en cada transición de un segmento a un arco de circunferencia (o recíprocamente). Estas paradas no son molestas cuando el camino contiene maniobras, pero no son posibles en ciertos casos (desplazamiento sobre una vía rápida, por ejemplo). Para reducir la divergencia inducida entre el camino planificado y el camino real que se recorrerá, es conveniente utilizar caminos a curvatura continua.

Un planificador local de caminos a curvatura continua toma en cuenta la totalidad de las restricciones de desplazamiento del vehiculo así como la presencia de obstáculos a evitar. Permite también obtener un camino más fácil de seguir.

Reeds & Shepp retoman los trabajos de Dubins la única diferencia es que viajar en reversa ahora esta permitido. Estos investigadores definieron un camino uniendo dos puntos u y v en el espacio de dimensión 2, tangente a los vectores de U y V , asociada a los puntos respectivamente u y v y autorizando los puntos de (transición).

Esta modificación aportada al problema inicial permite crear una curva de longitud inferior. Reeds & Shepp muestran que las curvas de longitud minimal están constituidas a lo máximo de cinco elementos con, a lo mas dos puntos de transición que denotamos por $|$. Estas curvas se resumen en nueve casos:

$$\{C|C|C, CC|C, C|CC, CSC, CC_{\beta}|C_{\beta}C, C|C_{\beta}C_{\beta}|C, \\ C|C_{\pi/2}SC, CSC_{\pi/2}|C, C|C_{\pi/2}SC|C\}$$

$\pi/2$ es dado en algunos casos indicando que la curva debe ser seguida por exactamente este numero de radianes, así mismo β indica que debe coincidir con el parámetro de otra curva.

Los caminos óptimos están constituidas por una secuencia finita de a lo mas cinco caminos elementales, los cuales pueden ser segmentos de línea recta o bien arcos de circunferencia. La figura 2.15, muestra la familia de curvas Reeds & Shepp.

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	$(L^+ S^+ L^+)(L^- S^- L^-)(R^+ S^- R^+)(R^- S^- R^-)$
$CC_\beta C_\beta C$	$(L^+ R_\beta^+ L_\beta^- R^-)(L^- R_\beta^- L_\beta^+ R^+)(R^+ L_\beta^+ R_\beta^- L^-)(R^- L_\beta^- R_\beta^+ L^+)$
$C C_\beta C_\beta C$	$(L^+ R_\beta^- L_\beta^+ R^+)(L^- R_\beta^+ L_\beta^- R^-)(R^+ L_\beta^- R_\beta^+ L^-)(R^- L_\beta^+ R_\beta^- L^+)$
$C C_{\pi/2} SC$	$(L^+ R_{\pi/2}^- S^- R^-)(L^- R_{\pi/2}^+ S^+ R^+)(R^+ L_{\pi/2}^- S^- L^-)(R^- L_{\pi/2}^+ S^+ L^+)$ $(L^+ R_{\pi/2}^- S^- L^-)(L^- R_{\pi/2}^+ S^+ L^+)(R^+ L_{\pi/2}^- S^- R^-)(R^- L_{\pi/2}^+ S^+ R^+)$
$CSC_{\pi/2} C$	$(L^+ S^+ L_{\pi/2}^+ R^-)(L^- S^- L_{\pi/2}^- R^+)(R^+ S^+ R_{\pi/2}^+ L^-)(R^- S^- R_{\pi/2}^- L^+)$ $(R^+ S^+ L_{\pi/2}^+ R^-)(R^- S^- L_{\pi/2}^- R^+)(L^+ S^+ R_{\pi/2}^+ L^-)(L^- S^- R_{\pi/2}^- L^+)$
$C C_{\pi/2} SC_{\pi/2} C$	$(L^+ R_{\pi/2}^- S^- L_{\pi/2}^- R^+)(L^- R_{\pi/2}^+ S^+ L_{\pi/2}^+ R^-)(R^+ L_{\pi/2}^- S^- R_{\pi/2}^- L^+)$ $(R^- L_{\pi/2}^+ S^+ R_{\pi/2}^+ L^-)(L^+ R_{\pi/2}^- S^- R_{\pi/2}^- R^+)(L^- R_{\pi/2}^+ S^+ R_{\pi/2}^+ R^-)$ $(R^+ L_{\pi/2}^- S^- L_{\pi/2}^- L^+)(R^- L_{\pi/2}^+ S^+ L_{\pi/2}^+ L^-)$

Figura 2.15: Familia de curvas Reeds & Shepp

Capitulo 3

Método reactivo de planificación basado en Lazy PRM y la DVZ

3.1 Zona Virtual Deformable

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) [Zapata 94], [Sánchez 01b], [Cacitti 00].

El control por ZVD esta 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 mas 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 cinemática del robot. Esta acción arrastra una modificación de la forma de la ZVD. El

ambiente, en cuanto a este, 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 vehiculo.

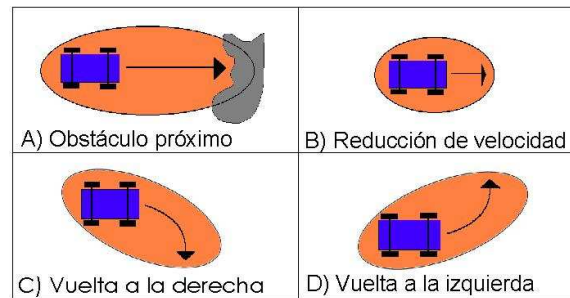


Figura 3.1: En A) la deformación de la ZVD es causada por la proximidad de un obstáculo; en B),C) y D) las deformaciones son causadas por comandos del robot

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.

3.1.1 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 esta dada por el mínimo de las dos distancias:

$$\Xi = \text{Min}(k * V^2 + d_{\text{seguridad}}, d_{\text{obstaculo}}) \quad (3.1)$$

donde:

$d_{\text{obstaculo}}$ es la distancia al primer obstáculo y

$k * V^2 + d_{\text{seguridad}}$ es la función que corresponde a la distancia de frenado necesaria para que el vehiculo se detenga.

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

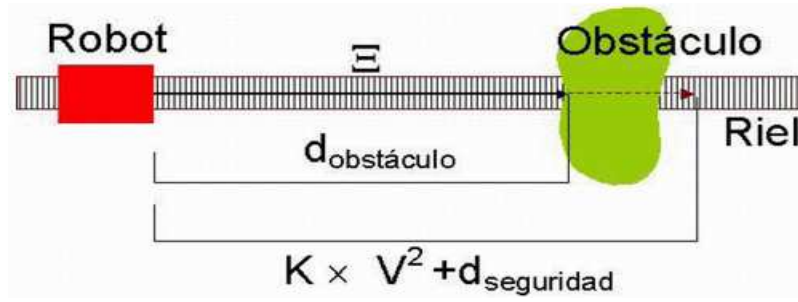


Figura 3.2: Forma de Ξ para un robot que se desplaza sobre un riel

3.1.2 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 esta representado por estos dos componentes:

$$\chi = \begin{bmatrix} \Xi \\ \sigma \end{bmatrix}, \text{ con } \Xi = \begin{bmatrix} \Xi_1 \\ \Xi_2 \\ \vdots \\ \Xi_c \end{bmatrix} \quad (3.2)$$

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 proximidad c_i .

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

$$\Phi = \pi \quad (3.3)$$

con

$$\pi = f(\chi) \quad (3.4)$$

El vector de control Φ de un robot móvil es la derivada de ciertas variables cinemáticas reagrupadas dentro del vector π (Ecuación 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 .

3.1.3 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 esta deformada por el ambiente:

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

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:

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

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

$$\forall \chi_h \in H, \rho \circ f(\chi_h) = \chi_h \in H \quad (3.7)$$

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.

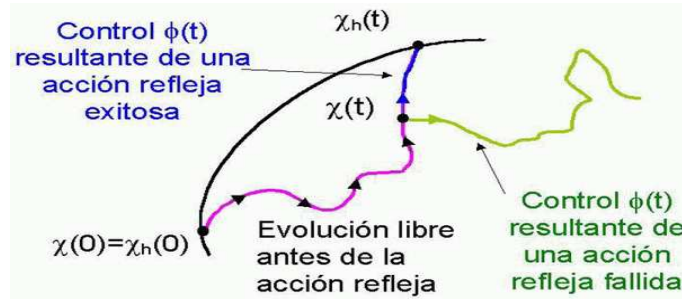


Figura 3.3: Evolución del estado intrínseco χ dentro de E

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.

$$\dot{\chi}_h = \rho'(\pi)\Phi \text{ donde } \ddot{\Xi}_h = \rho'_{\Xi}(\pi)\Phi \quad (3.8)$$

3.1.4 Intrusiones y deformaciones debidas al ambiente

Sea I el vector de intrusión (Ecuación 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:

$$I = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_i \\ \vdots \\ I_c \end{pmatrix} \quad \text{donde } I_i = d_{imax} - d_i \quad (3.10)$$

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

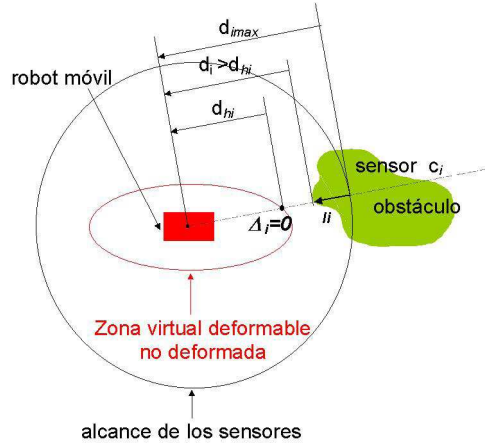


Figura 3.4: El vector de deformación $\Delta_i=0$

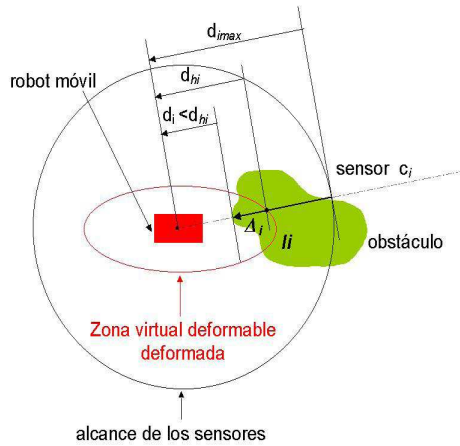


Figura 3.5: El obstáculo deforma la ZVD

Definimos el vector de deformación Δ (Ecuación 3.10), 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 .

$$\Delta = \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_i \\ \vdots \\ \Delta_c \end{pmatrix} \quad (3.10)$$

Cada elemento Δ_i esta definido gracias a la función mostrada en la ecuación 3.11 y que se ilustra con las figuras 3.4 y 3.5.

$$\Delta_i = \alpha(d_{hi}, I_i) = \begin{cases} 0 & \text{si } d_i > d \\ d_{hi} - d_i & \text{si } d_i \leq d \end{cases} \quad (3.11)$$

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

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 únicamente a Φ
- Un vector Δ que representa la deformación generada por el ambiente (Ecuación 3.12).

$$\Xi = \Xi_h - \Delta \quad (3.12)$$

3.1.5 Ecuación de estado

De la ecuación 3.11, se puede obtener la ecuación de evolución de Δ :

$$\dot{\Delta} = \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I) \dot{\Xi}_h + \frac{\partial \alpha}{\partial I}(\Xi_h, I) \dot{I} \quad (3.13)$$

Derivando 3.12 y utilizando las ecuaciones 3.13 y 3.8

$$\dot{\Xi} = \rho'_{\Xi}(\pi) \Phi - \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I) \rho'_{\Xi}(\pi) \Phi - \frac{\partial \alpha}{\partial I}(\Xi_h, I) \dot{I} \quad (3.14)$$

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

$$\dot{\chi} = \begin{cases} \dot{\Xi} = \left(Id_{c \times c} - \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I) \right) \rho'_{\Xi}(\pi) \Phi - \frac{\partial \alpha}{\partial I}(\Xi_h, I) \Psi \\ \dot{\sigma} = \rho_{\sigma}(\pi) \Phi \end{cases} \quad (3.15)$$

$$con = \begin{cases} \Phi = \pi \\ \Psi = \dot{I} \\ \dot{\Xi}_h = \rho'_{\Xi}(\pi) \Phi \end{cases} \quad (3.16)$$

3.1.6 Generación de comandos reflejos

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

$$\begin{aligned} A &= \frac{\partial \alpha}{\partial \Xi_h}(\Xi_h, I) \rho'_{\Xi}(\pi) \\ B &= \frac{\partial \alpha}{\partial I}(\Xi_h, I) \end{aligned} \quad (3.17)$$

Sustituyendo la ecuación 3.16 y la ecuación 3.17 en la ecuación 3.13, la evolución de la deformación es modelada por:

$$\dot{\Delta} = A\Phi + B\Psi \quad (3.18)$$

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; para tal efecto es necesario aplicar un vector de control optimo Φ_{opt} , el cual se obtiene al despejarlo en una ecuación con la forma 3.18, como se muestra en la ecuación 3.19.

$$\Phi_{opt} = A^+(\dot{\Delta}_{des} - B\Psi) \quad (3.19)$$

donde A^+ es la pseudo inversa 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:

$$\dot{\Delta}_{des} = K_p\Delta + K_d\dot{\Delta} \quad (3.20)$$

3.1.7 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.

3.2 Lazy PRM Reactivo

Lazy PRM reactivo integra los métodos de planificación Lazy PRM mencionado anteriormente en el capítulo 2 (sección 2.2) y de control reactivo por ZVD (sección 3.1) 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 o en un árbol RRTDual. 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 planificara un camino alternativo 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 interrumpe dichos procesos 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 alternativo para concluir su tarea. Lo anterior se puede ilustrar en la figura 3.6.

3.2.1 Planificador Lazy PRM

La descripción de alto nivel del algoritmo empleado, corresponde a la mostrada en la figura 2.9, 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.

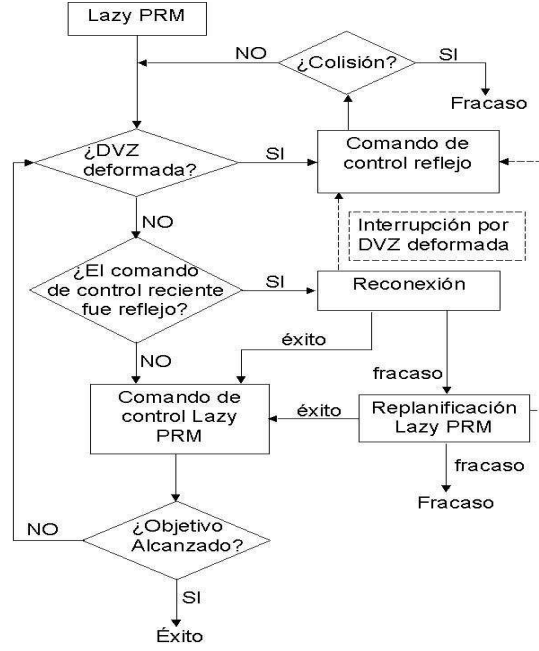


Figura 3.6: Descripción de alto nivel del algoritmo

3.2.2 Construcción del roadmap inicial.

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

$$V \leftarrow c_i, c_f$$

$$E \leftarrow (c_i, c_f)$$

mientras $n_{\text{nodos}} < N$ hacer

$$c \leftarrow \text{configuracion aleatoria}$$

$$V \leftarrow V \cup \{c\}$$

$\forall v \in V$, en orden ascendente de $d(v, c)$ hacer

$$\text{si } d(c, v) < d_{\max} \wedge \text{tamaño}(V_c) < k$$

$$E \leftarrow E \cup \{(c, v)\}$$

end

3.2.3 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 3.8.

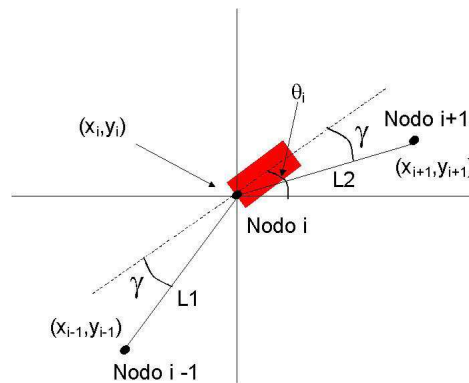


Figura 3.8: Obtención de θ para la configuración del nodo i -ésimo del camino mas corto

3.2.4 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 mas corto, si no hay nodos en colisión se llama al método local Reeds&Shepp para los arcos del camino, revisándose a cada cierta distancia en el caso de que sea un segmento de recta o cada determinados grados en el caso de que sea una curva. Si hay caminos en colisión se eliminan los arcos correspondientes y se busca nuevamente el camino mas corto, en caso contrario se discretizan las curvas Reeds&Shepp a una resolución muy fina entre configuraciones, para cada una de las cuales se llama al detector de colisiones.

Se utiliza un solo detector de colisiones que es proporcionada por la librería PQP, el cual revisa un punto en el plano (MSLVector) que cumpla de acuerdo al ángulo del carro en ese punto. En el caso de la curva Reeds & Shepp se utiliza el método ObtenerCurvaPath el cual revisa puntos dentro de la curva para saber si alguna porción de la curva esta en colisión, como se muestra en la figura 3.9.

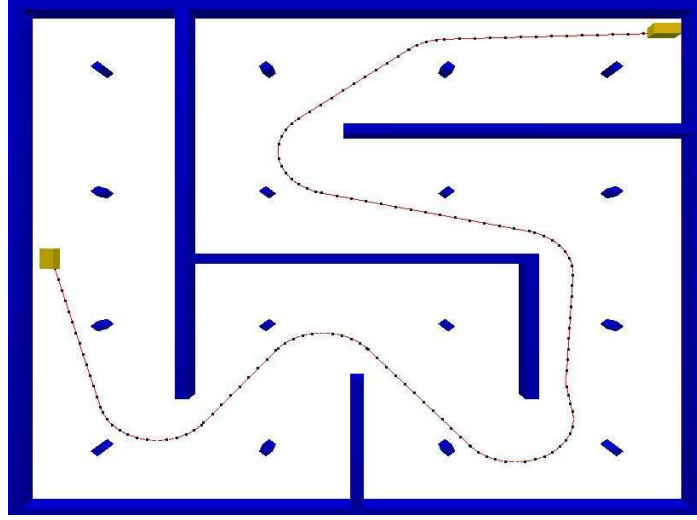


Figura 3.9: Puntos revisados por el detector de colisiones en la curva

3.3 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:

$$d_{hi} = K_1 V_1^2 \cos^2(\beta_i + K_2 \dot{\theta}) + d_i^{\text{sec}} \quad (3.21)$$

Donde K_1 y K_2 son constantes, V_1 y $\dot{\theta}$ son velocidades del robot (Ver la ecuación 3.26), β_i es el ángulo 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 c_i .

Para el primer caso de la ecuación 3.11 ($d_i > d_{hi}$), la ZVD no esta 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 θ descrito en la ecuación 3.4, y que en este caso adoptara la forma

$$\pi = \begin{bmatrix} V_1 \\ \dot{\theta} \end{bmatrix} \quad (3.22)$$

En este trabajo, en lugar de implementar el control como indica la ecuación 3.19, se utiliza lo siguiente:

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

$$f_i[n] = \begin{cases} \Delta_i[n] - \Delta_i[n-1] & \text{si } \Delta_i[n] - \Delta_i[n-1] > 0 \\ 0 & \text{si } \Delta_i[n] - \Delta_i[n-1] \leq 0 \end{cases} \quad (3.23)$$

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

$$F[n] = \sum_{i=1}^c f_i[n] \quad (3.24)$$

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

$$\pi[n] = \begin{cases} V_1[n] = V_1[n-1] + K_v * \|F[n]\| * \text{signo}(\cos(\hat{F}[n])) \\ \dot{\theta}[n] = \dot{\theta}[n-1] + K_t * \text{sen}(\hat{F}[n]) \end{cases} \quad (3.25)$$

3.4 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 3.10-B), sin embargo bajo el esquema Lazy PRM Reactivo, 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.

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 3.10-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 realizara un nuevo intento (Figura 3.10-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.

3.5 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.

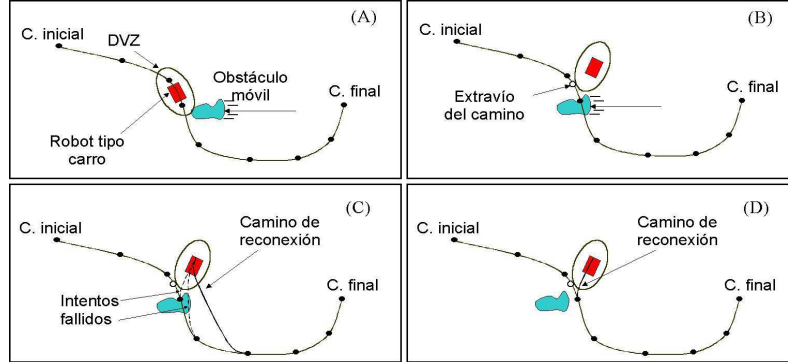


Figura 3.10: En (A) es necesaria una acción refleja para evitar el obstáculo móvil, en (B) se muestra la situación del robot después de la acción refleja, en (C) se muestra el camino de reconexión al camino planificado. En el caso de que todos los intentos de (C) hubieran estado en colisión, el proceso se repite después de algún tiempo, y es posible entonces obtener éxito, como sucede en (D).

3.6 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_l y $\dot{\theta}$, como se muestra en la ecuación 3.26 y conforman el vector π de la ecuación 3.4. Estas actitudes se indican a intervalos Δt , siendo constantes durante cada uno de ellos.

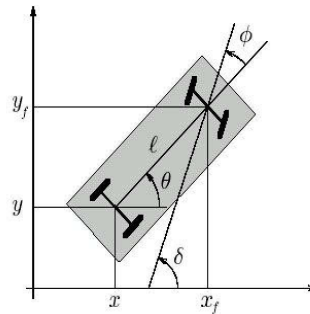


Figura 3.11: Diagrama del robot tipo carro.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ (\tan \phi)/l \\ 0 \end{bmatrix} V_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} V_2 \quad (3.26)$$

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 3.27 con ayuda de la figura 3.12, muestra como se obtiene la nueva configuración para el robot después de la aplicación de un impulso determinado.

$$\begin{aligned} x_2 &= x_1 + r(\cos \gamma_2 - \cos \gamma_1) \\ y_2 &= y_1 + r(\sin \gamma_2 - \sin \gamma_1) \\ \theta_2 &= \theta_1 + q * (V_1/r) * \Delta t \end{aligned}$$

donde: (3.27)

$$\begin{aligned} r &= \text{MAX}(\text{abs}(V_1/\dot{\theta}), R_{\min}) \\ q &= \text{signo}(V_1/\dot{\theta}) \\ \gamma_1 &= \theta_1 - q(\pi/2) \\ \gamma_2 &= \theta_2 - q(\pi/2) \end{aligned}$$

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

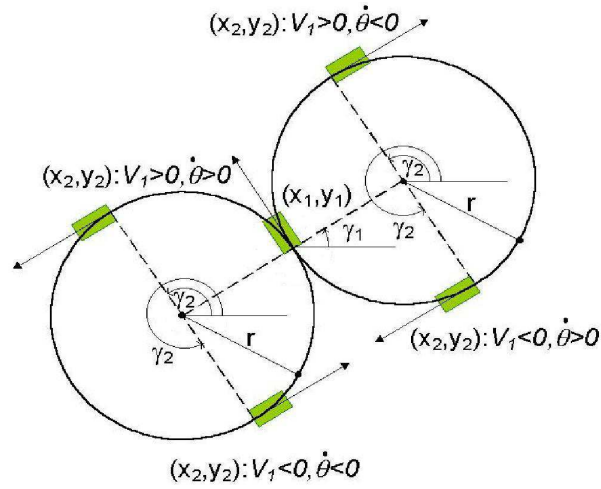


Figura 3.12: La figura muestra la relación entre las variables de la ecuación 3.27.

3.7 Distinción entre objetos 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.

Para resolver este problema se agregan los obstáculos estáticos cuando es cargado el ambiente, sin tomar hasta ese momento en cuenta ningún obstáculo móvil, así mismo cuando se tiene activada la ZVD los sensores únicamente toman en cuenta los obstáculos móviles sin importarle si un obstáculo fijo esta deformando la ZVD.

Capitulo 4

Otros métodos reactivos

En este capitulo, presentamos una extensión de los métodos de planificación por redes probabilistas con el fin de tratar eficientemente escenas dinámicas compuestas a la vez de obstáculos estáticos y móviles. El método propuesto está basado en la combinación de los métodos PRM con los métodos de difusión para explotar sus respectivas ventajas. El método igualmente integra una actualización eficiente de la red por medio de mecanismos de memorización y de evaluación perezosa (lazy). Los resultados muestran que el desempeño del enfoque permite su utilización para la deformación “tiempo-real” de trayectorias en presencia de obstáculos móviles que perturban la ejecución de movimiento planificado.

4.1 Ejemplo introductivo

La figura 4.1 presenta un ejemplo de ambiente cambiante. El robot (una plataforma móvil que transporta una tabla), evoluciona en un ambiente de interior de tipo oficina. Además, de dos puertas alternativamente abiertas o cerradas, la escena contiene tres obstáculos móviles que pueden aparecer alrededor de la tabla (ver imagen abajo a la derecha). La figura muestra igualmente para una misma consulta pero para tres contextos diferentes, tres caminos calculados en tiempo-real para nuestro planificador.

Este ejemplo introductivo ilustra las dificultades ligadas a la consideración de obstáculos móviles por los algoritmos de planificación. Se debe por un lado considerar diversos tipos de obstáculos: mientras que algunos pueden tener un número discreto de colocaciones (típicamente las puertas de un ambiente), otros pueden tener un conjunto finito de colocaciones posibles. Ciertos obstáculos también aparecen, desaparecen, o cambian de posición en el transcurso de las consultas sucesivas. Por otro lado, un obstáculo

móvil que cambia localmente de posición puede conducir a una modificación global de los caminos solución.

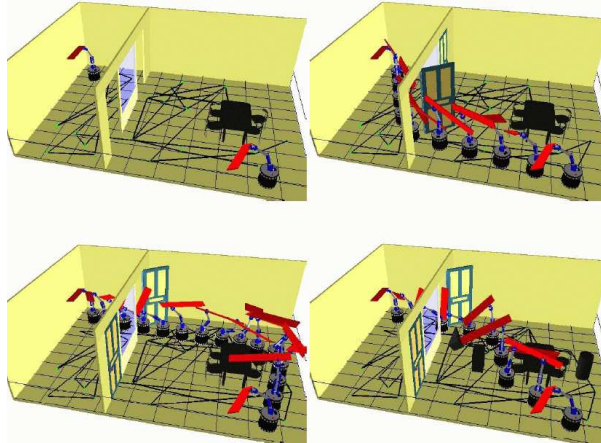


Figura 4.1: Configuraciones iniciales y finales de una plataforma manipuladora y tres caminos solución obtenidos para tres contextos de escenas diferentes.

Los enfoques probabilistas son en la actualidad los únicos métodos capaces de tratar de manera genérica los problemas para sistemas robóticos complejos. En el estado, estos son sin embargo mal adaptados a los problemas dinámicos. Por un lado los algoritmos de difusión que construyen un nuevo grafo en cada consulta, pierden cada vez toda la información ligada a los obstáculos estáticos. Por otro lado, los métodos de consultas múltiples construyen redes que pueden ser invalidadas por los obstáculos móviles. La utilización de estas redes no puede pues hacerse sin la adición de un mecanismo de actualización para tener en cuenta el contexto actual.

Proponemos un planificador híbrido, que explota las ventajas ligadas a estas dos familias de métodos: la construcción de redes probabilistas permite capturar el espacio libre sin considerar los obstáculos móviles y un método de difusión que permite enseguida reconectar las porciones invalidadas por un contexto dado. Se utiliza igualmente una estrategia eficiente que combina la actualización de la red y búsqueda de un camino solución. El objetivo de esta estrategia es reducir al máximo las pruebas de validez que representan generalmente las operaciones más costosas.

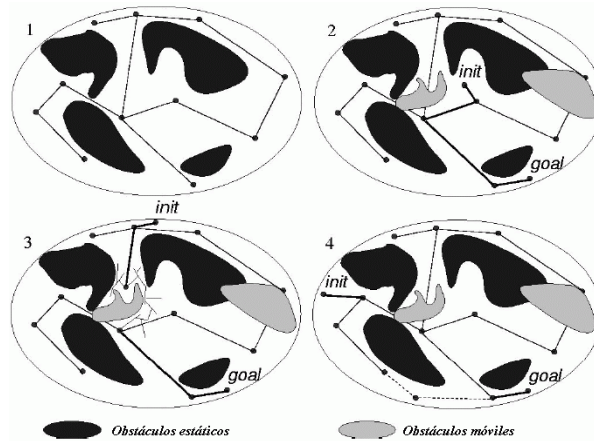


Figura 4.2: Se calcula una red valida para el ambiente estático (1). Durante las consultas, un camino solución puede encontrarse directamente en esta red (2) o con la ayuda de una técnica de conexión de los arcos “rotos” (3). Si la red existente no permite obtener una solución, se insertan nuevos nodos y se refuerza la red a través de la aparición de ciclos (4)

4.2 Principio del enfoque

A continuación, presentamos el principio general del enfoque descrito en [Jaillet 04]. Este comprende dos etapas principales. La primera es un etapa de inicialización, durante la cual se construye una red válida en el espacio de configuraciones, en ausencia de obstáculos móviles (figura 4.2, 1). Como esta etapa se efectúa una sola vez para un ambiente dado, las restricciones para el tiempo de construcción no son importantes. El objetivo de esta etapa es construir una estructura de datos que permita en lo subsiguiente tratar más eficientemente las consultas de planificación. Durante el transcurso de una consulta, los nodos extremos están conectados a la red, después las porciones son actualizadas por las pruebas de colisión limitadas a los obstáculos móviles. Los arcos invalidados entonces se etiquetan como bloqueados en la red. En ciertos casos, este etiquetado es suficiente para obtener un camino solución sin cálculo suplementario (figura 4.2, 2). Sino, un mecanismo de reconexión local trata de reparar los arcos invalidados (figura 4.2, 3). Este mecanismo permite reforzar la robustez de la red en relación a los obstáculos móviles y mejorar el desempeño. Finalmente, si la red actual no permite responder a la consulta, una fase de reforzamiento permite introducir nuevos nodos y arcos.

Estos nuevos elementos pueden conducir a la aparición de ciclos que igualmente robustecen a la red (figura 4.2, 4).

La eficiencia del enfoque se basa igualmente en otros enfoques que permiten evitar lo más posible las pruebas de colisión inútiles. En particular el planificador incluye los elementos siguientes:

- **Un método de evaluación perezosa:** permite seleccionar las conexiones de los nodos de consulta más prometedores y limitar la actualización de la red a ciertas porciones pertinentes.
- **Un proceso de memorización:** las pruebas de colisiones realizadas en el transcurso de las consultas sucesivas son memorizadas para nuevamente ser explotadas cuando un contexto ya encontrado se presente.

La arquitectura global del planificador se presenta en la figura 4.3.

4.2.1 Red inicial

La red calculada en la inicialización debe ser únicamente válida considerando los obstáculos estáticos. Enseguida, en una consulta de planificación, el cálculo de un camino solución necesita la actualización de ciertas porciones de la red precalculada. Se asocia a cada uno de los nodos y arcos de la red tres estados o etiquetas posibles, en función del contexto: validado, invalidado o no probado.

No importa que método de construcción de redes probabilistas pueda ser a priori utilizado para la construcción. Sin embargo, para que el planificador se desempeñe bien en las fases de consulta, estas redes deben verificar dos propiedades a priori antagonistas. Por un lado la red debe ser de baja densidad para que su actualización no sea penalizada. Por otro lado debe ser lo suficientemente rica para contener caminos alternativos cuando algunos arcos son invalidados por los obstáculos móviles.

La elección del método mejor adaptado se expone posteriormente.

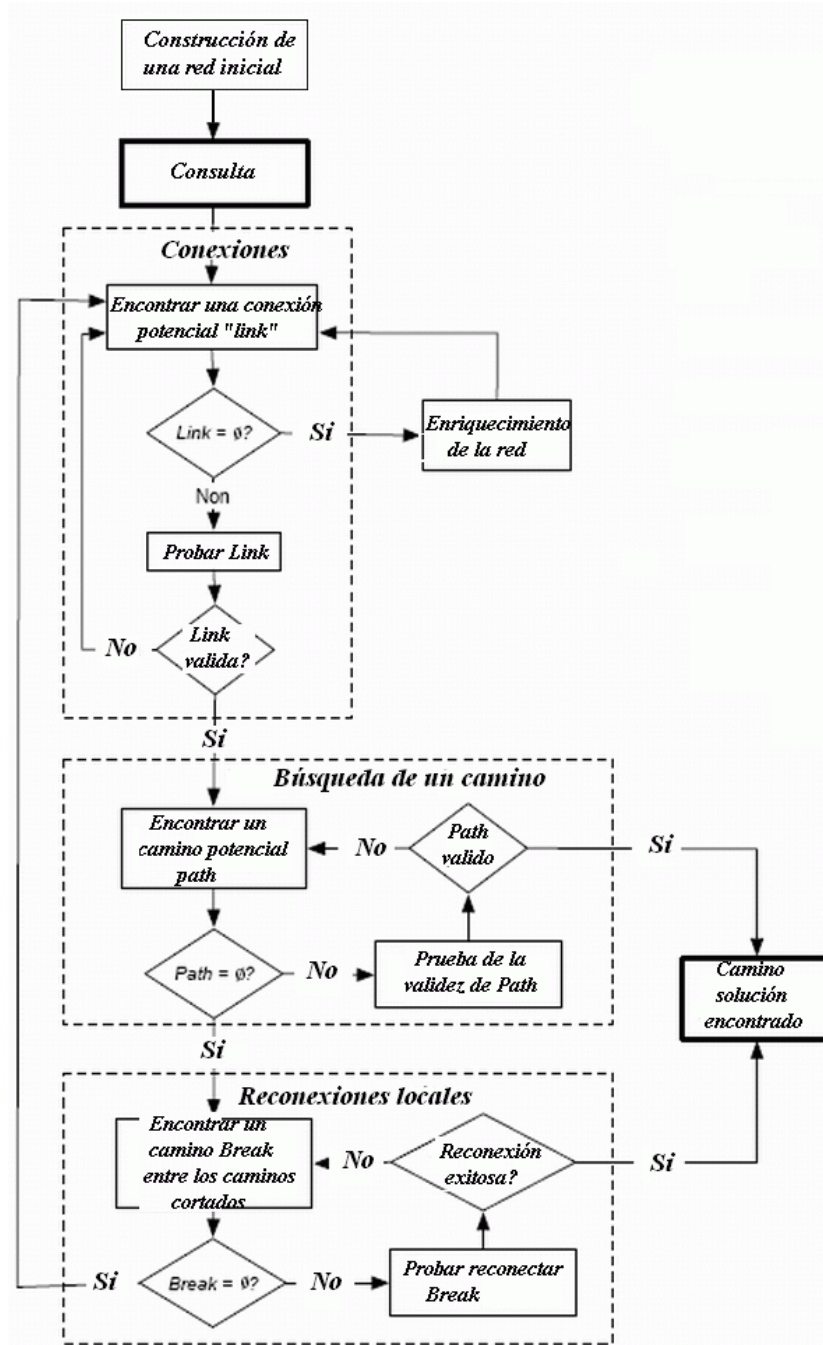


Figura 4.3. Arquitectura global del planificador para ambientes cambiantes.

4.2.2 Consulta de planificación

La resolución de una consulta de planificación es generalmente efectuada en tres etapas sucesivas [Lean 96]:

- Actualización del conjunto de nodos y arcos de la red inicial y cálculo de las componentes conexas asociadas al contexto actual.
- Conexiones de los nodos de consulta a las nuevas componentes conexas formadas.
- Búsqueda de un camino solución en la red actualizada.

En la práctica, este método “ingenuo” tiene un desempeño pobre. Es decir, la prueba sistemática de los arcos de la red es en la mayor parte de los casos inútil, en particular cuando la red contiene un camino validado con respecto al ambiente estático y que responde a la consulta de planificación (figura 4.4, 1). En la sección siguiente, se propone un método más interesante que prueba en prioridad los arcos asociados a los caminos más prometedores. La conexión sistemática de los nodos de consulta a todas las componentes se debe igualmente evitar (figura 4.4, 2 y 4.4, 3).

Estas operaciones son por otra parte más penalizantes, las pruebas de validez consideran también a los obstáculos estáticos. El método propuesto permite por el contrario limitar al máximo estas conexiones y seleccionar en prioridad aquellas que tienen más posibilidades de ser válidas.

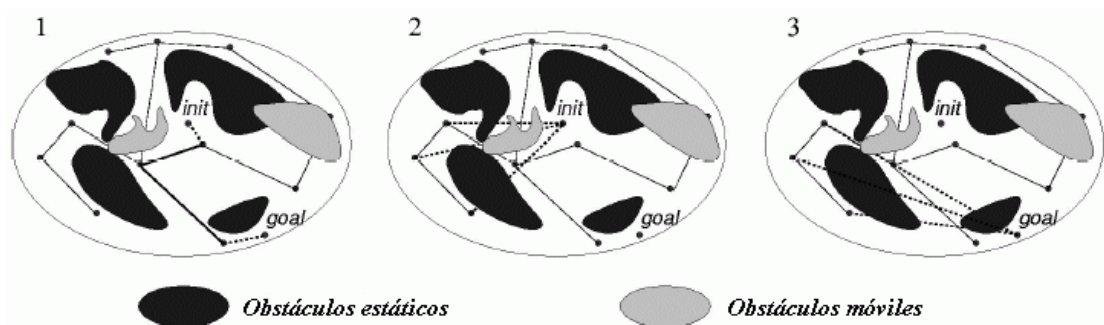


Figura 4.4. Un camino solución puede frecuentemente extraerse de la red sin etiquetar la totalidad de los arcos (1). De la misma manera, la conexión de los nodos de consulta al conjunto de las componentes es con frecuencia inútil (2 y 3).

Para esto, nuestro método se basa en una estrategia de evaluación perezosa que consiste en una actualización gradual de la red paralelamente a la búsqueda de un camino solución. Es sobre esta implementación en paralelo del proceso de búsqueda y del proceso de actualización que reposa el desempeño del algoritmo. Además, como se mencionó precedentemente, la robustez de un camino de la red se mejora, gracias a una operación de reconexión que repara ciertos arcos invalidados de la red. Así, tres operaciones correspondientes a los bloques de la figura 4.3 se efectúan de manera iterativa hasta que un camino válido sea encontrado:

- La conexión de los nodos de consulta a los nodos de la red (bloque conexión).
- La búsqueda de un camino solución en la red (bloque búsqueda de un camino).
- La reparación de los caminos invalidados con la ayuda de un método de reconexión (bloque reconexiones locales).

Estas tres operaciones así como el método de etiquetado perezoso de los arcos se detallan posteriormente.

4.3 Actualización de la red en las consultas de planificación

4.3.1 Conexiones a la red

El objetivo de esta etapa (bloque conexiones figura 4.3) es seleccionar en función de la red parcialmente actualizada, los nodos candidatos con los cuales se pueden conectar los nodos de consulta. Aquí, se supone que ciertos arcos han sido ya etiquetados como validados o invalidados, mientras que otros no se han probado. En este contexto, se dice

que un nodo es potencialmente alcanzable a partir de otro nodo si existe un camino que los une y que no contiene arcos validados o arcos que no han sido aun probados. Por el contrario, un nodo es no alcanzable a partir de otro si todos los caminos que los unen contienen al menos un arco etiquetado como bloqueado. Esta noción permite reagrupar los diferentes nodos de la red en tres subconjuntos:

- K_{init} , nodos potencialmente alcanzables a partir del nodo inicial.
- K_{goal} , nodos potencialmente alcanzables a partir del nodo objetivo.
- K_{na} , nodos que, para la actualización actual, son no alcanzables a partir de los nodos de consulta.

La utilización de estos tres subconjuntos conduce a distinguir tres tipos de conexiones de los nodos de consulta entre los cuales será escogida la más prometedora (conexión). El algoritmo asociado a este mecanismo de selección se presenta en la figura 4.5.

Los tres subconjuntos de nodos (K_{init} , K_{goal} , K_{na}) forman parte de los datos de entrada del algoritmo. Al inicio de una consulta, estos conjuntos son inicializados considerando que todos los nodos pertenecen a K_{na} . Enseguida estos conjuntos son actualizados considerando las conexiones de nodos de consultas o de las actualizaciones de los caminos de la red (función Update K de los algoritmos Connect y ExtractPath). La función BestNode, permite determinar los nodos más próximos de estos conjuntos para los cuales las conexiones no se han intentado probar. Se vislumbra entonces tres tipos de conexiones:

- L_{init} , conexión de nodo inicial a los nodos potencialmente alcanzables a partir del nodo final.
- L_{goal} , conexión del nodo final a los nodos potencialmente alcanzables a partir del nodo inicial-
- L_{na} , conexión simultánea de los dos nodos de consulta al tercer subconjunto, reagrupando los nodos no alcanzables a partir de la actualización actual.

Entre estos tres tipos de conexiones potenciales, aquella asociada a la más corta distancia se selecciona finalmente (línea 10 figura 4.5).

Con este método, la selección de la conexión interviene por lo tanto en dos niveles. La función *BestNode*, permite determinar para cada conjunto K_{init} , K_{goal} , K_{na} , las conexiones las más cortas no aún probadas. El algoritmo selecciona enseguida la conexión a probar como la más corta de los tres grupos. El método permite así al final escoger entre todas las conexiones presentando un interés para el cálculo de un camino solución aquel más corto no aún probado que tiene más posibilidades de estar validado.

```

Connect( $G, K_{init}, K_{goal}, K_{na}$ )
1  LinkValid  $\leftarrow$  False
2  While LinkValid = False
3       $n_{init} \leftarrow \text{BestNode}(K_{goal}, init)$ 
4       $L_{init} \leftarrow n_{init} - init$ 
5       $n_{goal} \leftarrow \text{BestNode}(K_{init}, goal)$ 
6       $L_{goal} \leftarrow n_{goal} - goal$ 
7       $n_{init}^{na} \leftarrow \text{BestNode}(K_{na}, init)$ 
8       $n_{goal}^{na} \leftarrow \text{BestNode}(K_{na}, goal)$ 
9       $L_{na} \leftarrow (n_{init}^{na} - init) \cup (n_{goal}^{na} - goal)$ 
10     Link  $\leftarrow \text{BestLink}(L_{init}, L_{goal}, L_{na})$ 
11     If Link =  $\emptyset$ 
12         Return  $\emptyset$ 
13     End If
14     LinkValid  $\leftarrow \text{TestLink}(\text{Link})$ 
15 End While
16 UpdateK( $G$ )
17 Return Link

```

Figura 4.5: El algoritmo Connect, de búsqueda de los mejores tipos de conexión de los nodos de consulta a la red

El escenario de la figura 4.6 ilustra este mecanismo de conexión. Al inicio de la consulta, ningún arco se ha actualizado y ningún nodo se ha considerado como alcanzable (es decir, todos pertenecen a K_{na}). Así, la primer conexión se hace uniendo los dos nodos de consulta a este conjunto (figura 4.6, 1). Enseguida, se busca un camino validado y una porción de la red se actualiza (figura 4.6, 2). En la etapa siguiente, los tres tipos de conexiones L_{init} , L_{goal} y L_{na} , representadas respectivamente en las figuras 4.6, 3 a, b y c, son posibles.

En el ejemplo, la conexión L_{na} (figura 4.6, 3) permite conectar el conjunto K_{na} a los nodos extremos y así encontrar un camino solución. Es por otro lado interesante notar que son estos los arcos bloqueados que conforman el conjunto L_{na} , reagrupando los nodos no alcanzables, mientras que por el contrario, la conexión de los nodos de consulta tiende a considerar los dos subconjuntos K_{init} y K_{goal} .

Una vez que una conexión validada se ha determinado, un camino solución se puede buscar en la red.

4.3.2 Búsqueda de un camino

Una vez que los nodos de consulta están conectados a la red, se busca extraer de esta red un camino valido (bloque Búsqueda de un camino figura 4.3). El algoritmo ExtractPath asociado a esta etapa de búsqueda se representa en la figura 4.7.

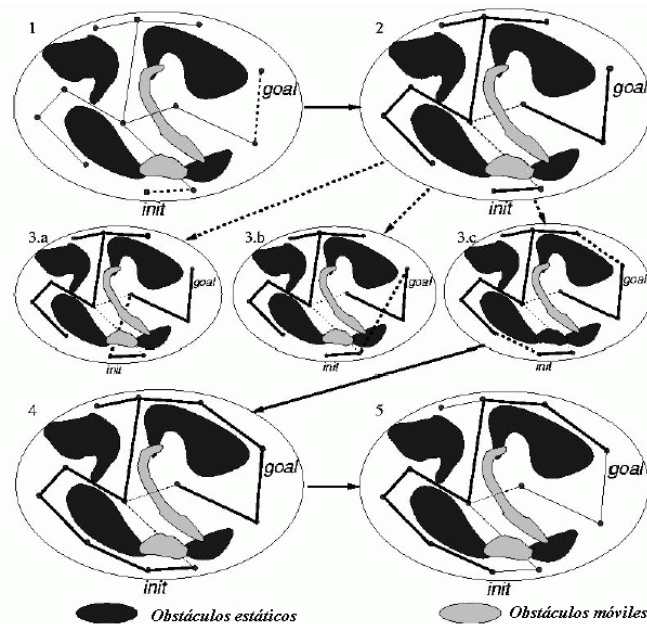


Figura 4.6. Los nodos de consulta están conectados a la red (1) y se realiza una primera tentativa de búsqueda (2). Si esta falla, tres conjuntos de nodos aparecen y tres tipos de

conexiones asociadas son posibles (3a, 3b, 3c). En el caso considerado, el último tipo de conexión (3c) permite encontrar un camino valido (4, 5)

```
ExtractPath( $G, Link$ )
1  BrokenPath  $\leftarrow$  EmptyList
2  Do
3    Path  $\leftarrow$  BestPath( $G, Link$ )
4    PathLabeling  $\leftarrow$  Valid
5    For All Edge in Path
6      If EdgeLabeling(Edge)=Invalid
7        PathLabeling  $\leftarrow$  Invalid
8        AddToList(BrokenPath,Path)
9      End If
10   End For All
11   If PathLabeling = Valid
12     Return Path
13   End If
14   While Path  $\neq \emptyset$ 
15     UpdateK( $G$ )
16   Return BrokenPath
```

Figura 4.7: Algoritmo ExtractPath, buscando un camino valido en la red

La función BestPath se basa en el algoritmo A* para determinar entre los caminos no aún probados y uniendo los nodos de consulta, aquel correspondiente a la distancia más corta. Enseguida, su validez considerando los obstáculos móviles se prueba por el método de etiquetado de los arcos descrito en la sección 4.3.5. Si el camino es valido, la consulta se resuelve sin cálculo suplementario. Sino, el camino se agrega a la lista de caminos potenciales para los cuales se debe utilizar un proceso de reconexión (sección 4.3.3). Este proceso de búsqueda se realiza hasta que todos los caminos potenciales se han probado o que se haya encontrado una solución.

Un ejemplo que ilustra las diferentes etapas de búsqueda/actualización se representa en la figura 4.8. Después de la conexión de los nodos de consulta (1), los diferentes caminos potenciales son probados y los arcos asociados aún no probados se actualizan (2 a 9). Como cada uno de los caminos no conduce a una solución, el algoritmo intenta entonces reconectar localmente las porciones de caminos invalidados (10). Para el ejemplo presentado, es esta última etapa que permite al final obtener una solución (11-12).

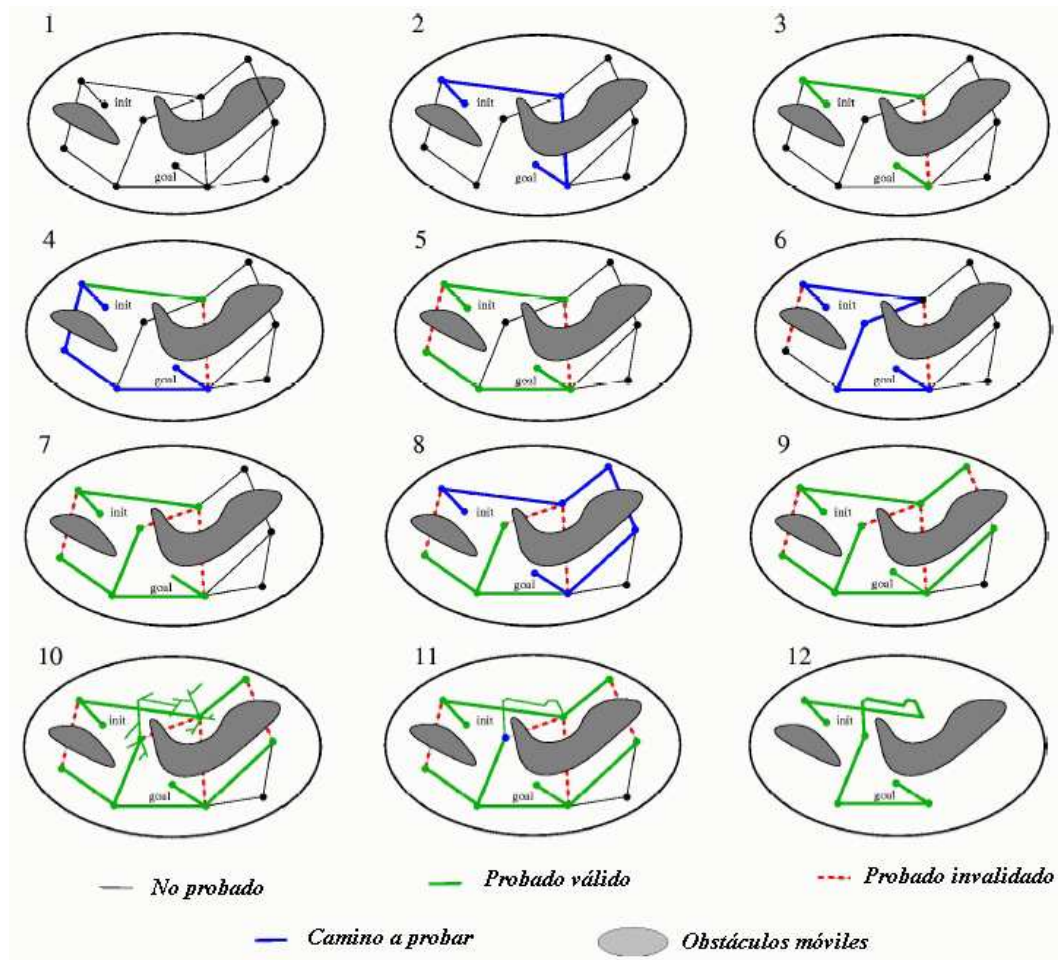


Figura 4.8. Ejemplo de búsqueda de un camino solución. Una primer conexión de los nodos de consulta se efectúa (1) y todos los caminos potenciales se prueban (2 a 9). Como cada uno de ellos no permite obtener un solución, un método de reconexión de caminos invalidados se efectúa (10) permitiendo al final la resolución del problema (11-12).

4.3.3 Reconexiones locales

Después de cada nueva conexión de los nodos de consulta, el algoritmo intenta encontrar un camino solución que no contenga ningún arco bloqueado. Cuando no existe un camino valido, el algoritmo intenta reconstruir localmente un camino valido reparando ciertos arcos invalidados por los obstáculos móviles (bloque Reconexiones locales figura 4.3).

La estrategia privilegiada, reconectará los caminos invalidados solamente después de haber probado el conjunto de los caminos potenciales y con esto explotar al máximo la red precalculada: ya que su actualización no necesita de pruebas que para el caso de los obstáculos móviles, las operaciones de reconexión necesitan igualmente tomar en cuenta los obstáculos estáticos. Los caminos que contienen las porciones inválidas las más cortas son probados primero, ya que estos son los caminos más susceptibles de ser reconectados.

La reconexión de los arcos se efectúa por un método de difusión de tipo RRT. Una versión más interesante de este algoritmo (RRT para ambientes dinámicos) se presentará posteriormente.

En último recurso, si este mecanismo de reconexión falla para la integralidad de los caminos que se buscan reparar, una nueva conexión de los nodos de consulta se prueba siguiendo el método descrito en la sección 4.3.1.

4.3.4 Enriquecimiento (reforzamiento) de la red

El método presentado, realiza dos operaciones en paralelo: la actualización de la red y la búsqueda de un camino solución. Si al final, el proceso de búsqueda conduce a un etiquetado total de la red sin llegar a una solución, esto significa que ciertos arcos invalidados por los obstáculos móviles conducen a una red no conexa en el contexto actual. La red es entonces reforzada insertando nodos y arcos que reconstruyen la conexidad de la red en el contexto actual. Este mecanismo de reforzamiento conduce a la inserción de nuevos ciclos al nivel de la estructura de datos de la red. El principio utilizado para reconstruir las componentes no conexas en el contexto actual es el mismo que aquel utilizado en la visibilidad: un nodo no es agregado si permite conectar dos componentes actualmente invalidadas o si este no es visible en el contexto actual. Esta técnica permite limitar la inserción de los nodos a aquellos realmente pertinentes, lo que conduce a un reforzamiento de la red gracias a la adición de nuevos ciclos conservando una estructura de datos de tamaño pequeño.

Notamos igualmente que esta fase de reforzamiento corresponde a la fase más costosa del proceso de resolución de consulta. Por esta razón, no se utiliza más que como último recurso, cuando todas las operaciones de actualización de la red han fracasado.

4.3.5 Etiquetado de los arcos

Conjuntamente a la estrategia perezosa de búsqueda/actualización, el planificador utiliza la información de las consultas pasadas para mejorar la eficiencia del etiquetado de los arcos. La memorización de las pruebas es posible gracias a una estructura de datos específica asociada a cada arco (figura 4.9). Esta permite de guardar las pruebas de validez del arco respecto a los obstáculos móviles situados en posiciones dadas.

El etiquetado de los arcos se efectúa en el transcurso de las pruebas sucesivas de los caminos de la red (línea 6 del algoritmo de la figura 4.7). Esta operación se detalla con ayuda del algoritmo de la figura 4.10. La primera vez que un arco es etiquetado, ninguna información no ha sido almacenada y las pruebas de validez se efectúan normalmente. Paralelamente, los resultados de estas pruebas son almacenados en la estructura de datos (figura 4.9), tomando como posiciones claves las posiciones actuales de cada uno de los obstáculos móviles. La próxima vez que el arco deba ser actualizado, antes de probar su validez con respecto al obstáculo móvil, la posición actual de este obstáculo se compara a las posiciones ya almacenadas. Si una de ellas corresponde a la posición actual, el resultado de la prueba de colisiones anteriormente realizado se extrae de la estructura, evitando así una nueva prueba inútil. Sino, la nueva posición y la prueba de validez asociada son memorizadas. La ventaja de esta estructura es que las pruebas son memorizadas para los obstáculos móviles considerados independientemente. Así, en cuanto uno de ellos reanuda uno de sus anteriores colocaciones, las pruebas de actualización pueden ser reducidas, de la misma forma si el contexto en su globalidad es nuevo.

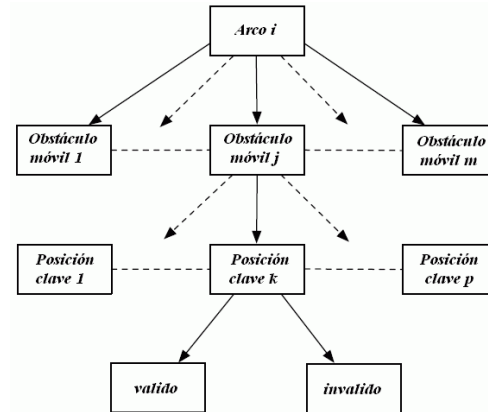


Figura 4.9: Almacenamiento de la información de colisión en los arcos.

En la práctica, es conveniente sin embargo por razones de memoria, limitar el mecanismo de memorización a las últimas posiciones tomadas por los obstáculos móviles. Este mecanismo es en realidad muy eficaz para evitar muchas pruebas inútiles para los dos tipos de situaciones siguientes:

- cuando un obstáculo posea un conjunto discreto de posiciones (es decir, una puerta sea cerrada o sea abierta). Las pruebas de colisión no se realizan más que una sola vez por posición discreta.
- cuando un obstáculo se detiene temporalmente (es decir, un objeto desplazado o un robot se detiene).

El método de memorización permite entonces tratar el obstáculo, como si este formara parte de la escena estática.

Al final, para un arco dado las pruebas verdaderamente efectuadas son aquello, difícilmente evitables, asociadas a nuevas posiciones de obstáculos móviles.

```
EDGE LABELING(E)
1  For each Movable Obstacle  $OM_i$ 
2    For each Key Position  $KP_j$  of  $OM_i$  stored in  $E$ 
3      If  $KP_j = \text{CurrentPos}$ 
4         $IsCollision \leftarrow \text{ExtractColInfo}(E, KP_j)$ 
5      Else
6         $IsCollision \leftarrow \text{TestCol}(E, OM_i)$ 
7        Store  $IsCollision$  in  $R$ 
8      End If
9      If  $IsCollision = \text{True}$ 
10        Return invalid
11      End If
12    End For
13  End If
14  Return valid
```

Figura 4.10: Algoritmo de etiquetado de arcos

4.4 RRT Reactivo

Este método consiste en la conjunción del método RRT descrito anteriormente (sección 2.1) con la ZVD mencionada en el capítulo 3 (sección 3.1), la idea surge por la comparación hecha entre los árboles de exploración rápida y los métodos probabilistas para casos con obstáculos estáticos; ahora lo que se pretende es ampliar esta comparación para el caso de obstáculos en constante movimiento y su efectividad en su comportamiento. En esta tesis ya se hablo del método Lazy PRM reactivo que es el que corresponde por el lado de los métodos probabilistas, RRT reactivo no es más que la solución propuesta por lado de los métodos basados en árboles aleatorios de exploración rápida.

RRT es extenso ya que existen diversas variantes que van desde RRT's simples hasta RRT's bidireccionales, en este caso se utiliza para la planificación, la versión RRTExtExt que es bidireccional, 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 la zona de protección, no se requieren comandos reflejos y el control corre por cuenta de RRT. En el 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, inmediatamente después 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 (al igual que en el método Lazy PRM, esto es factible) o en un árbol RRTDual. Si el intento de reconexión tiene éxito, el robot ejecuta el nuevo camino hacia su objetivo, sino se planifica nuevamente por medio de un árbol ExtExt. Durante los procesos de reconexión y de replanificación el robot permanece fijo, pero en caso de presentarse una deformación se interrumpe dichos procesos por acciones reflejas.

4.4.1 Revisión de colisiones

Los cálculos de intrusión y de los impulsos son calculados de la misma forma que en Lazy PRM Reactivo véase sección 3.3 y 3.4. Así mismo, se utiliza la misma cinemática del robot y la consideración de obstáculos estáticos y móviles se efectúa de la misma forma. Lo único que difiere es el detector de colisiones ya que aquí solo es llamado en los nodos del árbol generado y en el otro método es llamado cuando se busca el camino y cuando se calculan las curvas para dicho camino.

En cuanto al camino de reconexión generado, su comportamiento varía según el tipo de reconexión que se escoja, en el caso de que sea una curva simple se llama al detector de colisiones por cada punto resultante de la discretización a lo largo de la curva, por otra parte si la reconexión se efectúa por RRTDual, el detector es llamado más veces ya que se llama por cada nodo perteneciente a los dos árboles generados para buscar el camino.

4.4.2 Replanificación

Si después de varios intentos, la reconexión no se puede efectuar ya sea por la presencia de obstáculos estáticos o móviles se calcula nuevamente un árbol de tipo ExtExt pero esta vez la posición inicial es la última posición del carro, se actualiza el camino y todos los criterios de la ZVD empiezan nuevamente.

Capitulo 5

Resultados

A continuación se presentan los resultados obtenidos al comparar las diferentes estrategias que se implementaron en la herramienta desarrollada (RRT-BUAP), la cual fue implementada en Visual Studio C++ 6 y ejecutada en un computador personal basada en un procesador Pentium 4 a 2.6 GHz con 1 GB 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

El planificador fue probado en los ambientes mostrados en las figuras 5.1, 5.2 y 5.3, bajo diferentes condiciones: cantidad inicial de nodos N que conforman el roadmap de 25 y 50, distancia máxima d_{max} para poder insertar un arco entre dos nodos de 50 y 80, y finalmente diferentes restricciones cinemáticas, que corresponden a ángulos de conducción máximos de 25 y 45 grados, hay que recordar que el ángulo de conducción da la intensidad de las curvas Reeds&Shepp.

Para el ambiente uno que es el correspondiente a la figura 5.1 las condiciones iniciales son:

- Configuración Inicial: (5.0, 50.0)
- Configuración Final: (95.0, 50.0)
- Angulo Inicial: 270°
- Angulo Final: 270°

Las otras condiciones variaron según se presentan en la tabla 5.1 para los cuales se hicieron 5 corridas y se presenta el promedio de ellas.

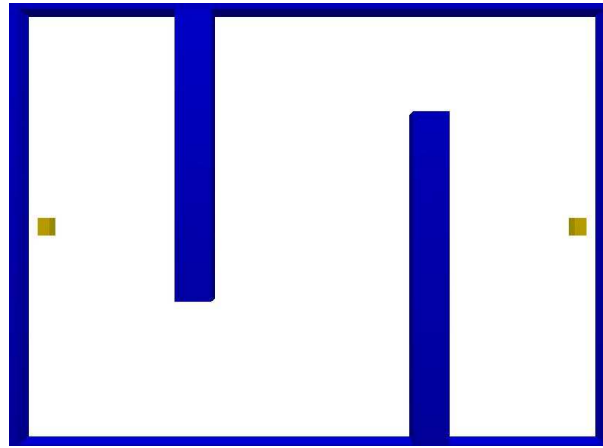


Figura 5.1: Ambiente 1

Nodos	$d_{\max}$	Angulo de dirección	Llamadas al detector de colisiones	Tiempo de construcción	Tiempo de planificación	Tiempo total	Vértices abiertos por A*	Éxitos
25	50	25	1742	0.0096004	0.1093982	0.1189986	16	5/5
25	50	45	1742	0.0094	0.0878	0.0972	16	5/5
25	80	25	4949	0.0124	0.2406	0.253	17	5/5
25	80	45	4949	0.0124	0.2218	0.2342	17	5/5
50	50	25	14079	0.0254	0.7342	0.7596	32	5/5
50	50	45	7049	0.028	0.3938	0.4218	32	5/5
50	80	25	58643	0.0346	2.7376	2.7722	35	5/5
50	80	45	19097	0.0376002	0.8939998	0.9316	33	5/5

Tabla 5.1: Resultados del ambiente 1 con Lazy PRM

Para el ambiente seis que es el correspondiente a la figura 5.2 las condiciones iniciales son:

- Configuración Inicial: (5.0, 94.0)
- Configuración Final: (6.0, 10.0)

- Angulo Inicial: 0°
- Angulo Final: 180°

Los resultados se presentan en la tabla 5.2.

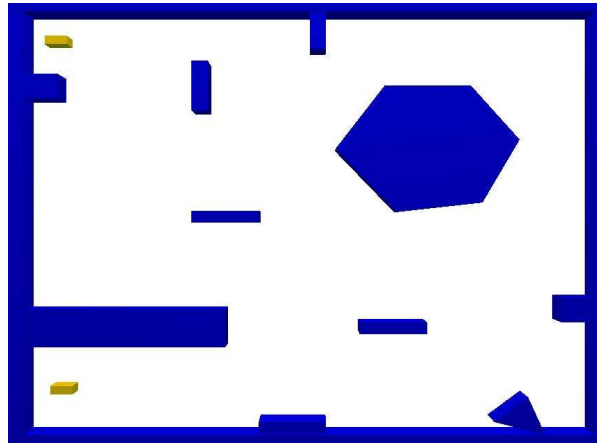


Figura 5.2: Ambiente 6

Nodos	$d_{\max}$	Angulo de dirección	Llamadas al detector de colisiones	Tiempo de construcción	Tiempo de planificación	Tiempo total	Vértices abiertos por A*	Éxitos
25	50	25	1337	0.0409998	0.0876002	0.1286	14	5/5
25	50	45	1337	0.0126	0.069	0.0816	14	5/5
25	80	25	7798	0.0158	0.3718	0.3876	14	5/5
25	80	45	7798	0.0128	0.3688	0.3816	14	5/5
50	50	25	5726	0.025	0.3027998	0.3277998	27	5/5
50	50	45	5726	0.0216	0.3186	0.3402	27	5/5
50	80	25	14554	0.0374	0.7437998	0.7811998	26	5/5
50	80	45	14554	0.0378	0.7468	0.7846	26	5/5

Tabla 5.2: Resultados del ambiente 6 con Lazy PRM

Para el ambiente nueve que es el correspondiente a la figura 5.3 las condiciones iniciales son:

- Configuración Inicial: $(5.0, 95.0)$

- Configuración Final: (90.0, 5.0)
- Angulo Inicial: 315°
- Angulo Final: 135°

Los resultados se presentan en la tabla 5.3.



Figura 5.3: Ambiente 9

Nodos	$d_{\max}$	Angulo de dirección	Llamadas al detector de colisiones	Tiempo de construcción	Tiempo de planificación	Tiempo total	Vértices abiertos por A*	Éxitos
25	50	25	2679	0.0094	0.1002	0.1096	20	5/5
25	50	45	2679	0.0094	0.1094	0.1188	20	5/5
25	80	25	5556	0.0094	0.2	0.2094	12	5/5
25	80	45	5556	0.0124	0.2062	0.2186	12	5/5
50	50	25	10640	0.028	0.4968	0.5248	38	5/5
50	50	45	10640	0.0282	0.5156	0.5438	38	5/5
50	80	25	22185	0.0376	1.019	1.0566	25	5/5
50	80	45	22185	0.0374	0.9312	0.9686	25	5/5

Tabla 5.3: Resultados del ambiente 9 con Lazy PRM

5.2 Evaluación del planificador RRT

Para poder llevar acabo una correcta comparación entre los dos métodos, las pruebas de RRT se hicieron en los mismos ambientes con las mismas condiciones, tanto iniciales como finales; con la diferencia que en las tablas no se presentan los promedios de los resultados lanzados por las cinco corridas si no que se presenta una por una y después el promedio.

Esto se hace así debido al comportamiento variado de este planificador, esto se puede comprobar viendo los datos obtenidos mostrados en las siguientes tablas.

Nodos	Llamadas al detector de colisiones	Tiempo de planificación	Nodos creados	Éxitos
1000	22920	8.29699	1253	No
	20796	16.157	1268	No
	18048	12.21899	1100	Si
	18288	26.937993	1169	Si
	18036	6.375	1160	Si
	19617.6	13.9971946	1190	3/5
2000	30228	13.952999	1712	Si
	29580	11.937	1554	Si
	39180	19.406	2090	Si
	19956	7.671	1301	Si
	35520	18.56199	2065	Si
	30892.8	14.3057978	1744.4	5/5

Tabla 5.4: Resultados del ambiente 1 con RRT

Nodos	Llamadas al detector de colisiones	Tiempo de planeación	Nodos creados	Éxitos
1000	11268	2.812	792	Si
	10692	2.422	674	Si
	19368	7.687	1302	No
	10656	3.062	765	Si
	11784	3.109	768	Si
	12753.6	3.8184	860.2	4/5
2000	25092	9.375	1484	Si
	14280	4.156	901	Si
	9468	2.313	641	Si
	13320	3.719	832	Si
	13896	4.625	996	Si
	15211.2	4.8376	970.8	5/5

Tabla 5.5: Resultados del ambiente 6 con RRT

Nodos	Llamadas al detector de colisiones	Tiempo de planeación	Nodos creados	Éxitos
1000	6888	1.672	571	Si
	6708	1.532	558	Si
	12252	3.843	1009	Si
	10488	3.016	873	Si
	9084	2.391	747	Si
	9084	2.4908	751.6	5/5
2000	11700	3.5	957	Si
	11388	3.437	945	Si
	5112	1.047	424	Si
	11496	3.453	950	Si
	4008	0.75	331	Si
	8740.8	2.4374	721.4	5/5

Tabla 5.6: Resultados del ambiente 9 con RRT

5.3 Evaluación de Lazy PRM Reactivo

El dinamismo de los ambientes consiste en la adición de obstáculos móviles colocados aleatoriamente los cuales se desplazan en línea recta a una velocidad constante, las pruebas se realizan en 2 ambientes mostrados en la figura 5.4 y 5.5 primero con 5 obstáculos y después con 10; los siguientes valores se utilizaron para las acciones reflejas:

- Sensores: 20
- K1: 1.0
- K2: 10.0
- Radio: 5.0
- KV: 0.5
- KT: 0.3

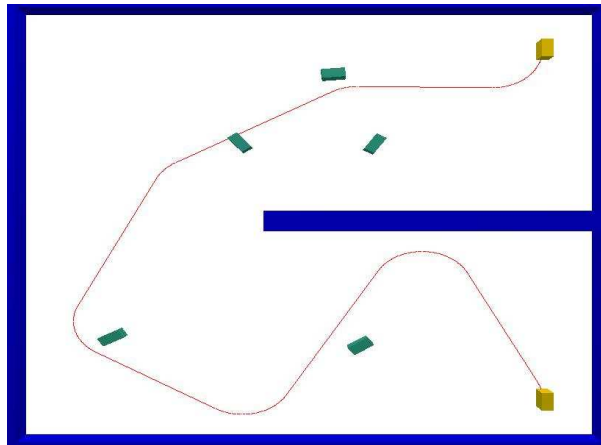


Figura 5.4: Ambiente 2 con obstáculos, planificación con Lazy PRM

La reconexión se realizó por medio de una curva simple, los resultados son mostrados en la tabla 5.7 y 5.8.

Reconexiones	Tiempo total	Tiempo promedio	Replani- ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
25	0.328999	0.01315996	0	0	0			*
12	0.422001	0.03516675	0	0	0			*
40	0.674995	0.016874875	0	0	0			*
9	0.376001	0.041777889	0	0	0			*
62	1.280968	0.020660774	0	0	0			*

Tabla 5.7: Resultados del ambiente 2 con 5 obstáculos bajo Lazy PRM Reactivo

Reconexiones	Tiempo total	Tiempo promedio	Replani- ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
62	1.108994	0.017887	0	0	0			*
-	-	-	-	-	-		*	
3	0.031	0.010333333	0	0	0			*
36	0.897022	0.024917278	0	0	0			*
8	0.295	0.036875	0	0	0			*

Tabla 5.7: Resultados del ambiente 2 con 10 obstáculos bajo Lazy PRM Reactivo

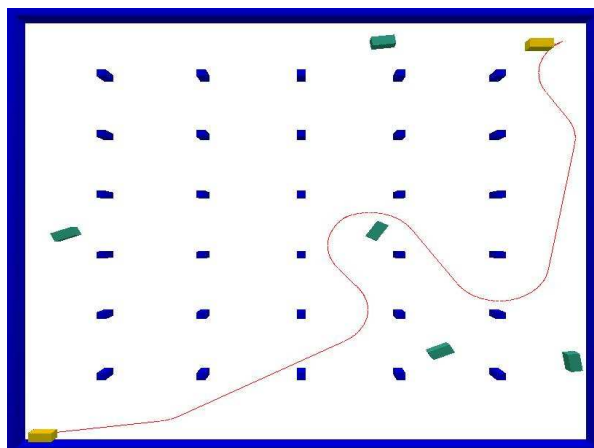


Figura 5.5: Ambiente 8 con obstáculos, planificación con Lazy PRM

Reconexiones	Tiempo total	Tiempo promedio	Replani- ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
16	0.248997	0.015562313	0	0	0			*
0	0	0	0	0	0			*
9	0.203997	0.022666333	0	0	0			*
8	0.137999	0.017249875	0	0	0			*
9	0.2989998	0.033222	0	0	0			*

Tabla 5.8: Resultados del ambiente 8 con 5 obstáculos bajo Lazy PRM Reactivo

Reconexiones	Tiempo total	Tiempo promedio	Replani- ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
78	2.194015	0.028128397	0	0	0			*
26	0.432993	0.016653577	0	0	0			*
-	-	-	-	-	-		*	
60	0.78008	0.01300133	1	0.172001	0.172001			*
4	0.108998	0.0272495	0	0	0			*

Tabla 5.9: Resultados del ambiente 8 con 10 obstáculos bajo Lazy PRM Reactivo

5.4 Evaluación de RRT Reactivo

En esta sección se presenta los resultados de prueba del método RRT reactivo, el cual fue probado en los mismos ambientes que el Lazy PRM reactivo con el propósito de obtener el mayor número de condiciones similares, así que los datos de las acciones reflejas son los mismos que en la sección anterior; los resultados son los mostrados de la tabla 5.10 a la 5.13.

Table 5.10: Results of the experiment 3 – 5 about the choice: PPTP or notTable 5.11: Resultados del ambiente 2 con 10 obstáculos bajo DDT Reactive

Reconexiones	Tiempo total	Tiempo promedio	Replani- ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
1	0.031002	0.031002	0	0	0			*
22	0.497978	0.022635364	0	0	0			*
31	1.452009	0.046839	0	0	0			*
20	0.872992	0.0436496	1	3.594002	3.594002			*
1	0.015999	0.015999	1	0.907	0.907			*

Tabla 5.12: Resultados del ambiente 8 con 5 obstáculos bajo RRT Reactivo

Reconexiones	Tiempo total	Tiempo promedio	Replani- ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
113	3.395971	0.030052841	0	0	0			*
14	0.467997	0.033428357	0	0	0			*
5	0.218998	0.0437996	0	0	0			*
16	0.267006	0.016687875	0	0	0			*
-	-	-	-	-	-		*	

Tabla 5.13: Resultados del ambiente 8 con 10 obstáculos bajo RRT Reactivo

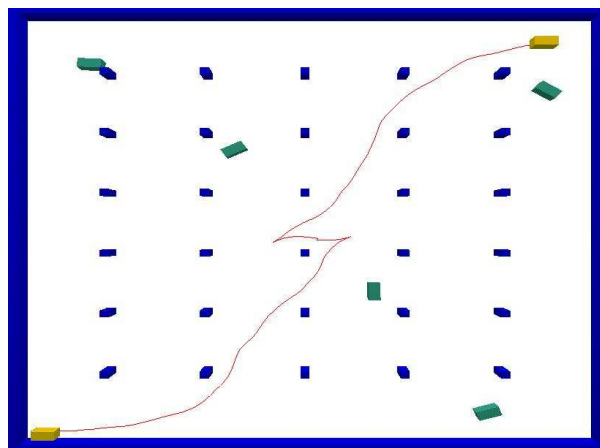


Figura 5.7: Ambiente 8 con obstáculos, planificación con RRT

5.4.2 Reconexión RRTDual

El método RRT Reactivo cuenta con dos formas diferentes de reconectarse una vez que el camino original es perdido por el resultado de acciones reflejas, una es por medio de una curva Reeds & Shepp simple cuyos resultados fueron mostrados en las tablas de la 5.10 a la 5.13, y la otra forma por analizar en este trabajo de tesis es por medio de un árbol RRTDual; los resultados obtenidos llevando acabo la reconexión mediante esta forma se muestran en las tablas de la 5.14 a la 5.17, la razón por la que son cuatro es porque como en los casos anteriores se analizo su comportamiento en dos ambientes y cada uno con diferentes números de obstáculos.

Reconexiones	Tiempo total	Tiempo promedio	Replani-ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
2	0.766003	0.3830015	0	0	0			*
10	14.480991	1.4480991	0	0	0			*
7	33.62599	4.803712857	0	0	0			*
0	0	0	0	0	0			*
4	10.794998	2.6987495	0	0	0			*

Tabla 5.14: Resultados del ambiente 2 con 5 obstáculos bajo RRT Reactivo

Reconexiones	Tiempo total	Tiempo promedio	Replani-ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
-	-	-	-	-	-		*	
2	2.453001	1.2265005	0	0	0			*
30	73.340977	2.44699233	0	0	0			*
4	2.265001	0.56625025	0	0	0			*
7	16.391991	2.341713	0	0	0			*

Tabla 5.15: Resultados del ambiente 2 con 10 obstáculos bajo RRT Reactivo

Reconexiones	Tiempo total	Tiempo promedio	Replani-ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
7	17.533001	2.504714429	0	0	0			*
4	7.624996	1.906249	0	0	0			*
13	26.639011	2.049154692	0	0	0			*
2	1.624996	0.812498	0	0	0			*
20	65.780996	3.2890498	0	0	0			*

Tabla 5.16: Resultados del ambiente 8 con 5 obstáculos bajo RRT Reactivo

Reconexiones	Tiempo total	Tiempo promedio	Replani-ficaciones	Tiempo total	Tiempo promedio	Camino NO encontrado	Colisión	Éxito
13	32.435993	2.495076385	0	0	0			*
-	-	-	-	-	-		*	
-	-	-	-	-	-		*	
16	36.845005	2.302812813	0	0	0			*
18	74.483994	4.137999667	0	0	0			*

Tabla 5.17: Resultados del ambiente 8 con 10 obstáculos bajo RRT Reactivo

Capitulo 6

Conclusiones y trabajo futuro

En cuanto a la planificación de movimientos se refiere, después de que se ha analizado el comportamiento de dos planificadores (Lazy PRM y RRT) se observa que los dos tienen ventajas y desventajas. Por un lado Lazy PRM es muy rápido ya que evita muchas llamadas al detector de colisiones y puede encontrar el camino incluso con muy pocos nodos, mas sin embargo si el ambiente cuenta con demasiados obstáculos estáticos el roadmap necesitara mas nodos, por lo que el tiempo para la búsqueda de un camino libre de colisiones aumenta e incluso puede que no lo encuentre (y se hagan llamadas a la fase de enriquecimiento del grafo, lo que implicará más tiempo de procesamiento), en las pruebas realizadas notamos una fuerte dependencia en la distancia permitida entre cada nodo y el numero de nodos.

En este tipo de ambientes el comportamiento de RRT es igualmente efectivo, a pesar de que en general la planificación es menos rápida con respecto a Lazy PRM, en ambientes con muchos obstáculos este planificador es más eficiente al encontrar un camino libre de colisión. En resumen si el ambiente cuenta con muchos obstáculos y pasajes estrechos es mejor la planificación con los métodos RRT, en caso contrario Lazy PRM es más eficiente ya que su tiempo de construcción y planificación es inferior a los métodos RRT.

La parte reactiva estudiada en este trabajo muestra que es una buena opción para solucionar el problema de la robótica móvil en ambientes dinámicos, no es una solución completa para todos los casos, ya que la planificación de movimientos en ambientes dinámicos es un problema muy complejo.

A pesar de que Lazy PRM Reactivo y RRT reactivo son dos métodos diferentes, su parte reactiva la resuelven de igual manera, es decir por medio de una curva simple para el caso de la reconexión, esta estrategia es efectiva y muy rápida ya que tan solo toma unas pocas milésimas de segundo para volver a reconectar el camino perdido debido a la

presencia de obstáculos con el camino antes planificado, en esta parte no hay gran diferencia; el asunto es elegir correctamente el planificador que se adapte a un ambiente dado. En lo que se refiere a la parte dinámica el comportamiento reactivo es similar.

La parte que difiere en ambas estrategias reactivas, es cuando en el RRT reactivo se utiliza una reconexión mediante un árbol RRT; aquí el tiempo de reconexión es importante, por otra lado, la reconexión es muy efectiva e incluso puede llevar a evitar mas obstáculos que están en movimiento, ya que el camino de la reconexión es mas amplio y puede tomar diferentes direcciones, a diferencia de una curva simple que es mas probable que pueda volver a colisionar con el mismo obstáculo, con el que inicialmente colisionó. Muy probablemente en un computador con mejor desempeño o en algún tipo de escenario donde el tiempo no sea primordial esta forma de reconexión sea una solución mas interesante, por ejemplo en una aplicación en tiempo real y con un robot real.

Otro método reactivo analizado en esta tesis fue el propuesto en la sección 4.1, este método utiliza el planificador Lazy-PRM para llevar acabo su planificación, mas sin embargo su parte reactiva la resuelve sin necesidad de utilizar las acciones reflejas provistas por el método de la ZVD, se pudo determinar que esta modalidad es muy rápida, la única problemática que se encontró es que si la presencia de obstáculos móviles es fuerte, entonces el tiempo de reacción va incrementándose.

En realidad este es el gran problema a resolver, ya que cualquier método reactivo utilizado en la robótica móvil, se va deteriorándose su efectividad conforme la presencia de obstáculos móviles se hace mayor, ahí radica la complejidad que existe en esta área de la robótica móvil.

Como trabajo futuro, se podría extender el trabajo de robots móviles no-holonómicos a otro tipo de sistema mecánico (por ejemplo, un brazo manipulador o una plataforma móvil provista de un brazo). En la práctica, los métodos PRM cuentan con una investigación importante en el desarrollo de métricas y métodos locales para distintos tipos de sistemas

mecánicos. Lo que no se ha realizado para los métodos RRT, cuyo principal talón de Aquiles es justamente la métrica utilizada para cierto sistema mecánico.

También sería interesante probar las estrategias en robots reales.

Bibliografía

- [Atramentov 02] A. Atramentov & S.M. LaValle. *Efficient Nearest Neighbor Searching for Motion Planning*. Proc. IEEE Int. Conf. on Robotics and Automation, pages 632–637, 2002.
- [Bohlin 00] R. Bohlin & L.E. Kavraki. *Path Planning using Lazy PRM*. Proc. IEEE Int. Conf. on Robotics and Automation, pages 521–528, 2000.
- [Borenstein 91] J. Borenstein & Y. Koren. *The vector field histogram - fast obstacle avoidance for mobile robots*. IEEE Transactions on Robotics and Automation, pages 278–288, 1991.
- [Brock 99] O. Brock & Oussama Khatib. *High-Speed Navigation Using the Global Dynamic Window Approach*. Proceedings of the International Conference on Robotics and Automation, pages 341–346, 1999.
- [Brock 00] O. Brock & O. Khatib. *Real time replanning in high-dimensional configuration spaces using sets of homotopic paths*. International Conference on Robotics and Automation, pages 550–555, 2000.
- [Brock 01] O. Brock & L. Kavraki. *Decomposition-based motion planning : A framework for real-time motion planning in high-dimensional configuration spaces*. In Proc. ICRA, volume 2, pages 1469–1474, 2001.
- [Bruce 02] J. Bruce & M. Veloso. *Real-time randomized path planning for robot navigation*. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 2383–2388, 2002.
- [Cacitti 00] Cacitti, A.: “*Reactive behaviour for mobile manipulators: application of the deformable virtual zone method*”, Technical Report LIRMM, 2000.
- [Erdmann 87] M. Erdmann & T. Lozano-Pérez. *On multiple moving objects*. Algorithmica, vol. 2(4), pages 477–521, 1987.
- [Fox 97] D. Fox, W. Burgard & S. Thrun. *The dynamic window approach to collision avoidance*. IEEE Robotics and Automation magazine, vol. 4(1), pages 23–33, 1997.
- [Fujimua 95] K. Fujimua. *Time-minimum routes in time-dependent networks*. IEEE Transactions on Robotics and Automation, vol. 11(3), pages 343–351, 1995.

- [Hsu 02] D. Hsu, R. Kindel, J.-C. Latombe & S. Rock. *Randomized Kinodynamic Motion Planning with Moving Obstacles*. International Journal of Robotics Research, vol. 21(3), pages 233–255, 2002.
- [Jaillet 04] L. Jaillet & T. Siméon. *A PRM-based Motion Planner for Dynamically Changing Environments*. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 1606–1611, 2004.
- [Jaillet 05] L. Jaillet, A. Yershova, S. M. LaValle & T. Siméon. *Adaptive Tuning of the Sampling Domain for Dynamic-Domain RRTs*. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005.
- [Kant 86] M.H. Kant & P.A Whitelock. *Toward efficient trajectory planning : The path-velocity decomposition*. The International Journal of Robotics Research, vol. 5(3), pages 72–89, 1986.
- [Kavraski 96] L. E. Kavraki, P. Svetska, J-C. Latombe, and M.H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. IEEE Transactions on Robotics and Automation, Vol. 12, No. 4, pp. 566–580, 1996.
- [Khatib 86] O. Khatib. *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*. International Journal of Robotics Research, vol. 5(1), pages 90–98, 1986.
- [Khatib 97] O. Khatib, H. Jaouni, R. Chatila & J.P. Laumond. *Dynamic path modification for car-like non-holonomic mobile robots*. Proc. IEEE International Conference on Robotics and Automation, 1997.
- [Ko 98] N.Y. Ko & R. Simmons. *The lane-curvature method for local obstacle avoidance*. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 1998.
- [Kuffner 00] J.J. Kuffner & S.M. LaValle. *RRT-Connect : An Efficient Approach to Single-Query Path Planning*. Proc. IEEE Int. Conf. on Robotics and Automation, pages 995–1001, 2000.
- [Lamiriaux 04] F. Lamiriaux, D. Bonnafous & O. Lefebvre. *Reactive Path Deformation for Non-holonomic Mobile Robots*. IEEE Transactions on Robotics, vol. 20, no. 6, pages 967–977, December 2004.
- [Large 03] F. Large. *Navigación Autónoma d'un robot mobile en environnement dynamique et incertain*. PhD thesis, Université de Savoie, 2003.

- [Large 04] F. Large, A.D. Vasquez, T. Fraichard & C. Laugier. *Avoiding Cars and Pedestrians using V-Obstacles and Motion Prediction*. In Proc. of the IEEE Intelligent Vehicle Symp., Pisa (IT), June 2004. Poster session.
- [Latombe 91] Latombe, J. C.: *“Robot motion planning”*. Kluwer Academic Publications, 1991.
- [LaValle 98] S. M. LaValle. *Rapidly-exploring random trees: A new tool for path planning*. Department of Computer Science, 1998.
- [LaValle 00] S. M. LaValle and J. J. Kuffner. *RRT-connect: An efficient approach to single-query path planning*. Proceedings of the IEEE, International Conference on Robotics & Automation, pp. 995–1001, 2000.
- [LaValle 01a] S. M. Lavalley and Z. S. Peng Cheng. *RRT-based trajectory design for autonomous automobiles and spacecraft*. Archives of control sciences, Vol.11, No. 3, pp. 51–78, 2001.
- [Lavalley 01b] S. M. Lavalley and J. J. Kuffner. *Rapidly-exploring random trees: Progress and prospects*. Algorithmic and Computational Robotics: New Directions, pp. 293–308, 2001.
- [LaValle 01c] S. M. LaValle and J. J. Kuffner. *Randomized kinodynamic planning*. Algorithmic and Computational Robotics: New Directions, Vol. 20, No. 5, pp. 378–400, 2001.
- [Lean 96] A. Mc Lean & I. Mazon. *Incremental roadmaps and global path planning in evolving industrial environments*. Proc. IEEE Int. Conf. on Robotics and Automation, vol. 2, pages 917–924, 1996.
- [Leven 00] P. Leven & S. Hutchinson. *Toward Real-Time Path Planning in Changing Environments*. Proc. Workshop on the Algorithmic Foundations of Robotics, 2000.
- [Leven 02] P. Leven & S. Hutchinson. *A framework for real-time path planning In changing environments*. The International Journal of Robotics Research, vol. 21, pages 999–1030, 2002.
- [Lozano-Pérez 83] T. Lozano-Pérez. *Spatial Planning : A Configuration Space Approach*. IEEE Transactions on Computers, vol. 32(2), pages 108–120, 1983.
- [Mazer 98] E. Mazer, J. M Ahuactzin, and P. Bessiere. *The ariadne’s clew algorithm*. Journal of Artificial Intelligence Research, No. 9, pp. 295–316, 1998.
- [Minguez 00] J. Minguez & L. Montano. *Nearness diagram navigation*. IEEE/RSJ Int.

- Conf. on Intelligent Robots and Systems, 2000.
- [Minguez 02] J. Minguez, L. Montano, T. Siméon & R. Alami. *Global nearness diagram navigation*. IEEE Int. Conf. on Robotics and Automation, 2002.
- [Overmars 95] M. H. Overmars and P. Svetska. *A probabilistic learning approach to motion planning*. Algorithmic Foundations of robotics, pp. 19–37, 1995.
- [Petti 05] S. Petti & T. Fraichard. *Partial Motion Planning Framework for Reactive Planning Within Dynamic Environments*. In Proc. of the IFAC/AAAI Int. Conf. On Informatics in Control, Automation and Robotics, Barcelona (SP), September 2005.
- [Quinlan 93] S. Quinlan & O. Khatib. *Elastic Bands : connecting path planning and control*. Proc. IEEE International Conference on Robotics and Automation, pages 802–807, 1993.
- [Reeds 90] Reeds, J. A., Shepp, R. A.: “*Optimal paths for a car that goes both forward and backwards*”. Pacific Journal of Mathematics. 145(2) (1990) 367-393.
- [Reif 85] J.H Reif & M. Sharir. *Motion planning in the presence of moving obstacles*. IEEE Symp. on Foundat. of Comp. Sci., pages 144–154, 1985.
- [Sánchez 01] Sánchez, A., Zapata R.: “*A lazy probabilistic roadmap planner for carlike robots*”, Rapport LIRMM 2001.
- [Sánchez 01b] Sánchez, A.: “*Zonas virtuales deformables(DVZ)*”, Technical Report LIRMM,2001.
- [Sánchez 03] Sánchez, A.: “*Contribution á la planification de mouvements en robotique: Approches probabilistas et approches déterministes*”. PhD Thesis, Université Montpellier II, 2003.
- [Sekhavat 98] S. Sekhavat, P. Svestka, J. P. Laumond, and M. H. Overmars. *Multilevel path planning for nonholonomic robots using semiholonomic subsystems*. Int. J. Robot. Res., No. 17, pp. 840–857, 1998.
- [Siméon 00] T. Siméon, J.-P. Laumond & C. Nissoux. *Visibility-Based Probabilistic Roadmaps for Motion Planning*. Advanced Robotics Journal, vol. 14(6), pages 477–494, 2000.
- [Simmons 96] R. Simmons. *The curvature-velocity method for local obstacle avoidance*. IEEE Int. Conf. on Robotics and Automation, 1996.
- [Svestka 95] P. Svestka, J. P. Laumond, and M. H. Overmars. *Coordinated motion*

- planning for multiple car-like robots using probabilistic roadmaps*. In Proc. of IEEE International Conference on Robotics and Automation, 1995.
- [Ulrich 98] I. Ulrich & J. Borenstein. *VFH+ : Reliable obstacle avoidance for fase mobile robots*. IEEE Int. Conf. on Robotics and Automation, 1998.
- [Ulrich 00] I. Ulrich & J. Borenstein. *VFH* : Local obstacle avoidance with lookahead verification*. IEEE Int. Conf. on Robotics and Automation, 2000.
- [Vallejo 00] D. Vallejo, C. Jones, and N. Amato. *An adaptive framework “for single shot” motion planning*. Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 1722–1727, 2000.
- [van den Berg 04] J.P. van den Berg & M.H. Overmars. *Roadmap-based motion planning in dynamic environments*. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 1598–1605, 2004.
- [van den Berg 05] J.P. van den Berg & M.H. Overmars. *Prioritized Motion Planning for Multiple Robots*. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005.
- [Yershova 05] A. Yershova, L. Jaillet, T. Siméon & S. M. LaValle. *Dynamic-Domain RRTs : Efficient Exploration by Controlling the Sampling Domain*. IEEE Int. Conf. on Robotics and Automation, 2005.
- [Yu 00] Y. Yu. *An information theoretical incremental approach to sensor-based motion planning for eye-in-hand systems*. PhD thesis, Simon Fraser University, 2000.
- [Zapata 94] Zapata, R., Lépinay, P., Thompson, P.: “*Reactive Behaviors of fast mobile robots*”, Journal of Robotic Systems 11(1), 13-20 (1994).