Revista Iberoamericana de Automática e Informática Industrial
ISSN: 1697-7912. Vol. 3, Núm. 3, Julio 2006, pp. 56-67
http://riai.isa.upv.es

PLANIFICACIÓN DE TRAYECTORIAS CON EL ALGORITMO RRT. APLICACIÓN A ROBOTS NO HOLÓNOMOS

D. López [**] **F. Gómez-Bravo** [**] **F. Cuesta** [*]
A. Ollero [*]

[*] *Grupo de Robótica, Visión y Control. Univ. de Sevilla.*
Camino de los Descubrimientos, 41092 Sevilla (Spain).
{fede,aollero}@cartuja.us.es

[**] *Escuela Politécnica Superior, Universidad de Huelva.*
Carretera de Palos-La Rábida s/n. 21071 Huelva.
{diego.lopez,fernando.gomez}@diesia.uhu.es

Resumen: Dentro de los métodos de planificación de trayectorias, las técnicas basadas en el algoritmo RRT (*Rapidly Exploring Random Trees*) están deparando especial interés. En este artículo se describen los fundamentos de este novedoso algoritmo. Asimismo, se detallan las versiones más significativas aportadas por la bibliografía más reciente. Se ilustra también su aplicación en sistemas robóticos no holónomos. Finalmente se presenta una técnica basada en el concepto de maniobra restringida y su aplicación tanto a vehículos con guiado diferencial como con configuración Ackerman.

Palabras Clave: Planificación de caminos, robots móviles, sistemas no holónomos, métodos aleatorios.

1. INTRODUCCIÓN

De forma genérica, el problema de planificación de movimientos consiste en llevar un cuerpo, desde una configuración inicial hasta otra final dentro del espacio de configuraciones libres de colisión. Este problema ha sido ampliamente abordado en la literatura (Latombe, 1991; Laumont *et al.*, 1994; Muñoz, 1995), existiendo un gran número de métodos efectivos para resolver el problema de la planificación en tiempo real, tales como campos potenciales, grafos de visibilidad, diagramas de Voronoi, etc. No obstante, cuanto más complejo es el entorno, mayor el número de grados de libertad, o más restricciones cinemáticas presenta el robot, mayores son también las limitaciones de los métodos y el tiempo necesario para encontrar la solución.

Por ejemplo, un conocido método de planificación es el de campo de potenciales, el cual también es utilizado en aplicaciones de control reactivo. Este método, pese a ser rápido, presenta el problema de la existencia de mínimos locales. Para solucionar esta limitación se recurre a algoritmos de generación aleatoria que permiten realizar planificaciones locales hacia un nuevo mínimo (Latombe, 1991).

Esta situación ha impulsado la búsqueda de métodos de planificación puramente aleatoria con objeto de lograr mayor velocidad en la obtención del resultado. Entre los más extendidos se encuentra el denominado "Rapidly Exploring

Random Trees"(LaValle, 1998), abreviadamente RRT. Sobre éste han aparecido diferentes versiones que intentan mejorarlo, obteniendo eficientes avances computacionales, como el "RRT-Ext-Con"(Kuffner y LaValle, 2000; LaValle y Kuffner, 2001), o aportando mejoras para su aplicación en escenarios específicos, como es el caso del ERRT (Bruce y Veloso, 2002). Este último resulta especialmente indicado en escenarios en los cuales los obstáculos den lugar a zonas conectadas a través de pasillos estrechos.

Este artículo introduce el algoritmo RRT. De este modo, se detallan las principales implementaciones y variantes de dicho método. Asimismo se ilustra su aplicación en vehículos no holónomos para facilitar la comprensión del método. En concreto, el método propuesto se aplica al caso de un robot móvil de conducción diferencial, si bien, el procedimiento es extensible con facilidad a robots de cinemática más compleja.

El artículo está organizado como sigue: en primer lugar se introducen los métodos de planificación con generación aleatoria. En la sección 3 se detalla el algoritmo RRT y diversas variantes del mismo. La sección 4 está dedicada a ilustrar un método de postprocesado de la solución. En el apartado 5 se presenta la aplicación del método a sistemas no holónomos. Asimismo, se propone una alternativa a los métodos existentes basada en el concepto de *maniobra restringida*, incluyendo resultados y comparativas. El artículo finaliza con las conclusiones y las referencias.

2. PLANIFICACIÓN CON GENERACIÓN ALEATORIA

Gracias al desarrollo del método de campos potenciales surgió una primera aproximación a los métodos de planificación aleatoria (Latombe, 1991). En este tipo de métodos, se parte de un escenario donde el espacio de configuraciones se encuentra dividido en porciones discretas estableciendo una rejilla regular. A cada casilla se le concede un valor de potencial dependiendo de su proximidad a un obstáculo o la cercanía a los puntos origen o destino. El planificador genera una trayectoria merced a un vector gradiente derivado del campo potencial. Esta técnica permite que el sistema siga siempre la dirección que minimiza el valor del campo potencial. El objetivo es alcanzar el mínimo absoluto que estará situado en la configuración destino. Sin embargo, el campo resultante puede tener una serie de mínimos locales, superiores siempre al mínimo absoluto. El problema aparece cuando el sistema alcanza un mínimo local. En este caso, el potencial no es nulo, pero el gradiente mantiene al sistema en la configuración alcanzada. Para resolver este inconveniente se recurre a un método de generación aleatoria. Se planifican movimientos aleatorios que permitan que el sistema abandone el mínimo y, a continuación, se aplica de nuevo el método del gradiente. Este proceso continuará hasta hallar un nuevo mínimo (Latombe, 1991).

El modo de generar caminos consistía en crear una sucesión de desplazamientos aleatorios. En éstos, cada coordenada podía experimentar un incremento positivo o negativo de una longitud constante dada. Seguidamente, se comprobaba la existencia o no de colisión y en función de ello se agregaban al camino. Este proceso se repetía un número de iteraciones determinado y, a partir del último punto, se volvía al algoritmo de campo de potencial original.

El camino generado debía ser lo suficientemente largo como para abandonar cualquier mínimo local. No obstante es deseable que dicho camino no sea excesivamente largo, para evitar costes computacionales innecesarios. Por tanto, el método utilizado determinaba también un tamaño aleatorio para la longitud del camino de salida. De este modo se mantenía la capacidad de escape de cualquier mínimo, al tiempo que se proporcionaba mayor probabilidad a las longitudes menores. Estos métodos utilizan una distribución de Laplace, por ser de máxima entropía para la selección aleatoria de las magnitudes (Latombe, 1991).

El éxito de los métodos aleatorios sugirió la posibilidad de usar dichas técnicas de forma exclusiva, eliminando el coste del procesamiento para el cálculo del potencial. Habían de ser más simples, para competir en velocidad y suplir la carencia de una inteligencia en la búsqueda de caminos. Uno de estos métodos es el llamado "Rapidly Exploring Random Trees", RRT (LaValle, 1998).

El RRT no precisa el establecimiento de un campo de potencial, con el consecuente ahorro del procesamiento. Al mismo tiempo, el RRT asegura una exploración equiprobable de todo el espacio de configuraciones. Por último, es sencillo, rápido y de fácil extensión a escenarios complejos.

3. RAPIDLY EXPLORING RANDOM TREES (RRT)

El RRT original pronto dio lugar a varias versiones más sofisticadas y de diferente comportamiento según el entorno. En esta sección se describirá, en primer lugar, el algoritmo original y, posteriormente, se presentarán distintas modificaciones efectuadas sobre el mismo.

3.1 Algoritmo RRT

El algoritmo RRT original se basa en la construcción de un árbol de configuraciones que crece buscando a partir de un punto origen. Para entender el algoritmo se usarán los siguientes conceptos:

- C es el conjunto de todas las configuraciones posibles del robot en un espacio dado.
- C_{free} es el subconjunto de C de las configuraciones que no intersectan ninguno de los obstáculos existentes en dicho espacio.
- R métrica definida dentro de C. Puede ser distancia euclídea u otra ponderación de proximidad que pueda interesar.
- q_{ini} es la configuración inicial (en el caso de un robot que se mueve en un plano, las coordenadas x e y de un punto de referencia y la orientación del vehículo respecto a uno de los ejes del sistema de referencia).
- q_{fin} es la configuración que se desea alcanzar.
- q_{rand} es una configuración aleatoria que genera el algoritmo.
- q_{near} es la configuración más próxima a q_{rand}, en el sentido definido por R, de entre las existentes en un árbol.
- q_{new} es la configuración que se va a añadir al árbol.
- ϵ es la longitud del segmento de crecimiento. En realidad, es la distancia entre un punto del árbol y el siguiente con el que está conectado.

El objetivo original del método RRT consistía en construir un árbol de exploración que cubriera uniformemente todo el espacio libre de colisión. Para ello, se desarrolló el algoritmo que se muestra en la figura 1 (LaValle, 1998). Dicho algoritmo tiene como misión seleccionar un punto (q_{rand}) de forma aleatoria y extender hacia él el árbol de configuraciones. Obsérvese cómo se hace uso de la función `Extiende`. Dicha función tiene el cometido de ampliar el árbol en el sentido que marca q_{rand} determinando si existe un camino libre de colisión. El esquema de dicha función se presenta en la figura 2.

```
RRT (qini)
{
  Arbol[0] = qini;
  Para k = 1 hasta Kmax
  {
    qrand = ConfiguraciónAleatoria();
    Extiende(Arbol, qrand);
  }
  Devuelve Arbol;
}
```

Figura 1. Algoritmo RRT

El algoritmo comienza inicializando la tabla asociada al árbol con la configuración origen. Seguidamente, se entra en un bucle, limitado por un valor `Kmax`, cuya función es finalizar el algoritmo una vez se ha realizado un número prefijado de iteraciones. Este valor, se utilizará posteriormente para parar el algoritmo en el caso en que no se alcance la configuración final. Es importante resaltar que la determinación de dicho valor dependerá de las características del problema (número de obstáculos, tiempo límite del algoritmo, etc...).

```
Función Extiende(Arbol, qrand)
{
  qnear= VecinoMásPróximo(qrand, Arbol);
  Si ( NuevaConfiguración(qrand, qnear, qnew) )
  {
    AñadeVértice(Arbol, qnew);
    Si ( qnew=qrand )
    {
      Devuelve alcanzado;
    }
    Si no
    {
      Devuelve avanzado;
    }
  }
  Si no
  {
    Devuelve rechazado;
  }
```

Figura 2. Función `Extiende`

Dentro del bucle del algoritmo RRT hay dos instrucciones. Con la primera se obtiene un punto al azar dentro del espacio libre de colisión (C_{free}); la segunda hace crecer el árbol en dirección a la configuración aleatoria anteriormente obtenida.

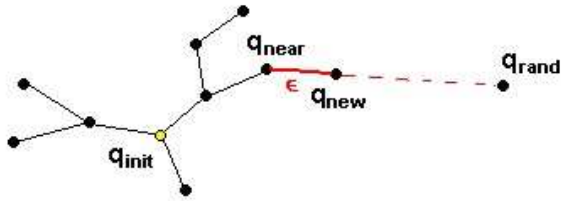


Figura 3. Crecimiento del árbol

El crecimiento del árbol se consigue con la función `Extiende`. La estructura de dicha función comienza con el cálculo de q_{near}. Esto se realiza gracias a la función `VecinoMásPróximo` que aplica la métrica R definida anteriormente a todos los vértices del árbol, obteniendo el punto más cercano a q_{rand}. Seguidamente, la función `NuevaConfiguración` calcula q_{new}, el nuevo punto a agregar, mediante un salto de tamaño ϵ partiendo de q_{near} en dirección hacia q_{rand} (ver figura 3). Para la obtención de q_{new} se tiene en cuenta si hay alguna colisión en dicho desplazamiento, devolviendo "verdadero" o "falso" según sea un movimiento posible o, por el contrario, colisione con algún obstáculo.

Si no se ha detectado colisión, se agrega el nuevo punto al árbol distinguiendo entre dos casos. Si el punto aleatorio se encuentra dentro de un círculo de centro q_{near} y radio ϵ, entonces se considera que se ha alcanzado q_{rand} y, por tanto, dicho punto ha sido *alcanzado*. El algoritmo notificará entonces tal circunstancia. Si por el contrario (caso más

usual), no se ha producido el alcance, entonces devolverá el valor *avanzado*. Por último, en caso de que la función `NuevaConfiguración` haya advertido de la existencia de algún obstáculo en el camino de q_{near} a q_{new}, se informa de que no ha habido nuevas ramas y el punto no es agregado al árbol.

El comportamiento de este algoritmo con respecto a otros es mejor en cuanto a la homogeneidad del espacio explorado.

La naturaleza del RRT le hace avanzar con más avidez en aquellas zonas donde haya un mayor espacio libre, pues es allí donde hay más posibilidad de encontrar puntos q_{rand} factibles. Esto puede comprenderse observando el crecimiento del árbol en distintos entornos. Obsérvese, por ejemplo, la progresión del algoritmo representado a través de las figuras 4, 5, 6 y 7. En este caso, el espacio libre consiste en un círculo. Esto significa, que desde el punto central, no hay ninguna preferencia en cuanto a la dirección de crecimiento. Efectivamente, puede verse que las ramas iniciales del árbol surgen en direcciones aleatorias y crecen sin predominio entre ellas. Otro hecho a observar es la inexistencia de una alta densidad de ramas en el origen. La naturaleza del RRT evita precisamente esto. Es más, se puede contemplar que el árbol tiende a lograr una densidad uniforme de ramas sobre el espacio libre.

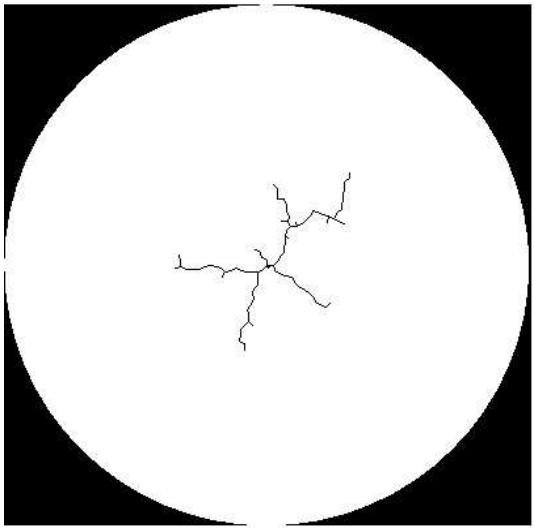


Figura 4. RRT tras 100 iteraciones.

Obsérvese ahora la figura 8. En esta ocasión se representa la progresión del algoritmo RRT sobre un espacio diferente. Se observa que, desde el punto origen, situado dentro del rectángulo pequeño, parte una rama que crece con acusada preponderancia con respecto a las demás. La causa es la alta frecuencia con que aparecen puntos aleatorios sobre el rectángulo de la derecha debido a que que presenta mayor área libre de obstáculos.

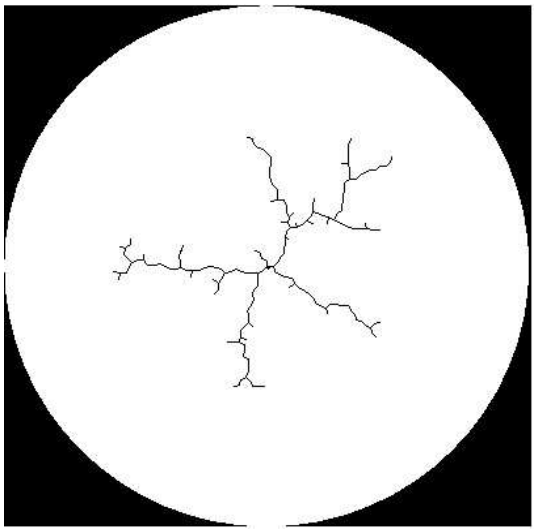


Figura 5. RRT tras 300 iteraciones.

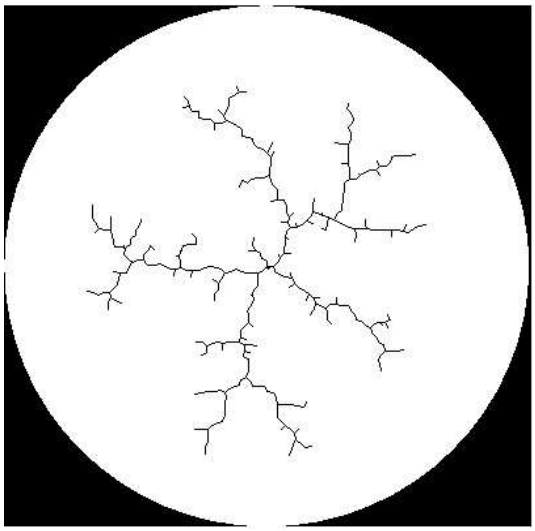


Figura 6. RRT tras 400 iteraciones.

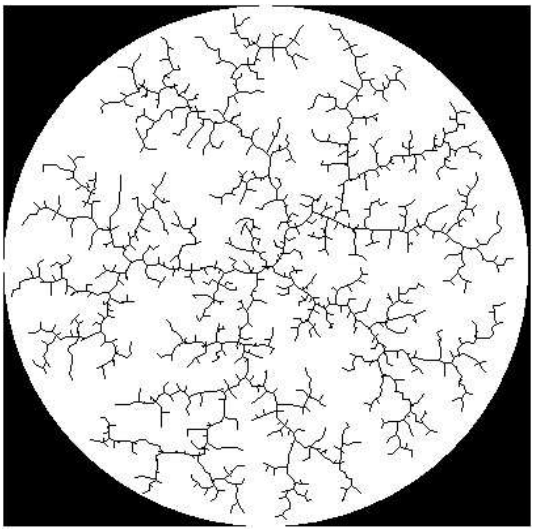


Figura 7. RRT tras 800 iteraciones.

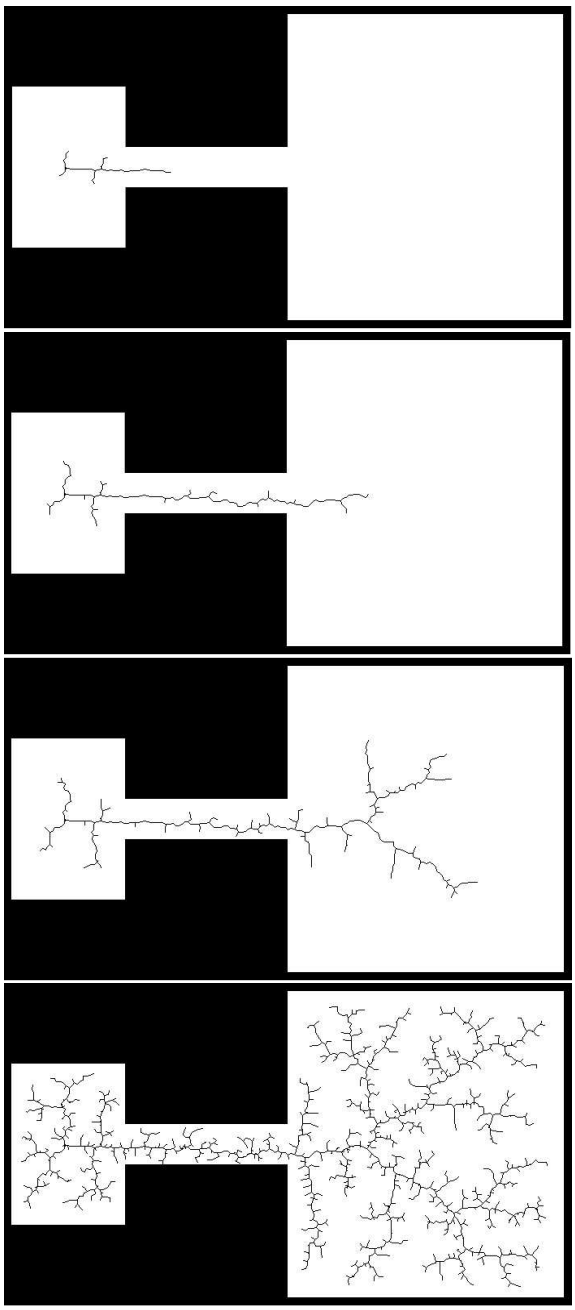


Figura 8. Evolución en entorno asimétrico.

Otro detalle relevante consiste en el crecimiento de ramas principales hacia las esquinas de los rectángulos. Si se compara con el círculo, un cuadrado presenta más puntos desde su centro a cualquiera de sus vértices que en otras direcciones. Esto significa, igualmente, una mayor densidad de puntos aleatorios en dichas direcciones y, por tanto, una predominancia de crecimiento.

Al igual que en el caso anterior, tras las suficientes iteraciones, puede observarse que la densidad de ramas resulta razonablemente homogénea a pesar del desequilibrio inicial existente entre los dos rectángulos.

Por último, se advierte que este algoritmo sólo se ocupa de generar un árbol capaz de explorar de modo equiprobable el espacio libre. Por tanto, no es el método más adecuado para hallar el camino entre dos puntos. En las siguientes secciones se detallan algunas extensiones y mejoras del RRT que permiten establecer un camino entre configuración origen y destino.

3.2 Algoritmo RRT-Bidireccional básico

La adaptación del algoritmo RRT para conseguir conectar una configuración inicial con una final se logra sustituyendo el algoritmo original RRT por el algoritmo *RRT-Bidireccional básico* mostrado en la figura 9.

```
RRT-Bidireccional básico(qini, qfin)
{
  ArbolA[0]= qini;
  ArbolB[0]= qfin;
  Para k = 1 hasta Kmax
  {
    qrand= ConfiguraciónAleatoria( );
    Si (Extiende(ArbolA, qrand)≠rechazado)
    {
      Si (Extiende(ArbolB, qrand)=alcanzado Y
          Extiende(ArbolA, qrand)=alcanzado)
      {
        Devuelve Camino(ArbolA, ArbolB);
      }
    }
    Intercambiar(ArbolA, ArbolB)
  }
  Devuelve Error
}
```

Figura 9. Algoritmo RRT-Bidireccional básico

Este algoritmo se basa en la construcción de dos árboles que parten de los puntos origen y destino simultáneamente. El algoritmo concluye en cuanto dichos árboles conectan. Es decir, cuando la función `Extiende` devuelve *alcanzado* para el segundo árbol. Si alcanzado el valor de `Kmax` ambos árboles no han coincidido, se devuelve un mensaje de error.

Es importante destacar que ambos árboles comparten el mismo punto q_{rand} para crecer. Esto se hace para ahorrar coste computacional sin menoscabar las propiedades de los RRT. Debido a ello, sendos árboles tenderán a explorar el espacio vacío, aumentando con el tiempo la probabilidad de conexión.

Obsérvese el uso de la función `Intercambiar`, que intercambia el orden de los árboles de manera que el crecimiento de ambos sea equilibrado. Si esta función no existiera, el `ArbolB` sólo crecería cuando la extensión del `ArbolA` no presentara colisión.

Este algoritmo admite ciertas mejoras. Por una parte, presenta una condición muy restrictiva ya que exige que en una misma iteración coincidan ambos árboles en un mismo punto. Para relajar esta restricción se ha desarrollado el algoritmo *RRT-Ext-Con* (Kuffner y LaValle, 2000). Por otra, el

crecimiento de los árboles puede ser dirigido de forma que tiendan el uno hacia el otro, en lugar de crecer por zonas donde la interconexión es difícil. Para conseguir esto se ha desarrollado el algoritmo *RRT-Ext-Ext* (LaValle y Kuffner, 2001). Seguidamente se presentan ambos métodos.

3.3 RRT-Ext-Ext

Una mejora que permite agilizar la conexión entre los árboles en el anterior algoritmo se logra en el *RRT-Ext-Ext*. Con una muy pequeña variación, es posible atribuirle una capacidad para que cada árbol pueda dirigir su crecimiento hacia su homólogo. Esto lo hace más competitivo frente a otros algoritmos puramente aleatorios.

```
RRT-Ext-Ext (qini, qfin)
{
  ArbolA[0]= qini;
  ArbolB[0]= qfin;
  Para k = 1 hasta Kmax
  {
    qrand= ConfiguraciónAleatoria( );
    Si (Extiende(ArbolA, qrand)≠rechazado)
    {
      Si (Extiende(ArbolB, qnew)=alcanzado)
      {
        Devuelve Camino(ArbolA, ArbolB);
      }
  }
    Intercambiar(ArbolA,ArbolB);
  }
  Devuelve Error;
}
```

Figura 10. Algoritmo RRT-Ext-Ext.

Este sutil cambio con respecto al algoritmo anterior (figura 9) es posible observarlo en la figura 10, donde el punto q_{rand} se sustituye en el segundo árbol por q_{new}, es decir, el punto recién agregado al árbol precedente. Finalizada la iteración, la función `Intercambiar` se ocupa de que la próxima vez, sea el último árbol el que crezca hacia q_{rand} y el primero hacia q_{new}. Así pues, las acciones resultantes de dos iteraciones consecutivas serían:

1. `ArbolA` crece hacia q_{rand} (1^a iteración).
2. `ArbolB` crece hacia q_{new} de `ArbolA`.
3. `ArbolB` crece hacia q_{rand} (2^a iteración).
4. `ArbolA` crece hacia q_{new} de `ArbolB`.

De esta forma cada árbol invierte la mitad de su tiempo en explorar el espacio libre, y la otra mitad, en buscar a su compañero.

3.4 Algoritmo RRT-Ext-Con

En este algoritmo (figura 11) se sustituye la función `Extiende`, que agregaba un nuevo segmento al árbol, por otra denominada `Conecta` (ver figura 12), más ambiciosa, que va agregando segmentos consecutivos hasta alcanzar la configuración objetivo o un obstáculo (Kuffner y LaValle, 2000).

```
RRT-Ext-Con (qini, qfin)
{
  ArbolA[0]= qini;
  ArbolB[0]= qfin;
  Para k = 1 hasta Kmax
  {
    qrand= ConfiguraciónAleatoria( );
    Si (Extiende(ArbolA, qrand)≠rechazado)
    {
      Si (Conecta(ArbolB, qnew)=alcanzado)
      {
        Devuelve Camino(ArbolA, ArbolB);
      }
    }
    Intercambiar(ArbolA,ArbolB);
  }
  Devuelve Error;
}
```

Figura 11. Algoritmo RRT-Ext-Con.

```
Función Conecta(Arbol, q)
{
  Repetir
  {
    S=Extiende(Arbol, q);
  } Mientras (S=avanzado);
  Devuelve S;
}
```

Figura 12. Función `Conecta`

Se atenúa así el problema de la excesiva restrictividad del algoritmo *Bidireccional básico*. Ahora, los segmentos agregados pueden ser múltiples, con lo que para conectar basta con que exista sólo un camino rectilíneo libre entre el punto q_{new} y el vértice más próximo del otro árbol.

La única desventaja consiste en el mayor coste computacional de la función `Conecta`, que se compensa en muchos escenarios con la mayor eficiencia del algoritmo.

Este algoritmo presenta una gran eficacia cuando se trata de planificar trayectorias para sistemas holónomos. Por el contrario, como se ilustra en (Cheng y LaValle, 2001), el algoritmo RRT-Ext-Ext representa la mejor opción cuando se trata de planificar trayectorias para sistemas no holónomos.

4. POSTPROCESADO

Los árboles obtenidos en la aplicación de los distintos algoritmos RRT pueden resultar complejos para ser recorridos. Normalmente, los árboles son susceptibles de simplificación (por ejemplo, en algunos casos los extremos de ambos árboles podrían unirse con una simple línea recta).

Por consiguiente, en las aplicaciones prácticas del método RRT resulta conveniente aplicar a los árboles obtenidos un postprocesado que permita reducir su irregular topología. En este sentido, al igual que ocurre en otras técnicas de planificación, se suelen aplicar algoritmos de simplificación, muy rápidos y sencillos, que parten de la información

suministrada por el planificador y generan, de forma iterativa, un camino más simple. Dichos algoritmos suelen tener un coste computacional bajo.

Nótese, que el algoritmo de postprocesado deberá ser diseñado en función de la aplicación concreta, constituyendo un elemento más del sistema de planificación.

Así, por ejemplo, para el caso de planificación de una trayectoria del objeto de planta rectangular mostrado en la figura 14, se ha utilizado la función que se muestra en la figura 13, inspirada en un algoritmo de postprocesado introducido en (Laumont *et al.*, 1994).

```
Función Postprocesado(CaminoOriginal)
{
  fin = 0;
  Mientras ( no fin )
  {
    colisión = 1;
    i=NúmeroDeVértices;
    Mientras ( colisión )
    {
      Si ( CuerdaVálida(1, i) )
      {
        EliminaArco(1,i,CaminoOriginal,CaminoFinal);
        colisión=0;
      }
      Si no
      {
        i=i/2;
      }
      Si ( i≤ 2 )
      {
        EliminaArco(1,2,CaminoOriginal,CaminoFinal);
        colisión=0;
      }
    }
    Si ( NúmeroDeVértices≤ 2 )
    {
      fin=1;
    }
  }
  Devuelve(CaminoFinal);
}
```

Figura 13. Función `Postprocesado`.

Este algoritmo permite obtener un camino final simplificado a partir del árbol original suministrado por el RRT. La idea del algoritmo consiste en eliminar aquellos tramos del árbol que puedan ser sustituidos por una trayectoria recta libre de colisión.

Como se puede ver, `NúmeroDeVértices` es una función que devuelve el número de vértices que componen el camino original e `i` representa el número de vértices que se pretende sustituir. En un principio, el algoritmo prueba a sustituir todo el camino original por una trayectoria recta. Si esto no es posible, `i` va adquiriendo un valor cada vez menor (típicamente se escoge el punto medio del tramo de camino con el que se prueba).

En cada iteración, la función `CuerdaVálida(1, j)` establece una ruta recta entre el vértice `1` y el `j`, y genera un conjunto de puntos alternativos sobre los que se comprueba la posibilidad de colisión. Si la ruta presenta colisiones, se intenta establecer una nueva ruta recta, pero ahora entre el primer vértice y el punto medio del trozo de camino con el que se acaba de probar.

Si en una de las iteraciones la ruta es válida, es decir es libre de colisiones, entonces se da paso a la función `EliminaArco`, que sustrae del camino original los vértices que van desde el `1` al `j`, disminuyendo, por tanto, el valor que devolverá `NúmeroDeVértices` e incorporando la nueva ruta al camino final. Por tanto, el camino original queda configurado con un nuevo vértice inicial y el camino final con un nuevo tramo recto.

En la figura 14a se presenta un escenario donde aún no se ha aplicado el postprocesado. Se muestran los árboles generados por el RRT así como, las distintas posiciones y orientaciones del robot móvil a lo largo del camino solución. Una vez obtenido dicho camino se procede a su postprocesado. En la figura 14b se muestra el camino sin los árboles de generación. La figura 14c ilustra el postprocesado aplicado a la primera mitad del camino, mientras que la figura 14d muestra el camino final resultado del proceso de postprocesado.

5. APLICACIÓN DE ALGORITMO RRT EN SISTEMAS NO HOLÓNOMOS

5.1 Sistemas no holónomos

Todos los vehículos rodados presentan restricciones no holónomas. Considérese, por ejemplo, un vehículo de planta rectangular y conducción diferencial cuya configuración puede determinarse mediante las coordenadas de un punto de referencia (se suele escoger el centro del eje de las ruedas, ver figura 15) y el valor de θ, la orientación del eje longitudinal. El movimiento del vehículo puede modelarse mediante la ecuación (Ollero, 2001):

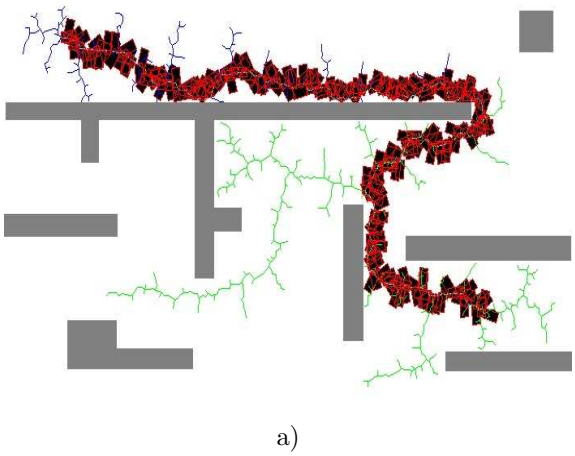
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{r\cos\theta}{2} \\ \frac{r\sin\theta}{2} & \frac{r\sin\theta}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\alpha}_1 \\ \dot{\alpha}_2 \end{bmatrix}. \qquad (1)$$

En estos vehículos, ver figura 15b, la dirección del movimiento está controlada por las velocidades de rodado de cada una de las ruedas $(\dot{\alpha}_1, \dot{\alpha}_2)$.

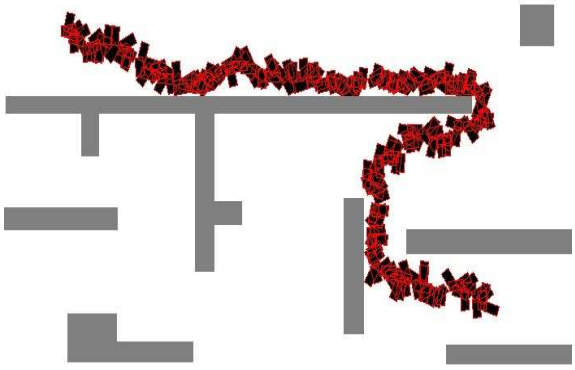
Este vehículo presenta la siguiente restricción cinemática (Ollero, 2001):

$$\dot{x}_0 \sin\theta - \dot{y}_0 \cos\theta = 0. \qquad (2)$$

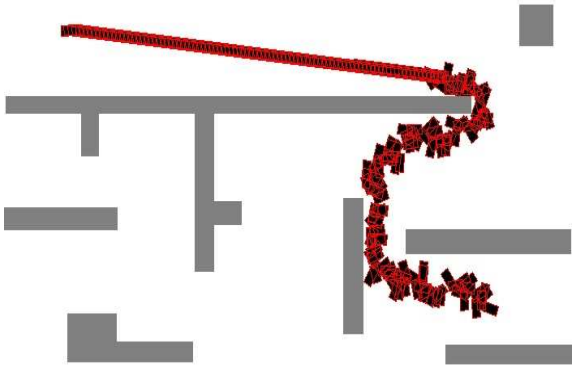
Esta ecuación implica la imposibilidad de mover el vehículo en una dirección distinta a la determi-



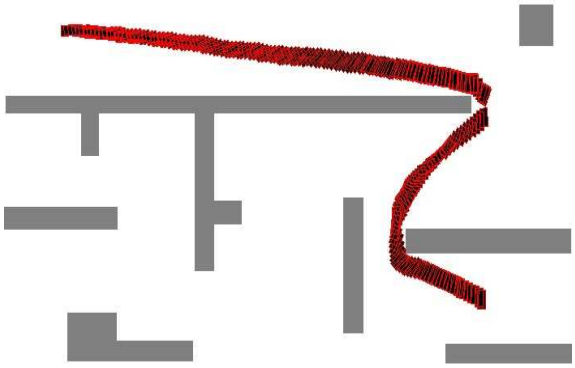
a)



b)

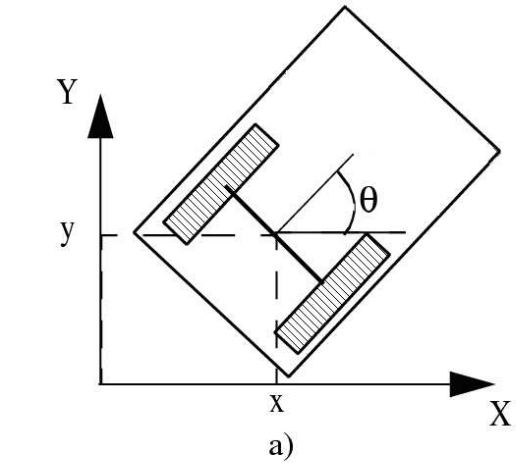


c)

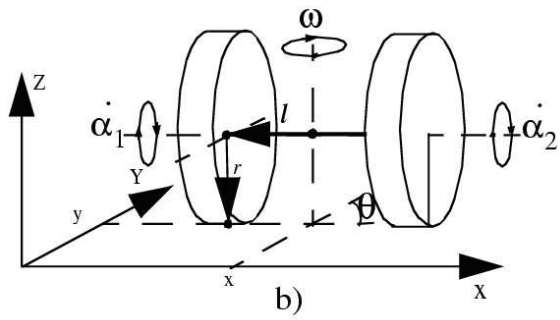


d)

Figura 14. Planificación y postprocesado.



a)



b)

Figura 15. Vehículo con conducción diferencial.

nada por la orientación del vehículo; es decir, sólo se puede mover en el sentido del eje longitudinal de las ruedas.

5.2 RRT y sistemas no holónomos

La aplicación del algoritmo RRT a sistemas con restricciones no holónomas presenta algunos problemas. Así, en muchas ocasiones, el camino obtenido viola las restricciones cinemáticas del sistema (el camino resultante es tal que el vehículo ha de moverse en direcciones distintas a la determinada por su orientación, ver figura 14). Además, no es inmediato determinar la secuencia de movimientos necesarios para conseguir que el móvil se desplace según el camino planificado.

Los métodos existentes en la actualidad se basan en una modificación sobre el RRT original (LaValle, 1998). Consisten en considerar un conjunto discreto de órdenes de control. A la hora de extender el árbol, se escogen los vértices más cercanos a q_{rand} y se les aplican todas y cada una de las posibles señales de control. De entre los puntos así generados se escoge el más próximo y se agrega al árbol (LaValle y Kuffner, 1999).

Estos métodos tienen varios inconvenientes (LaValle y Kuffner, 1999). Por una parte, es necesario definir la métrica que permita generar trayectoria factibles, lo cual es un problema de cierta relevancia (LaValle y Kuffner, 1999; Cheng *et al.*, 2001).

En muchos casos es necesario hacer uso de una heurística particular, con la que hay que tener un especial cuidado debido a la alta sensibilidad del algoritmo respecto de este elemento (Cheng y LaValle, 2001; Cheng *et al.*, 2001). Por otra, el coste computacional se eleva: el primero de los métodos necesita aplicar al camino obtenido técnicas de optimización que permitan filtrar la aparición de fluctuaciones inútiles en las señales de control aplicadas (Cheng y LaValle, 2001; Cheng *et al.*, 2001); en el caso del segundo método, se plantea el problema del retardo que supone aplicar todas las órdenes de control.

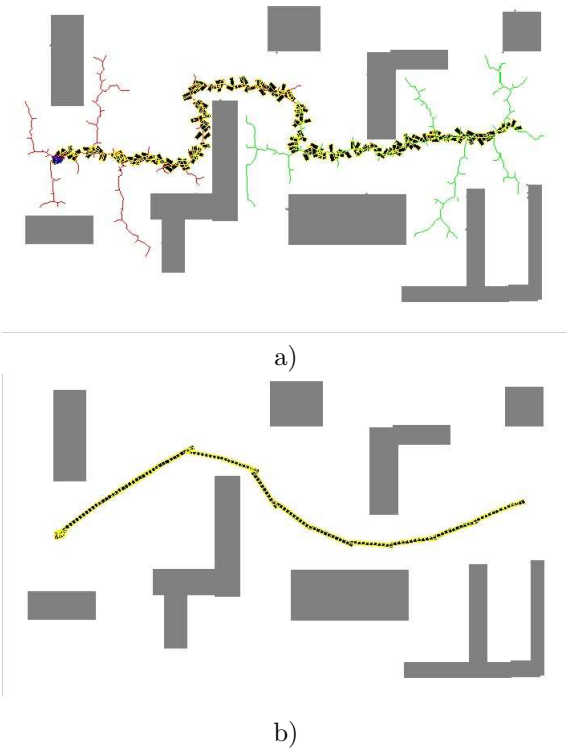
Los mejores resultados obtenidos al aplicar estos métodos se han conseguido considerando órdenes de control que generan líneas rectas y giros a la derecha, y a la izquierda, ambos con curvatura acotada. Si durante el movimiento sólo se utilizan valores de velocidad positiva, se trata de una planificación basada en los resultados de Dubins (1957); si por el contrario se consideran valores de velocidad positivos y negativos se trata de una planificación basada en los resultados de Reeds y Shepp (1990).

5.3 Maniobras restringidas y RRT

La solución propuesta en este artículo se basa en la aplicación del concepto de *maniobra restringida* (Gómez-Bravo *et al.*, 2001). Este tipo de trayectorias representan un conjunto de maniobras que permiten variar una y sólo una de las coordenadas del espacio de configuración. Así, es posible establecer una trayectoria válida desde cualquier punto del árbol hasta el nuevo punto a agregar con sólo obtener la correspondiente combinación de maniobras. Dado que son maniobras predefinidas, es extremadamente sencillo obtener una plantilla del espacio que ocupará el vehículo durante su ejecución. Por tanto, es posible establecer la posibilidad de colisión sin necesidad de computar explícitamente la trayectoria.

Por ejemplo, para el caso del vehículo no holónomo de conducción diferencial, se consideran dos maniobras restringidas. La primera consiste en el cambio de orientación del vehículo, manteniendo la posición cartesiana del punto de referencia. Suponiendo que el vehículo tiene una planta rectangular, la plantilla que define esta maniobra es un círculo centrado en el punto de referencia y de radio igual a la distancia de los vértices a dicho punto. La segunda maniobra consiste en desplazar longitudinalmente el vehículo, variando en este caso la coordenada x del sistema de referencia de igual orientación a la del robot. La plantilla asociada a esta maniobra es un rectángulo, de longitud igual al desplazamiento efectuado y de anchura idéntica a la de la planta del vehículo.

Cualquier movimiento genérico del vehículo estará compuesto de un giro para orientarse a la posición de destino, un desplazamiento hasta el punto destino y una nueva reorientación para alcanzar la orientación final deseada.



a)

b)

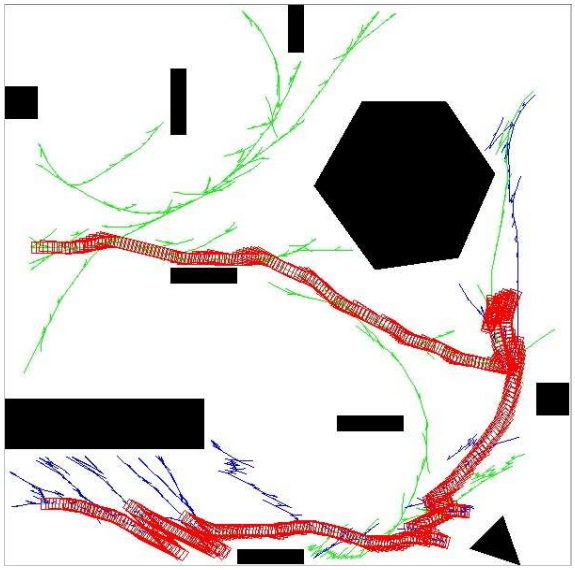
Figura 16. RRT aplicado a un robot de conducción diferencial.

En este artículo, en lugar de utilizar el algoritmo RRT-Ext-Ext con un conjunto discreto de órdenes de control, se propone emplear el algoritmo RRT-Ext-Ext original junto con un postprocesado basado en el concepto de maniobra restringida. Así, el RRT sigue manteniendo su velocidad de computación, y es la progresiva simplificación de la trayectoria la que depara que el camino no viole las restricciones no holónomas.

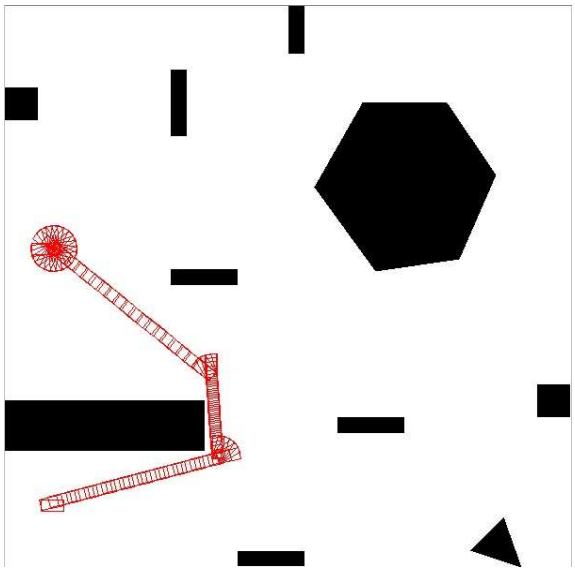
Así, partiendo del algoritmo de postprocesado propuesto en la Sección 4, la idea consiste en comprobar, en las sucesivas iteraciones, la posibilidad de conexión entre vértices del árbol mediante alguna maniobra restringida. De este modo, la función `CuerdaVálida`, en lugar de establecer una línea recta entre vértices, comprueba que la plantilla asociada a la maniobra no presenta colisiones.

Por tanto, el camino completo se segmenta al descomponerlo en una secuencia de movimientos acordes con las restricciones no holónomas. En la figura 16 se ilustran los distintos pasos seguidos en la obtención de un camino libre de colisiones para un vehículo de conducción diferencial. En la figura 16a se presentan los árboles y el camino obtenido al aplicar el algoritmo RRT-Ext-Ext. Por su parte, en la figura 16b, se muestra el resultado de aplicar el suavizado mediante las maniobras descritas.

Para evaluar la técnica propuesta con los métodos existentes se han realizado diversas comparaciones, incluyendo distintos tipos de vehículos. Así, la figura 17 presenta una comparación entre una trayectoria para un vehículo de conducción diferencial generada utilizando el algoritmo RRT con un conjunto discreto de consignas de control (ver figura 17a) y una trayectoria generada mediante el método propuesto (ver figura 17b). El entorno considerado es el presentado en LaValle y Kuffner (2001), y se ha elegido para facilitar la comparación.



a)



b)

Figura 17. Comparación de aplicación a un robot con conducción diferencial. a) RRT con conjunto discreto de consignas de control. b) RRT basado en maniobras restringidas.

El conjunto discreto de consignas de control considerado en la figura 17a, está compuesta de catorce posibilidades, incluyendo, por ejemplo, marcha hacia adelante, marcha hacia atrás y giros de diversa amplitud. Tras realizar una serie de diez experimentos se obtuvieron los datos mostrados en la tabla 1. Los algoritmos están implementados en JAVA y las simulaciones se realizaron con un procesador AMD $K7-1700$ a 1,4MHz.

Tabla 1. Comparación aplicación RRT a vehículo diferencial.

	Tiempo medio cálculo (ms)	**Desv. típica tiempo (ms)**	**Long. media camino (m)**	**Desv. típica long. (m)**
RRT consignas discretas	7620,0	3563,8	179,7	34,1
RRT maniobras retringidas	652,7	93,8	51,9	4,7

Asimismo, la figura 18 muestra una comparación entre una trayectoria para un vehículo con configuración Ackerman, similar a un coche (Ollero, 2001), con radio de curvatura mínimo de 2,5 metros, utilizando el algoritmo RRT con un conjunto discreto de consignas de control (ver figura 18a) y una trayectoria generada mediante el método propuesto (ver figura 18b).

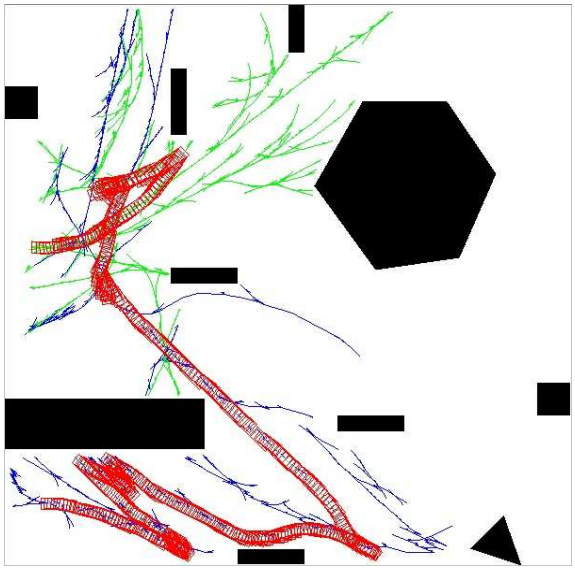
En este caso las maniobras restringidas son las descritas en (Gómez-Bravo *et al.*, 2001). Por su parte, el conjunto de consignas de control estaba compuesto también por catorce posibilidades, incluyendo giros de curvatura acotada. Tras realizar una serie de diez experimentos se obtuvieron los datos mostrados en la tabla 2.

Tabla 2. Comparación aplicación RRT a vehículo con configuración Ackerman.

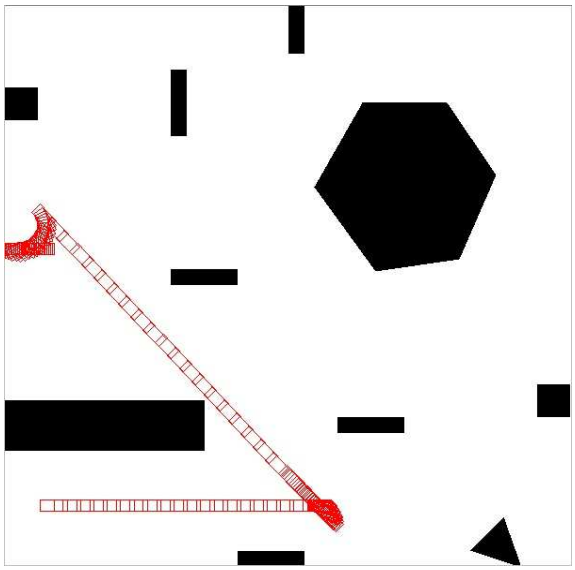
	Tiempo medio cálculo (ms)	**Desv. típica tiempo (ms)**	**Long. media camino (m)**	**Desv. típica long. (m)**
RRT consignas discretas	11408,2	7146,7	173,2	31,0
RRT maniobras restringidas	701,8	73,5	84,7	3,4

Como muestran los datos, el método propuesto requiere menor tiempo de cálculo y genera caminos más cortos. De este modo, el elevado coste computacional requerido por el algoritmo RRT al considerar un conjunto discreto de órdenes, se compensa con la utilización del algoritmo RRT-Ext-Ext original (mucho más rápido) y un post-procesado, en el que el ahorro de tiempo se debe, fundamentalmente, a que para la detección de colisiones se utiliza la plantilla asociada a cada maniobra en lugar de los puntos de la trayectoria.

Finalmente, para ilustar la efectividad del método, en la figura 19 se presenta un experimento con un vehículo Ackerman, cuyo radio mínimo



a)



b)

Figura 18. Comparación de aplicación a un robot con configuración Ackerman. a) RRT con conjunto discreto de consignas de control. b) RRT basado en maniobras restringidas.

de curvatura es de 6 metros, en un entorno más complejo dada su maniobrabilidad.

6. CONCLUSIONES

Los algoritmos puramente aleatorios de planificación de caminos (tales como RRT y sus extensiones) constituyen una alternativa a los métodos de planificación convencionales para determinados entornos complejos y robots con restricciones. Los requisitos computacionales hacen viable su implementación en tiempo real, incluso para una posible replanificación en línea ante obstáculos no modelados. En este sentido la inclusión de las maniobras restringidas como herramienta en los

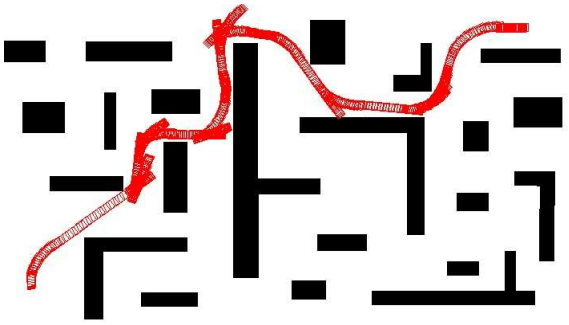


Figura 19. Aplicación de RRT-Ext-Ext con maniobras restringidas a un robot con configuración Ackerman.

algoritmos existentes puede suponer una mejora interesante.

7. AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el proyecto DPI2005-02293.

REFERENCIAS

Bruce, J., and M. Veloso (2002). Real-Time Randomized Path Planning for Robot Navigation, *Proc. IROS 2002*, Switzerland.

Cheng, P., and S. M. LaValle (2001). Reducing metric sensitivity in randomized trajectory design. In: *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pp. 43–48.

Cheng, P., Z. Shien and S. M. LaValle (2001). RRT-Based Trajectory Design for Autonomous Automobiles and Spacecraft.

Dubins, L.E. (1957). On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal position and tangents. *Amer. J. Math.*, Vol 79, pp. 497–516.

Gómez Bravo F., F. Cuesta and A. Ollero. (2001). Parallel and diagonal parking in nonholonomic autonomous vehicles. *Engineering Aplication of Artificial Inteligence*. Vol 14 No. 1, pp. 419–434.

Kuffner, J.J., and S. M. LaValle (2000). RRT-connect: An efficient approach to single-query path planning. In: *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 995–1001.

Latombe, J.C. (1991). *Robot Motion Planning*, Kluwer Academic Pulisher.

Laumont, J.P., P.E. Jacobs, M. Taix and M. Murray (1994). A Motion Planner for Nonholonomic Mobile Robots. *IEEE Trans. on Robotics and Autom*, Vol 10, No 5: pp. 577–593.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. *Computer Science Dept., Iowa State University* TR 98-11.

LaValle, S. M., and J. J. Kuffner (1999). Randomized kinodynamic planning. In: *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 473–479.

LaValle, S.M., and J.J. Kuffner (2001). Rapidly-Exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*. B. R. Donald, K. M. Lynch, and D. Rus, editors, , pp. 293–308.

Muñoz, V. (1995). *Planificación de Trayectorias para Robots Móviles*. Tesis Doctoral. Universidad de Málaga.

Ollero, A. (2001). *Robótica: manipuladores y Robots Móviles*. Marcombo Boixareu.

Reeds, J.A. and R.A. Shepp, (1990). Optimal paths for a car that goes both forward and backward. *Pacific J. Math.*, Vol 145 n 2, pp. 367–393.