CAPÍTULO 2

METODOS PROBABILÍSTICOS

La planeación de movimientos tiene aplicaciones en muchas áreas tales como la robótica, sistemas de realidad virtual y diseño asistido por computadora. Aunque muchos métodos de planeación de movimientos se han propuesto la mayoría no son usados en la práctica dado que no son factibles computacionalmente excepto para algunos casos como por ejemplo cuando el robot tiene muy pocos grados de libertad. A pesar de la tecnología existente la enorme complejidad del problema solo puede ser resuelta a través del uso de métodos heurísticos o probabilísticas opuestos a los métodos completos los cuales garantizan encontrar una solución o determinar que ésta no existe [7].

A pesar del éxito obtenido por algunas heurísticas en ciertos problemas de planeación de movimientos, el encontrar rutas que conduzcan al robot a librar los pasajes estrechos formados entre los obstáculos sigue siendo una tarea difícil de alcanzar. Este capítulo describe de manera detallada los métodos probabilísticos implementados en ésta tesis (*método probabilístico de carreteras, método probabilístico de carreteras basado en obstáculos* y *método probabilístico de carreteras basado en visibilidad*) así como la forma particular como cada uno de ellos intenta solucionar la presencia de pasajes estrechos en el espacio de trabajo.

Todos los métodos probabilísticos hacen uso extenso de un método local llamado *Planeador Local* y es importante su comprensión antes de iniciar la descripción de tales métodos.

2.1 PLANEADOR LOCAL

Un método local consiste en un algoritmo que calcula una ruta entre dos configuraciones dadas q y q', denotado como $L(q, q')$, esta ruta conecta a ambas configuraciones en la ausencia de obstáculos; tales métodos son conocidos como *Planeadores Locales* y pueden

simplemente construir rutas en línea recta cuando los movimientos del robot no están restringidos, formar curvas como en el método *Reeds&Shepp* para el caso de robots móviles no holonómicos o trazar rutas *Manhattan* para sistemas mecánicos que requieren mover un grado de libertad a la vez.

Los planeadores locales son métodos determinísticos, simples y rápidos empleados para forman las conexiones entre los nodos de un roadmap y para conectar las configuraciones inicial y final durante las consultas. El planeador local produce la misma ruta cada vez que es llamado con las mismas configuraciones de entrada. Si el planeador local no es determinístico las rutas locales deben ser almacenadas en el roadmap con la desventaja de que el roadmap requerirá más espacio.

Un planeador poderoso resulta frecuentemente exitoso al buscar una ruta si ésta existe. Respecto a su desempeño, existe claramente una relación entre el tiempo gastado en cada llamada individual y el número de veces que se invoca.

Si pocos nodos son requeridos para construir un roadmap que capture la conectividad del espacio libre, suficientemente bien para responder a consultas confiablemente, no importa que el planeador local sea lento, ya que esto será compensado dado el número reducido de veces que es llamado. Por otro lado un planeador muy rápido será poco exitoso si el número de configuraciones en el roadmap es excesivo, ya que habrá un número mayor de llamadas, aunque cada llamada sea barata [6].

La selección del planeador local adecuado afecta también a la fase de consulta, ya que es importante poder conectar cualquier configuración inicial y final al roadmap o bien detectar rápidamente que tales conexiones no son posibles, si el planeador local es rápido es posible ocupar éste mismo para conectar éstas configuraciones.

El planeador local usado en esta tesis conecta dos configuraciones dadas en línea recta, discretizando la distancia existente entre ellas de acuerdo a un parámetro definido por el usuario y verificando en cada una de éstas partes si no existe colisión entre el robot y los

obstáculos del ambiente. Si una colisión se detecta inmediatamente el planeador reporta que la conexión entre las configuraciones no puede ser establecida.

Discretizar la distancia significa calcular el conjunto de configuraciones c_1, …, c_m existentes entre las configuraciones de entrada para posteriormente verificar si el robot posicionado en cada configuración c_i está libre de colisión. Si ninguna de las m configuraciones produce colisión se concluye que la ruta está libre de colisión. A este tipo de planeador se le conoce como *planeador local de línea recta* (ver figura 2.1).

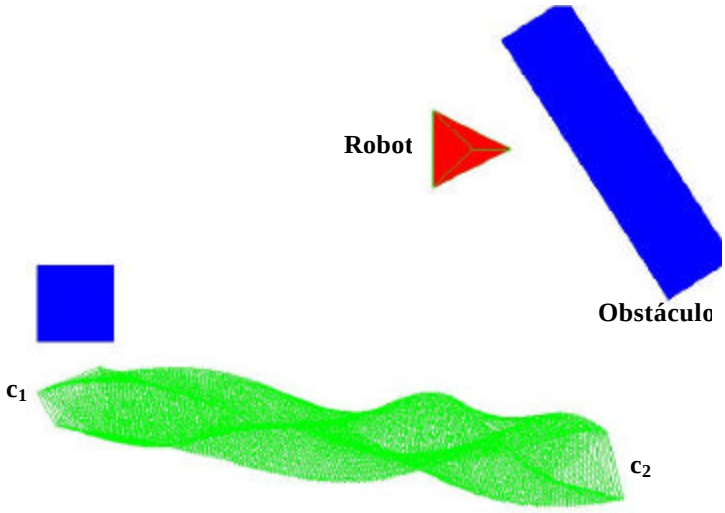

Figura 2.1 Ruta generada entre dos configuraciones c_1 y c_2 utilizando el planeador local de línea recta.

Existen versiones modificadas del planeador local de línea recta conocidas como *rotar en s* o por su término en inglés "*rotate-at-s*" donde ($0 < s = 1$) explicados brevemente a continuación:

2.1.1 Rotar en s

Cuando el robot se mueve de c_1 a c_2 el planeador *rotar en s* primero se traslada desde c_1 a una configuración intermedia *c'*, rota a una segunda configuración intermedia *c''* y finalmente se traslada a c_2. El parámetro *s* representa la parte fraccional de la distancia que el robot se traslada desde c_1 a *c'*. El planeador de línea recta se usa para verificar la

conexión entre cada par de configuraciones, es decir entre (c_1, c'), entre (c', c'') y entre (c'', c_2) [8].

El planeador *rotar en 1/2* hace más conexiones que el planeador de línea recta debido a que tiene un volumen de barrido más pequeño y por lo tanto toma el doble de tiempo. Las otras dos versiones del planeador *rotar en s* son cuando $s = 0$ y $s = 1$ los cuales aunque no son muy exitosos hacen algunas conexiones que los otros dos planeadores anteriores no.

2.2 MÉTODO PROBABILÍSTICO DE CARRETERAS

Método desarrollado en 1991 por el *Dr. Jean-Claude Latombe* y su estudiante Lydia E. Kavraki en la *Universidad de Stanford* y de manera independiente por los investigadores Mark H. Overmars y Petr Svestka de la Universidad de Utrecht, el cuál es conocido por sus términos en inglés como *Probabilistic Roadmap Method* ó *PRM* y cuyo objetivo es encontrar rutas libres de colisión para robots de cualquier tipo en espacios de trabajo estáticos, es decir ambientes donde los obstáculos no se mueven. El método consta de dos fases: *la fase de construcción* y la *fase de consulta* [6].

En la fase de construcción se genera un mapa probabilístico de carreteras conocido típicamente por su término en inglés como "**roadmap"**. Primero se generan aleatoriamente *n* configuraciones libres para el robot y posteriormente se intentan conectar a través de un simple y rápido planeador local. El roadmap construido se encuentra en el espacio libre y se representa como un grafo no dirigido. Las configuraciones generadas constituyen los nodos del grafo y las rutas calculadas por el planeador local sus aristas.

Una vez construido el roadmap se pueden realizar múltiples consultas, una consulta pregunta si existe una ruta entre dos configuraciones cualesquiera del robot. Para ejecutar una consulta el método primero intenta conectar las configuraciones inicial y final a nodos en el grafo, si esto se logra se realiza un búsqueda en el grafo a partir de los nodos con los cuales hubo conexión para encontrar la secuencia de aristas que conectan a estos nodos.

Finalmente la concatenación de los segmentos sucesivos se transforma en una ruta factible para el robot. Es decir la ruta esta formada por tres subrutas, la subruta que conecta la configuración inicial con un nodo del grafo, la subruta presente en el grafo entre los dos nodos de conexión y la subruta que conecta un nodo del grafo con la configuración final. El pseudocódigo del método se muestra a continuación:

Método probabilístico de carreteras

I. Fase de construcción del roadmap

1. Generación de nodos (encontrar configuraciones libres de colisión)
2. Conexión (conectar nodos para formar el roadmap)

II. Fase de consulta

1. Conectar las configuraciones inicial y final al roadmap
2. Encontrar una ruta entre los nodos de conexión del roadmap.

Las fases de construcción y de consulta no tienen por que ser ejecutadas secuencialmente al contrario ambas pueden ser intercaladas hasta que el roadmap alcanza un tamaño adecuado que indique que se ha capturado completamente la conectividad del espacio libre. En un principio el método construye un roadmap pequeño el cuál continuamente va aumentando conforme el tiempo se incrementa, paralelamente la fase de consulta puede ser realizada.

Antes de ejecutar el método el valor de algunos parámetros debe ser seleccionado, por ejemplo el tiempo que va a ser consumido en la fase de construcción o el número de configuraciones libres a generar. Estos valores dependen del espacio de trabajo y del tipo de robot, sin embargo los mejores resultados se obtienen cuando se involucran intervalos largos de tiempo. Después de algunos experimentos preliminares no es difícil seleccionar un conjunto de valores satisfactorios para un espacio de trabajo dado o para una familia de ellos.

La eficiencia de éste método se ha demostrado aplicándolo a un número diverso de problemas de planeación de movimientos que involucran diferentes tipos de robots, en todos los casos los resultados experimentales mostraron que los tiempos requeridos para la

construcción de un roadmap adecuado (*que capture completamente la conectividad del espacio libre*) fueron cortos, desde pocos segundos para los problemas fáciles hasta algunos minutos para los más difíciles; construido el roadmap las consultas son procesadas en fracciones de segundos.

Los tiempos pequeños de consulta hacen el método viable para robots de muchos grados de libertad, por ejemplo robots que realizan tareas como el mantenimiento de tuberías en plantas nucleares, la soldadura punto a punto en el ensamblado de autos y la limpieza del fuselaje de aviones, en tales tareas muchos grados de libertad son necesarios para alcanzar las configuraciones deseadas del *"end-efector"* (parte final de un robot articulado) evitando así colisiones con el resto del robot en un ambiente complicado [6].

2.2.1 Método general

El roadmap probabilístico generado durante la fase de construcción es un grafo no dirigido $R = (N, E)$, donde los nodos en N son el conjunto de configuraciones libres de colisión para el robot y las aristas en E corresponden a simples rutas entre cada par de nodos.

Una arista (a, b) corresponde a una ruta factible (*se encuentra en el espacio libre*) entre las configuraciones a y b, estas rutas se conocen como *rutas locales* ya que son calculadas a través de un planeador local, dado que éste es determinístico las rutas locales no son almacenadas en el roadmap, básicamente para ahorrar espacio y debido a que pueden ser fácilmente recalculadas [6].

En la fase de consulta el roadmap se usa de la siguiente manera: dadas una configuración inicial s y una configuración final g el método primero intenta conectar s y g a dos nodos s' y g' de N, si esto ocurre, entonces se busca en R una secuencia de aristas de E que conecten a s' y g'. Finalmente la secuencia de aristas encontrada se transforma en una ruta factible para el robot recalculando las rutas locales y concatenándolas, de manera más detallada se describen a continuación las fases anteriores:

2.2.2 Fase de construcción

Esta fase tiene como objetivo obtener una carretera lo suficientemente conectada que cubra completamente y de manera uniforme el espacio libre, asegurando así que *regiones difíciles* como es el caso de los pasajes estrechos, contengan al menos unos cuantos nodos.

Algunas versiones del método probabilístico de carreteras incluyen en su fase de construcción una etapa llamada de *expansión* cuyo objetivo es mejorar la conectividad del grafo; consiste en seleccionar nodos de R que se encuentren en las regiones difíciles del espacio de configuraciones para posteriormente buscar expandir el grafo a partir de ellos, generando nodos adicionales en sus vecindades. Por este motivo después de la ejecución de esta etapa la cobertura del espacio libre en el roadmap no siempre es uniforme.

a).- Etapa de generación de configuraciones

Al inicio de la fase de construcción $R = (N, E)$ está vacío. De manera aleatoria se buscan configuraciones libres en el espacio de trabajo que luego son adicionadas a N, estos nodos deben representar un muestreo uniforme del espacio libre (ver figura 2.2).

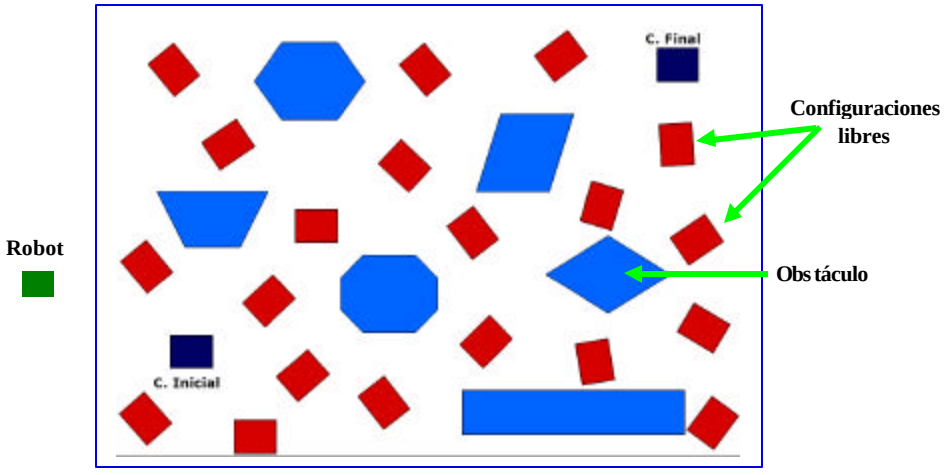

Figura 2.2 Configuraciones libres (rojo) para un robot, cuya forma es un cuadrado, se generan aleatoriamente de manera que se logre una cobertura uniforme para el espacio libre.

b).- Etapa de conexión de nodos

Para cada nodo c se selecciona un número de nodos de N y cada uno de ellos se trata de conectar a c usando un planeador local, si el planeador logra una ruta factible entre c y un nodo seleccionado n, la arista (c, n) es adicionada a E (ver figura 2.3). La ruta local no es almacenada en el roadmap.

Para seleccionar que nodos trataremos de conectar con c primero se escoge un conjunto N_c de vecinos candidatos de N. Este conjunto se encuentra dentro de cierta distancia a c y los nodos son colocados en orden ascendente de su distancia a c. Posteriormente se trata de conectar a c con cada uno de los nodos escogidos, el planeador local es llamado a conectar en cada caso y el costo acumulativo de estas invocaciones domina siempre el tiempo de la fase de construcción.

El planeador local se invoca para verificar la conexión entre parejas de configuraciones cuya distancia relativa (de acuerdo a una función de distancia D) sea más pequeña que alguna constante [6] se define:

$$N_c = \{ c' \in N \mid D\ (c, c') \ =\ Maxdist \}$$

Se trata de conectar c con todos los nodos de N_c en orden ascendente de su distancia con respecto a c, la función D se usa tanto para construir como para ordenar el conjunto N_c de vecinos candidatos de cada nuevo nodo c. Para el caso particular de esta tesis la función D está definida por la distancia euclidiana entre dos configuraciones y el conjunto N_c contiene k nodos de N, donde k es seleccionada por el usuario.

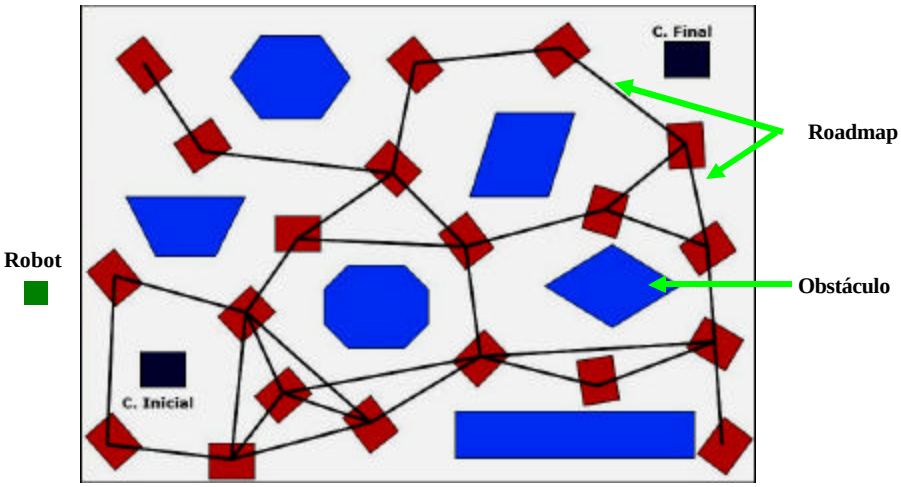

Figura 2.3 Las rutas locales determinan las aristas del grafo formado en la fase de construcción.

c).- Etapa de expansión

Si el número de nodos libres generados es suficientemente grande el espacio libre será cubierto uniformemente y entonces *R* estará bien conectado. En el caso contrario *R* estará compuesto de muchos componentes, algunos grandes y otros pequeños, indicando que no se ha capturado eficientemente la conectividad del espacio libre.

La etapa de expansión intenta mejorar la conectividad del grafo generado en la fase de construcción, si el grafo está desconectado en algún lugar, ése lugar corresponde a una región difícil, la idea de esta etapa es seleccionar un número de nodos de esa región y expandirlos.

Expandir una configuración *c* significa generar una nueva configuración libre en la vecindad de *c* (*la nueva configuración se encuentra dentro de cierta distancia a c*), adicionar ésta a *N* y tratar de conectarla con otros nodos del grafo de la misma manera que en la etapa de conexión. Esta etapa incrementa la densidad de las configuraciones del roadmap en regiones consideradas difíciles; dado que los huecos entre componentes de *R* están localizados en estas regiones la conectividad del roadmap se mejora indudablemente.

Una alternativa para llevar a cabo esta etapa es proponer un esquema probabilístico en el cual a cada nodo se le asocie un *peso* que indique una medida heurística de la "*dificultad*" de la región en que se encuentra, si el peso es grande significa que la región es muy difícil.

Para definir el peso heurístico se determina el número de nodos comprendidos dentro de cierta distancia a una configuración dada o bien de acuerdo al comportamiento del planeador local, por ejemplo si el planeador falla frecuentemente para conectar una configuración dada a otros nodos, esto indica que la configuración se encuentra en una región difícil.

En esta tesis para expandir un nodo *c* se ejecuta una caminata aleatoria ("*short random-bounce walk*") iniciada desde *c*. La caminata aleatoria consiste en repetidamente tomar direcciones aleatorias de movimiento en el espacio de configuraciones y moverse en esa dirección hasta que un obstáculo es golpeado, cuando una colisión ocurre una nueva dirección aleatoria es seleccionada; la configuración final *n* alcanzada por la caminata y la secuencia de aristas desde *c* a *n* es almacenada dentro del roadmap, ya que esta técnica no es determinística [6].

Sabemos que la configuración *n* pertenece al mismo componente conectado de la configuración *c*, entonces se trata de conectar *n* a otro componente del grafo de la misma manera que en la etapa de conexión. La etapa de expansión no crea nuevos componentes en el roadmap pero si puede fallar al tratar de reducirlos.

Por otra parte, para mejorar aún más la conectividad del roadmap se rota *n* veces esta configuración *c*, buscando nuevas configuraciones libres en la misma posición de *c* para luego intentar conectarlas a *R*, el parámetro *n* es también definido por el usuario.

2.2.3 Fase de consulta

Durante la fase de consulta se buscan rutas entre configuraciones inicial y final arbitrarias usando el roadmap generado en la fase de construcción. Se asume por el momento que el

espacio libre está conectado y que el roadmap consiste de un solo componente. Dadas las configuraciones inicial *s* y final *g* se trata de conectar *s* y *g* a dos nodos de *R*, respectivamente *s'* y *g'* y luego se busca una ruta entre estos nodos a través de un algoritmo de búsqueda en grafos; si esto falla la consulta también falla (ver figura 2.4).

La principal pregunta es cómo calcular estas subrutas. La estrategia para conectar *s* a *R* es considerar nodos de *R* en orden ascendente de sus distancias con respecto a *s* (de acuerdo a la función *D*) y tratar de conectar cada uno de éstos a *s* con la ayuda del planeador local hasta que una conexión suceda. Se descartan los nodos más lejanos que la constante definida desde *s* porque la oportunidad de éxito del planeador local en éste caso es muy baja.

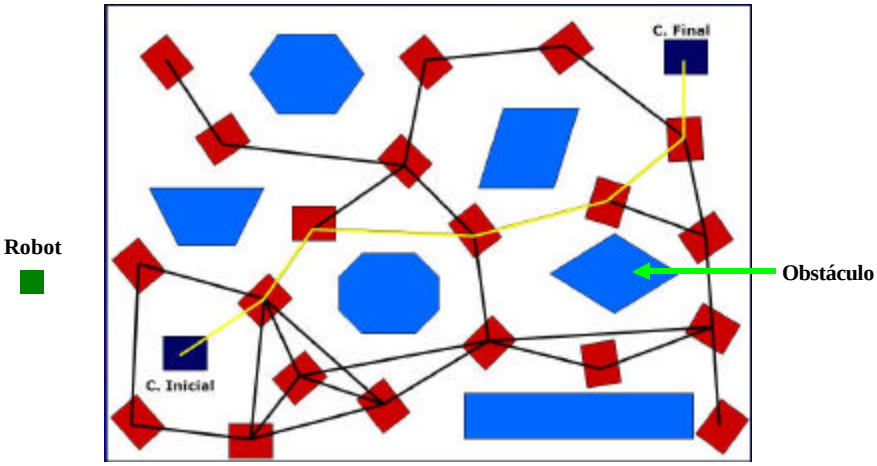

Figura 2.4 La fase de consulta busca la ruta entre las configuraciones inicial y final (línea amarilla) si esta existe; en caso contrario reporta una falla.

Si todas las conexiones fallan es posible ejecutar una o más caminatas aleatorias, tomando en cuenta que además de adicionar el nuevo nodo a la subruta al final de cada caminata también se trata de conectar con algún nodo de *R* usando el planeador local. Tan pronto como se encuentre conexión de *s* a *R* se aplica el mismo procedimiento para *g* [6].

Una vez que las configuraciones inicial y final se han conectado al roadmap y se tienen identificados los nodos *s'* y *g'*, se lleva a cabo la búsqueda de ruta entre éstos nodos a través de un algoritmo de búsqueda en grafos. Los algoritmos mayormente empleados son *búsqueda primero en amplitud* ("breadth first search") y *búsqueda primero en profundidad* ("depth first search").

Si empleamos el primer algoritmo obtenemos un árbol cuya raíz es el vértice dado como recurso, si este componente del grafo representa la conectividad de todo el espacio libre el árbol construido por el algoritmo es suficiente, pero si esto no ocurre existirán nodos que no pertenecerán a ningún componente. En el caso del segundo algoritmo el resultado es un bosque, cada nodo sin excepción es visitado y de acuerdo al bosque obtenido podemos saber si se ha alcanzado a cubrir toda la conectividad del espacio libre. Sin embargo la decisión respecto al uso de uno de estos algoritmos queda en manos del usuario; para el caso particular de esta tesis empleamos el segundo algoritmo.

Como una mejora a las rutas obtenidas en esta tesis, también se empleó un algoritmo de búsqueda de ruta mínima conocido como *Dijkstra*, donde el grafo recurso es dirigido y sus aristas tienen pesos. En este caso, el roadmap construido es dirigido y los pesos de las aristas corresponde a la distancia euclideana existente entre los nodos que las forman.

Reconstruir la ruta para el robot dada la secuencia de nodos encontrada se reduce a la concatenación de las rutas locales calculadas por el planeador local en los nodos adyacentes de *R*. Si algunas de estas rutas fueron producidas por la caminata aleatoria durante la fase de construcción ésta ya se encuentran almacenadas en las aristas de *R*.

Esta ruta tal y como se observa en la figura anterior consiste en la concatenación de tres subrutas, la subruta desde la configuración inicial a un nodo *s'* del roadmap, la subruta entre los nodos *s'* y *g'* del roadmap y la subruta desde un nodo *g'* del roadmap a la configuración final.

Si la consulta falla frecuentemente indica que el roadmap construido hasta ese momento no ha capturado bien la conectividad del espacio libre, entonces será necesario dedicar más tiempo a la fase de construcción. Sin embargo no es necesario construir un nuevo roadmap desde el comienzo, dado que la fase de construcción es incremental podemos simplemente ampliar el actual roadmap ejecutando de nuevo las etapas de generación de nodos, de conexión y de expansión a partir del grafo ya obtenido, intercalando así la fase de construcción con la consulta.

2.3 MÉTODO PROBABILÍSTICO DE CARRETERAS BASADO EN OBSTÁCULOS

Método desarrollado por la *Dra. Nancy Amato* en la *Universidad de Texas* en 1996 y conocido por sus términos en inglés como *Obstacled based PRM* o *OBPRM* y cuyo objetivo es obtener roadmaps de alta calidad aún cuando el espacio de trabajo está muy poblado de obstáculos. Su principal característica es que los nodos candidatos del roadmap son seleccionados alrededor de la superficie de los obstáculos, como consecuencia este roadmap contendrá rutas en regiones difíciles como por ejemplo aquellas que atraviesan pasajes estrechos [7].

El método de manera general sigue el tradicional *método probabilístico de carreteras* y se caracteriza por la forma de generar nodos candidatos al roadmap. Se busca generar nodos distribuidos uniformemente alrededor de la superficie de cada obstáculo, para producir mejores roadmaps que si los nodos se generaran uniformemente en todo el espacio de trabajo; esto se muestra en la siguiente figura:

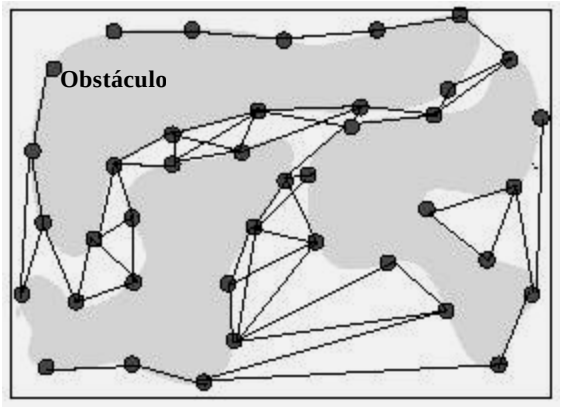

Figura 2.5 Los nodos en el roadmap deben estar uniformemente distribuidos alrededor de la superficie de cada obstáculo [7].

Overmars y Svestka [9] describen una técnica llamada *adición de nodo geométrico* en la cual los nodos del roadmap son generados cerca de los límites del obstáculo, aunque los autores observan que la técnica produce mejores roadmaps que la generación uniforme de configuraciones, tal técnica no puede ser aplicada a robots articulados.

El método probabilístico para encontrar los nodos candidatos al roadmap es más costoso que la generación uniforme de nodos en el espacio de configuraciones, sin embargo resultados experimentales muestran que el tiempo consumido en cálculos adicionales durante la etapa de generación de configuraciones es mínimo en comparación con el tiempo total de la fase de construcción.

2.3.1 Etapa de generación de nodos candidatos

Durante la fase de construcción se genera un conjunto de nodos candidatos al roadmap; la estrategia general consiste en generar un conjunto de nodos candidatos para cada obstáculo, tal que cada uno de ellos se encuentre alrededor y cerca de la superficie del obstáculo [7].

El conjunto de nodos candidatos al roadmap es la unión de los conjuntos candidatos calculados para cada obstáculo. El conjunto de nodos candidatos para cada objeto debe estar distribuido uniformemente alrededor del obstáculo de acuerdo a alguna medida definida en el espacio de configuraciones como por ejemplo lo distancia euclideana.

El número de nodos que se desean obtener alrededor de cada obstáculo es un parámetro conocido y establecido por el usuario. Para evitar el cálculo costoso de la superficie del obstáculo se propone un método heurístico para generar los nodos [7]:

1. Generar aleatoriamente un número *n* de configuraciones libres para un obstáculo *s*.

2. Generar aleatoriamente un número *n* de configuraciones en colisión con el obstáculo *s*.

3. Para cada par de configuraciones (*una libre y otra en colisión*) emplear búsqueda binaria para encontrar una configuración libre lo más cerca de la superficie del obstáculo *s*.

4. Repetir los pasos anteriores para cada obstáculo del espacio de trabajo.

La etapa de generación de configuraciones se ilustra en la siguiente figura:

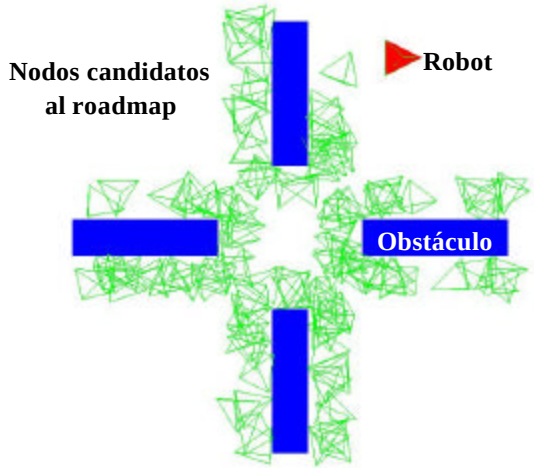

Figura 2.6 Generación uniforme de configuraciones alrededor de los obstáculos.

Es claro que la forma de los obstáculos y las configuraciones generadas aleatoriamente tendrán gran influencia en la calidad de la distribución de las configuraciones resultantes.

2.3.2 Etapa de conexión

Esta etapa es igual que en el *método probabilístico de carreteras* y la conectividad del roadmap depende del número y distribución de los nodos candidatos, lo efectivo del planeador local y el número de conexiones intentadas para cada nodo candidato.

La estrategia de conexión usada consiste en tratar de conectar cada nodo con los *k* nodos vecinos más cercanos (*se encuentran dentro de cierta distancia definida por la función D*) pertenecientes a cada uno de los conjuntos de nodos generados para cada obstáculo.

Es importante determinar el número de nodos candidatos a generar y el número de conexiones a intentar para cada nodo al formar el roadmap, por ejemplo se puede intentar un número pequeño de conexiones para cada nodo pero si el roadmap resultante no esta suficientemente conectado será necesario realizar conexiones adicionales. Este proceso

podrá ser iterado hasta que el roadmap se considere estar bien conectado o hasta que un mejoramiento haya sido logrado, un ejemplo se ilustra en la figura 2.7.

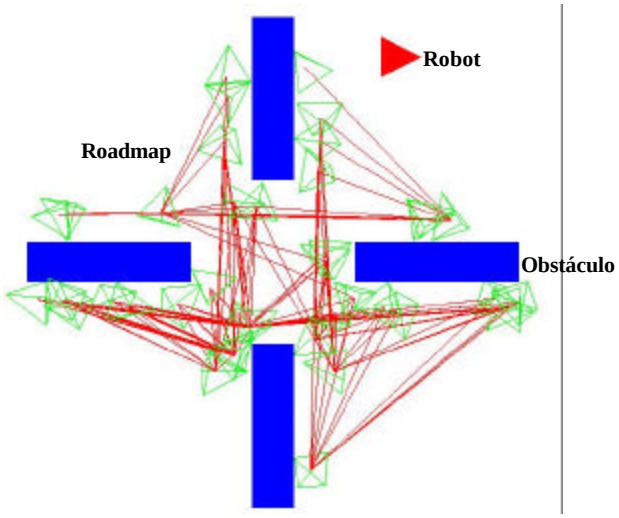

Figura 2.7 Roadmap generado en la fase de construcción.

2.3.3 Fase de consulta

Esta fase, igual que en el método explicado anteriormente, busca conectar con ayuda del planeador local las configuraciones inicial y final al mismo componente conectado del roadmap y encontrar una ruta en él entre los dos nodos de conexión. Las conexiones son intentadas con los nodos *k* mas cercanos del roadmap, si esto no es posible entonces se ejecuta una caminata aleatoria intentando conectar el nodo final de la caminata a un nodo del grafo. Después de que ambas conexiones son realizadas se lleva a cabo la búsqueda de ruta en el roadmap entre los dos puntos de conexión usando un algoritmo de búsqueda en grafos.

La ruta final estará formada por la concatenación de tres subrutas, una subruta desde la configuración inicial a un nodo del roadmap, una subruta contenida dentro del roadmap entre los dos puntos de conexión para las configuraciones inicial y final y la subruta desde un nodo del roadmap a la configuración final.

2.4 MÉTODO PROBABILÍSTICO DE CARRETERAS BASADO EN VISIBILIDAD

Método desarrollado por el *Dr. Jean - Paul Laumond* en el *Laboratorio de Análisis y Arquitectura de Sistemas* (*IAAS*) en *Toulouse, Francia* en 1999 y conocido por sus términos en inglés como *Visibility based Probabilistic Roadmap*. Mientras que en el método probabilístico de carreteras cada configuración libre generada se conectada al roadmap, en el método *visibilidad* (*denominado así de ahora en adelante para mayor comodidad*) se conservan solo configuraciones que conecten al menos dos componentes distintos del roadmap (llamadas *connections*) y además aquellas que no son visibles a ninguna otra (*guards*) [10].

El roadmap construido por este método tiene un número pequeño de nodos y se conoce como *grafo de visibilidad*; el roadmap se construye incrementalmente explorando el espacio de configuración e intentando conectar parejas de configuraciones a través de un planeador local.

2.4.1 Dominio de visibilidad

Consideremos un robot en un espacio de trabajo definido por un conjunto de obstáculos, *CS* denota el espacio de configuración del ambiente y CS_{free} es el espacio libre definido por el subconjunto de configuraciones libres de colisión.

Dado el planeador local *L*, el dominio de visibilidad de una configuración *q* esta definido como el dominio [10]:

$$VisL_{(q)} = \{\ q' \in Cs_{free} \mid L(q, q') \subset CS_{free}\ \}$$

Un conjunto de *guards* proporciona la cobertura de CS_{free} si la unión de cada uno de sus dominios de visibilidad cubre el espacio libre CS_{free}. La existencia de una cobertura finita depende de la forma de CS_{free} y del planeador local empleado (ver figura 2.8).

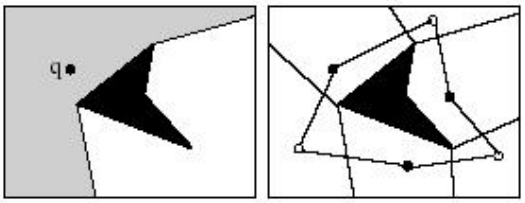

Figura 2.8 Dominio de visibilidad de una configuración y el roadmap de visibilidad construido con tres nodos *guards* (negros) y tres nodos *connections* (blancos) [10].

Cuando un nodo q_i no ve, es decir no conecta a través de un planeador local a otro nodo q_j se le llama *guard*, $L(q_i, q_j) \not\subset CS_{free}$ y se adiciona al roadmap como un nodo. Para dos dominios de visibilidad que se intersecten $Vis_{L(q_1)}$ y $Vis_{L(q_2)}$ se crean un nodo q llamado *connection* y dos aristas (q, q_1) y (q, q_2) que se adicionan también al grafo [10]. El roadmap resultante esta compuesto de nodos *guard* y de nodos *connections*.

De manera general en el método visibilidad se generan aleatoriamente configuraciones libres de colisión las cuales son adicionadas al roadmap, si la configuración no *ve* a cualquier otra configuración del grafo actual se considera como un nuevo nodo *guard*; si la configuración ve al menos dos nodos pertenecientes a distintos componentes del roadmap se considera como un nodo *connection* (ver figura 2.9).

La etapa de construcción del roadmap se controla a través de un parámetro definido por el usuario que significa el número de intentos necesarios para lograr cubrir completamente el espacio libre o bien un tiempo estimado para que se alcance tal cobertura, dependiendo de la implementación hecha para el método; en esta tesis la etapa de construcción del método depende del número de intentos que se deseen hacer.

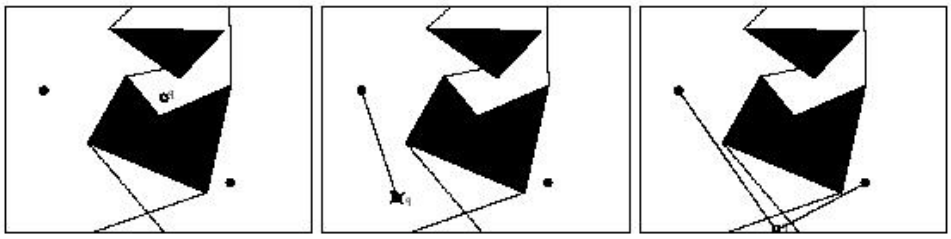

Figura 2.9 Tres casos: una configuración libre generada aleatoriamente será un nuevo *guard* insertado en el roadmap; una configuración rechazada ya que se encuentra en el mismo dominio de visibilidad y un nodo *connection* que une dos componentes disconexos del grafo [10].

El grafo de visibilidad ha mostrado ser eficiente en espacios de trabajo que contienen pasajes estrechos; con respecto al método *PRM* su principal ventaja es el tamaño pequeño del roadmap construido, como se muestra en la figura 2.10.

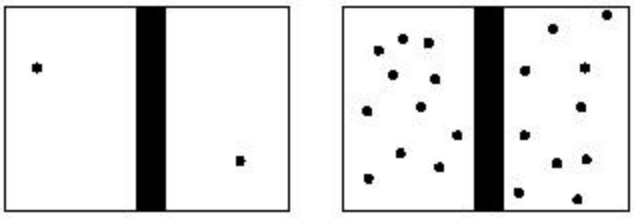

Figura 2.10 Número de nodos en el grafo de visibilidad en comparación con el número de nodos en el grafo del *PRM* [11].

En todos los *métodos probabilísticos de carreteras* se requieren costosas operaciones para verificar si una configuración o una ruta encontrada por un planeador local se encuentran libre de colisión; la estrategia empleada por el método de visibilidad para la generación de configuraciones es más costosa que la generación uniforme de configuraciones en el espacio libre llevada a cabo por éstos métodos.

En lugar de que cualquier configuración libre de colisión sea agregada al roadmap el método de visibilidad rechaza algunas de éstas antes de que un nuevo *guard* se encuentre. Sin embargo podemos notar que el verificar si una configuración es libre de colisión es menos costoso que intentar realizar conexiones al roadmap, dado que este paso requerirá repetidas llamadas al verificador de colisiones a lo largo de toda la ruta generada por el planeador local empleado.

La etapa de conexión consume la mayor parte del tiempo de la ejecución de los métodos probabilísticos, principalmente cuando el tamaño del roadmap es demasiado grande, sin embargo dado el tamaño pequeño del roadmap que construye el método de visibilidad el tiempo extra consumido durante la etapa de selección de las configuraciones se compensa por una ganancia notable de rendimiento cuando se establecen las conexiones al roadmap.

Existen casos en donde el método tiene algunas deficiencias, por ejemplo cuando el dominio de intersección de todos los dominios de visibilidad es demasiado pequeño y la única manera de completar el roadmap es encontrando una configuración *connection* en esa pequeña área.

El algoritmo falla si el parámetro de cobertura del espacio libre no es lo suficientemente grande, sin embargo la probabilidad de que esto ocurra en un espacio de trabajo real es muy pequeña. Otro caso aparece cuando la probabilidad de colocar una configuración *guard* en un pasaje demasiado estrecho es muy baja dadas las características del ambiente; estos dos casos se ilustran mejor en la siguiente figura:

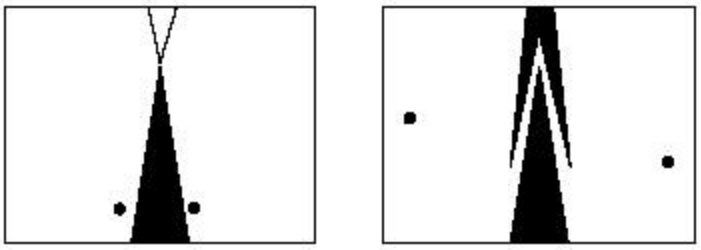

Figura 2.11 Dos situaciones difíciles a resolver por el método de visibilidad, en el primero los dominios de visibilidad tienen muy poca intersección y en el segundo caso la existencia de un corredor estrecho dificulta la captura total del espacio libre para cualquier muestreo aleatorio [11].

Construido el roadmap la fase de consulta se ejecuta de igual manera que en el método probabilístico de carreteras y la ruta final consisten en la concatenación de las tres subrutas explicadas anteriormente.

2.5 COMPARACIÓN CUALITATIVA DE LOS MÉTODOS PROBABILÍSTICOS

Dada la descripción hecha de cada uno de los métodos probabilísticos abordados en éste capítulo podemos concluir que cada uno de ellos ofrece diferentes técnicas para atacar el problema de planeación de movimientos y la presencia de pasajes estrechos en el espacio de trabajo.

Sin embargo para afirmar cuál método es mejor que otro correspondería la realización de un análisis detallado de cada uno de ellos en cuanto a su efectividad al encontrar soluciones en diferentes espacios de trabajo con diferentes tipos de robots.

Por otra parte podemos resaltar algunas características de cada uno de ellos:

a. El método *Visibilidad* genera un roadmap pequeño con las configuraciones necesarias para capturar la conectividad del espacio libre del espacio de trabajo, con respecto a los otros dos métodos. Sin embargo el proceso de selección de configuraciones y la mezcla de componentes que el método realiza aumentan considerablemente la etapa de conexión del roadmap.

b. El método OBPRM permite atacar directamente la presencia de pasajes estrechos en la etapa de generación de configuraciones, sin embargo el roadmap resultante es muy similar al producido por el método PRM.

c. El método PRM es uno de los métodos más fáciles de implementar y su filosofía es la base de muchos otros, sin embargo dadas las características anteriormente descritas produce los roadmap más densos y éstos no aseguran capturar de manera completa todo el espacio libre del espacio de trabajo.

Por otra parte es importante señalar que tal y como se han descrito cada uno de éstos métodos corresponden a la implementación más básica, existen numerosos trabajos dónde la implementación de cada uno de ellos se ha mejorado con diferentes técnicas, mejorando aún más su efectividad y las cuales no son descritas en éste trabajo.

La selección de estos métodos probabilísticos para intentar solucionar algunas de las versiones del problema de los clavos obedeció básicamente a que sus autores afirman haber solucionado dicho problema con éstos métodos, esto será explicado de manera detallada en la sección 3.2 del siguiente capítulo.