Universidad de Huelva

Departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática

Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos

Memoria para optar al grado de doctor
presentada por:

Diego Antonio López García

Fecha de lectura: 21 de julio de 2011

Bajo la dirección del doctor:

Fernando Gómez Bravo

Huelva, 2012

ISBN: 978-84-15147-78-7
D.L.: H 53- 2012

Universidad
de Huelva

Tesis Doctoral

Título:

Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos.

Autor:

D. Diego A. López García

Dirigido por:

Dr. D. Fernando Gómez Bravo

Huelva, 8 de Junio del 2011

A mis padres

Agradecimientos

Deseo expresar mi agradecimiento en primer lugar a Fernando Gómez Bravo, por su implicación más allá de lo esperado en un director de tesis, su acertada orientación y su ingente trabajo.

A Aníbal Ollero y Federico Cuesta, por la atención recibida, la cooperación y la eficaz gestión de recursos y contingencias.

A Francisco Real, cuya generosidad en el esfuerzo y disposición han excedido cualquier expectativa.

A José Manuel Matamoros, Luís Merino, Roberto Conde, Pablo Soriano y al resto de compañeros cuya colaboración ha sido indispensable para el desarrollo de los experimentos con el robot ROMEO-4R.

A José Manuel Martín-Ramos, por sus apreciadas aportaciones y su estímulo.

A Ismael López y Antonio Pérez-Aranda, por la búsqueda de la calidad en el trabajo.

Al IPFN (Instituto de Plasmas e Fusão Nuclear), y en especial a Alberto Vale, por su colaboración aportando los datos respecto al proyecto ITER y su actitud abierta, animando el diálogo científico.

A Estefanía Cortés, Tomás Mateo, Manuel Vasallo y resto de compañeros de departamento, por su flexibilidad y comprensión en la asunción de las tareas docentes, y el buen clima gracias al cual ha sido más llevadera la elaboración de esta tesis.

Y por último y en especial a mi familia, por tolerar tantas horas robadas.

Índice de contenidos

CAPÍTULO 1 Introducción

1.1 Introducción

La robótica ha sido una de las disciplinas que mayor desarrollo ha experimentado en los últimos tiempos. Hoy día es frecuente hallar robots operando en diversos ámbitos de aplicación: producción en cadena, exploración de entornos inaccesibles para el ser humano (profundidades submarinas, misiones espaciales, etc.), manipulación de materiales peligrosos (explosivos, ambientes tóxicos, etc.), trabajos de elevada precisión, etc. Sin embargo, muchos de ellos necesitan un operador humano para dirigir la tarea. El esfuerzo científico de los últimos años intenta proporcionar autonomía a los robots. En el caso de la robótica móvil, los robots han de desplazarse en su entorno. Uno de los retos consiste en caracterizar dicho entorno a través de sensores, identificar obstáculos y zonas de paso, e incluso ubicarse con la mayor precisión posible con respecto a un sistema de referencia dado. Una vez hecho esto, el robot debe trazar una trayectoria hacia su destino, y después tratar de seguirla lo más fielmente posible. A veces, no se dispone de ningún mapa y el robot ha de maniobrar de acuerdo a lo que detecta a través

de sus sensores. Este tipo de técnicas reciben el nombre de “navegación reactiva” y una de las formas de implementarla es generando trayectorias locales destinadas a evitar los obstáculos que el robot va reconociendo. En cualquier caso, el cálculo de trayectorias en un entorno dado es un aspecto de gran relevancia en el ámbito de la robótica.

El término entorno hace referencia al espacio físico en el que se hallan el robot y los lugares que debe alcanzar. Por ejemplo, un vehículo autónomo destinado al transporte de mercancías tendrá como entorno la nave industrial en la que realiza sus tareas, uno de exploración aérea la zona que deba vigilar, etc. En el caso de los robots articulados también es necesario el cálculo de trayectorias, siendo éstas referidas a todas y cada una de las partes móviles del robot.

Si no hay obstáculos, el robot podrá ocupar cualquier posición dentro de este espacio. Si los hay, habrá lugares prohibidos (donde el robot coincida con el obstáculo) y otros permitidos. Estos últimos, en conjunto, definen el espacio de configuraciones libres de colisión (Latombe, 1991). Por ejemplo, en un robot de cadena de montaje cuyo entorno es la esfera de alcance de su pinza, son obstáculos la cinta transportadora, los productos que desplaza y demás elementos que estén dentro de su radio de acción. El espacio de configuraciones libres de colisión será en este caso el conjunto de configuraciones que puede adoptar el robot sin tocar ninguno de los obstáculos anteriormente descritos. Como se definirá más adelante el concepto de configuración está ligado a conocer la posición y orientación de los elementos del robot, que en este caso serían el brazo y la pinza, pero en un robot móvil habitualmente corresponde a su posición y orientación en el plano.

De forma genérica, el problema de la planificación de movimientos en robots móviles consiste en determinar una trayectoria admisible o conjunto de movimientos que permiten llevarlos desde una configuración inicial hasta otra final dentro del espacio de configuraciones libres de colisión. Por admisible se entiende que la trayectoria cumple con las restricciones cinemáticas de los vehículos rodados.

Este problema ha sido ampliamente abordado en la literatura (Fortune y Wilfong, 1988; Latombe, 1991; Laumont y otros, 1994; Muñoz, 1995; LaValle, 1998, Bruce y Veloso, 2002; Garrido y otros, 2007b), describiendo múltiples formas de solventarlo. Entre éstas se hallan los denominados métodos “roadmap”, caracterizados por construir una descripción del espacio libre con la forma de una red de curvas unidimensionales.

Dos de los más representativos de esta especie son los grafos de visibilidad (Nilsson, 1969) y los diagramas de Voronoi (Aurenhammer, 1991). Clasificación aparte merecen los algoritmos que subdividen el espacio libre en celdas, como la descomposición vertical (Chazelle, 1987) o las técnicas quadtree y octree (Ahuja y otros, 1980). Otros procedimientos son los basados en campos de potencial (Khatib, 1980), y por último, aquellos que utilizan algún tipo de función probabilística en su desarrollo (LaValle, 1998).

Los métodos “roadmap” construyen un grafo que representa en sus nodos algunas configuraciones factibles del robot, y en sus arcos la posibilidad constatada de conexión entre dos nodos. Una vez construido este grafo, hallar un camino entre el origen y el destino se reduce a conectar dichos puntos con el grafo y hallar una secuencia de nodos dentro de él (Latombe, 1991).

Dentro de los métodos “roadmap” se encuentra el método de los grafos de visibilidad, el cual genera el grafo identificando sus nodos con los vértices de los polígonos que definen los obstáculos a los que luego agrega los puntos de partida y meta. Recorriendo este grafo el robot puede alcanzar cualquier vértice. Así pues, la solución al problema consiste en escoger una sucesión de vértices conexos dentro del grafo en la que el primero sea el punto de partida y el último la meta (Overmars y Welzl, 1988; Angelier, 2002).

Otro de los métodos “roadmap” es el del diagrama de Voronoi, que construye el grafo de conectividad maximizando la distancia entre el robot y los obstáculos. Un diagrama de Voronoi representa los puntos que equidistan de los límites del espacio de configuraciones libres de colisión. Este método se utiliza bien directamente (Aurenhammer, 1991) o bien como base a otros métodos (Wilmarth y otros, 1998).

Los métodos de descomposición en celdas tratan de dividir el espacio de configuraciones libres de colisión en un conjunto de celdas (Latombe, 1991). Éstas han de tener la peculiaridad de que resulte sencilla la conexión de dos puntos cualesquiera de su interior. Una vez dividido el espacio de configuraciones libres de colisión en un conjunto de celdas completo, se procede a resolver su conectividad a un nivel superior, por ejemplo construyendo un grafo de adyacencia. Es decir, a cada celda se le asigna un nodo, y éstos se unirán o no en función de que las celdas que representen sean adyacentes.

Dentro de los métodos de descomposición en celdas los hay que utilizan celdas cuadradas como los quadtree (Hayward, 1986), celdas trapezoidales como en el método de descomposición vertical (Chazelle, 1987), estructuras en rejilla homogénea (Sethian, 1996), etc.

Las técnicas de campos de potencial se basan en el diseño de unas funciones que pueden ser interpretadas como campos de potencial (Kim y Khosla, 1991; Rimon y Koditschek, 1992; Volpe y Khosla, 1987). Dicha interpretación obedece a que el robot se comporta como una partícula deslizándose en el campo de potencial definido. Si la orografía asociada al campo ha sido adecuadamente definida, la partícula iniciará su marcha de modo espontáneo en el punto de salida, seguirá la línea de máxima pendiente evitando los obstáculos hasta llegar a la meta.

A pesar de la variedad de métodos para resolver el problema de la planificación, la complejidad del entorno, el número de grados de libertad y las restricciones cinemáticas y dinámicas que puede presentar el robot son factores que suelen exceder las limitaciones de estos métodos o incrementar excesivamente el tiempo de cómputo para hallar la solución. Frente a esta problemática, los métodos aleatorios parecen ofrecer interesantes alternativas por su sencillez de implementación y su comportamiento ventajoso en algunos tipos de escenarios.

Dentro de los métodos de generación aleatoria se encuentran los mapas probabilísticos o "*Probabilistic Roadmaps"* (PRMs) que también utilizan un grafo. En un proceso iterativo se escoge un punto aleatorio dentro del espacio de configuraciones y se agrega al grafo como un nuevo nodo. Después se trata de conectar dicho nodo a los demás y si existe conectividad se añade el correspondiente arco al grafo (Amato y Yan, 1996; Kavraki y otros, 1996b).

Uno de los métodos de generación aleatoria más reciente es el algoritmo *"Rapidly Exploring Random Trees"*, o RRT (LaValle, 1998). Se ideó en principio como herramienta de apoyo para ser integrada dentro de otros planificadores (PRM y campos de potencial). Sin embargo, utilizado como planificador autónomo, el RRT pronto reveló cualidades muy interesantes (LaValle y Kuffner, 1999). Muestra de ello ha sido la abundancia publicaciones que se han editado al respecto hasta la fecha, ofreciendo diferentes versiones que intentan mejorarlo como el "RRT-Connect" (Kuffner y LaValle, 2000), o el ERRT (Bruce y Veloso, 2002).

Dentro de este problema general, existe un ámbito particular centrado en los robots no holónomos, o dicho de otra forma, robots que presentan restricciones diferenciales no integrables (como se describirá más adelante, dichas restricciones son difíciles de tratar, y sin embargo muy habituales). A esta clasificación se adscriben todos los vehículos rodados, debido a la imposición de no deslizamiento del punto de contacto de las ruedas. De entre éstos cobran evidente interés aquellos que responden a la configuración “Ackerman” (Campion y otros, 1996), es decir, los que se comportan desde el punto de vista cinemático como el automóvil común. Algunos de los métodos para obtener trayectorias con este tipo de vehículos son heurísticos, entre los que se dispone de diversas adaptaciones del RRT.

Por otro lado existen diversos recursos relativos a la planificación de trayectorias en robots no holónomos aportados en recientes trabajos de investigación, como por ejemplo las maniobras restringidas (Gómez Bravo y otros, 2001b), o sucesión de maniobras capaces de alterar una y sólo una de las coordenadas de configuración del robot. Estas maniobras restringidas y el planificador RRT han sido la base a partir de la cual se ha desarrollado esta investigación.

1.2 Objetivos de la investigación

Como primera medida, ha sido objetivo de esta investigación el estudio de los algoritmos de planificación en general y del RRT en particular, contrastando sus ventajas comparativas en su aplicación a robots móviles no holónomos.

La pretensión inicial ha sido el desarrollo de un nuevo planificador que mejore las variantes del RRT propuestas, aprovechando las virtudes que ofrecen las maniobras restringidas. En concreto, se ha particularizado su aplicación sobre dos tipos de sistemas no holónomos muy frecuentes como son el robot diferencial y el tipo Ackerman. Para ello se ha desarrollado una aplicación informática en la que se han integrado las distintas versiones del RRT y la innovación propuesta.

Posteriormente, vistos los resultados de la integración de las maniobras restringidas, se sopesó la idoneidad de un planificador diferente al RRT, con la propiedad de distanciarse de los obstáculos. Es el caso de los diagramas de Voronoi. Se optó por

buscar un algoritmo menos complejo y no necesariamente tan preciso como los diagramas de Voronoi. Así surgió VODEC (diagrama de Voronoi con Descomposición Exacta de Celdas), como una herramienta específica para su uso combinado con las maniobras restringidas.

Las trayectorias obtenidas con estos algoritmos presentan trazados a veces próximos a lo que un conductor humano escogería, y a veces alejados, con innecesarios cambios de sentido o recorridos excesivamente amplios. Sin embargo, no resulta fácil decidir si una determinada trayectoria es mejor que otra, dado que existen múltiples factores implicados. Algunos fácilmente cuantificables como la distancia recorrida y otros no tanto, como por ejemplo la cercanía de los obstáculos que influye decisivamente en la factibilidad del camino de forma autónoma. Así pues ha sido necesario también un estudio de la aplicación de técnicas de decisión multicriterio para evaluar las trayectorias.

La validación de las trayectorias para ser seguidas por un robot ha sido uno de los objetivos de mayor interés en esta tesis. Con este fin se ha utilizado el robot Romeo-4R, con el que se han realizado diversos experimentos para constatar el correcto seguimiento de las mismas. Además, se ha explorado la posibilidad de integrar el planificador en un escenario con una red de sensores distribuida. La información proporcionada por éstos permite detectar cambios en el escenario que pueden requerir modificar localmente la trayectoria. Esta capacidad resulta especialmente útil en escenarios dinámicos. Lograrlo ratifica el trabajo teórico previo, y culmina el trabajo de investigación realizado.

1.3. Marco de realización

Parte de los trabajos realizados en esta tesis han sido desarrollados dentro del marco de actividades subvencionadas por el proyecto “Robots aéreos y redes de sensores con nodos móviles para la percepción cooperativa” (AEROSENS), financiado por la Dirección General de Investigación (DPI2005-02293. 2005-2008) y cuyos resultados han sido utilizados en el proyecto europeo URUS (Ubiquitous Networking Robotics in Urban Settings).

1.4. Aportaciones

Las aportaciones realizadas en esta tesis como es lógico están ligadas a los objetivos de la investigación expresados anteriormente, y quedan detalladas en la lista siguiente:

I. La idea de partida de esta investigación ha consistido en combinar las bondades del RRT (que como se verá más adelante consisten en su velocidad computacional y sencillez de implementación fundamentalmente), con las maniobras restringidas (especialmente útiles cuando surge el problema de la no holonomía). De este nuevo planificador se estimaba que fuera, al menos, superior en prestaciones al RRT con restricciones. En efecto, así ha sido. Es más, tras los experimentos realizados, los resultados han sido mucho mejores de lo esperado, obteniendo una velocidad de cómputo un orden de magnitud menor con respecto al RRT adaptado. Además, las soluciones obtenidas pueden ser tanto maniobras elaboradas (trayectorias con cambios de sentido) como trayectorias simples. En consecuencia se ha desarrollado un método prometedor que, trascendiendo los particulares feudos de aplicación preferencial de los RRT, puede llegar a ser decididamente competitivo con otros planificadores más tradicionales. Esta innovación se ha implementado en concreto sobre robots tipo diferencial y sobre robots tipo “Ackerman”, aportando un conjunto diferente de maniobras restringidas para cada uno de ellos. En ambos casos y para los escenarios estudiados, resultó ventajosa la mejora disminuyendo ostensiblemente el tiempo de cómputo. No obstante, este método no está restringido a dichos vehículos. Para cada nuevo tipo bastaría con desarrollar el conjunto de maniobras restringidas adecuado e incluirlas sin más en el algoritmo.

II. Una nueva y específica implementación de los diagramas de Voronoi, designada como VODEC, ha sido desarrollada a partir de las necesidades particulares de las maniobras restringidas. Este nuevo método utiliza una métrica no euclídea en interés de una mayor simplicidad y la capacidad de reducir el problema de obtención del Voronoi global a resolverlo en celdas individuales que además conforman un grafo de adyacencia.

III. Otra de las aportaciones logradas en este trabajo, ha sido el estudio de diferentes aplicaciones del nuevo planificador. Durante el proceso investigador surgió

una cuestión que deriva directamente de la naturaleza probabilística del RRT que consiste en la capacidad de generar diferentes caminos para un mismo problema. La selección de la mejor trayectoria no es sencilla, pues además de la longitud existen otros parámetros de calidad dignos de tener en cuenta. Los algoritmos de decisión multicriterio constituyen una herramienta ideal para acometer este problema. Una de las aportaciones ha sido su aplicación.

IV. Es de interés asimismo la navegación utilizando información de redes de sensores distribuidos, es decir, la planificación de trayectorias que además de obstáculos pueden verse condicionadas por el resultado de ciertos sensores distribuidos por el entorno de trabajo. La dificultad estriba en la necesidad de atender puntos de paso obligados y eludir zonas de riesgo. En este caso se ha adaptado el planificador con resultados satisfactorios.

V. Otra aportación de utilidad generada durante la investigación consiste en el programa de trabajo y su integración en la arquitectura de un robot real, donde debe gestionar una red de sensores inalámbricos. Esta aplicación se ha desarrollado sobre java, con idea de facilitar la extensión de los resultados a la comunidad científica dada su portabilidad. Este planificador contiene los algoritmos RRT y Vodec en todas las variantes referidas, además de los nuevos métodos desarrollados. En el apéndice se recogen de forma más detallada sus características.

Por último los trabajos de investigación hasta ahora realizados han permitido publicar los siguientes artículos:

1) Artículos en revistas:

1. "Aplicación del algoritmo RRT a la planificación de movimientos en robots no holónomos.", D. López, F. Gómez-Bravo, F. Cuesta y A. Ollero. *Revista Iberoamericana de Automática e Informática Industrial (2006)* ISSN: 1697-7912

2. "A New Approach for Car Like Robots Manoeuvring Based on Rrt". Fernando Gomez Bravo, Anibal Ollero Baturone, Federico Cuesta Rojo, Diego López García. *Robótica: Automaçao, Controlo e Instrumentaçao*. Núm. 71. 2008. Pag. 10-14

3. "Diseño de Trayectorias para el Guiado de Vehículos Autónomos en Entornos Hospitalarios". Fernando Gomez Bravo, Diego López García, José Macías Macías,

José Manuel Martin Ramos, Marcos del Toro Peral, Juan Carlos Fortes Garrido. *Todo Hospital*. Núm. 252. 2008. Pag. 572-577

4. "Application of Multicriteria Decision-Making Techniques to Manoeuvre Planning in Nonholonomic Robots". José Manuel Martin Ramos, Diego López García, Fernando Gómez Bravo, Armando Blanco Morón. *Expert Systems With Applications*. 2010. Pag. 3962-3976.

5. "Vodec: A fast Voronoi algorithm for car-like robot path planning in dynamic scenarios". Diego A. López García, Fernando Gómez Bravo. Enviado a la revista "Robotica" y en proceso de adecuación a los comentarios de los revisores.

2) Aportaciones a congresos:

1. "Aplicación del algoritmo RRT a la planificación de movimientos en robots no holónomos", D. A. López, F. Gómez-Bravo, F. Cuesta y A. Ollero. *XXV Jornadas de Automática (2004)* ISBN: 84-688-7460-4

2. "Aplicación de técnicas de decisión multicriterio a la planificación de maniobras en robots no holónomos", F. Gómez Bravo, J.M. Martín Ramos, D. A. López García, M. P. Polo Almohano, M. Del Toro Peral, E. Cortés Ancos, J. C. Fortes Garrido. *II Simposium de Control Inteligente (2006)*

3. "Comparativa entre planificadores de trayectorias para su uso combinado en la generación de maniobras" Diego A. López García, Fernando Gómez-Bravo, *Jornadas de automática 2007*

4. "A New Approach for Car-Like Robots Manoeuvering". *Robótica 2007 - 7th Conference on Mobile Robots and Competitions* . Paderne (Albufeira), Portugal. Centro Ciência Viva Do Algarve. 2007. Fernando Gómez Bravo, Aníbal Ollero Baturone, Federico Cuesta Rojo, Diego A. López García.

5. "Rrt-D: a Motion Planning Approach for Autonomous Vehicles Based on Wireless Sensor Network Information". *Proceedings of the 6th IFAC Symposium on Intelligent Autonomous Vehicles.* (6). Num. 6. Toulouse, Francia. IFAC. 2007 Fernando Gomez Bravo, Diego López García, Federico Cuesta Rojo, Anibal Ollero Baturone.

6. "Rrt-D: Planificación Distribuida de Caminos Basada en la Información de una Red de Sensores wireless". *XXVIII Jornadas de Automática*. 2007. Pag. 1-8. ISBN: 978-

84-690-74. Fernando Gómez Bravo, Aníbal Ollero Baturone, Diego A. López García, Federico Cuesta Rojo, Marcos del Toro Peral. **Artículo premiado como mejor aportación en la temática de robótica en las XVIII Jornadas de Automática.**

7. "Integrated Path Planning and Tracking for Autonomous Car-Like Vehicles Maneuvering". Fernando Gomez Bravo, Diego López García, Francisco Javier Real Pérez, Luis Merino Cabañas, Jose Manuel Sánchez-Matamoros Pérez. *Proceedings 6th International Conference on Informatics in Control, Automation and Robotics (Icinco 2009)*. Num. 6. Milán. Institute for Systems and Technologies of Information Control and Communication (Insticc). 2009. Pag. 457-464. ISBN: 978-989-674-0

8. "Algoritmo genético con evaluación borrosa multicriterio. Aplicación a la planificación de movimientos en manipuladores paralelos". Diego López García, Fernando Gómez Bravo. José Manuel Martín Ramos, Ismael López Quintero Giusseppe Carbone. *XV Congreso Español Sobre Tecnologías y Lógica Fuzzy* (ESTYLF'10). 2010.

1.4 Estructura del documento

La información aportada en esta tesis se divide en siete capítulos y un apéndice.

El capítulo 1 recoge tal y como se observa, la introducción.

El capítulo 2 se dedica a los diferentes algoritmos de planificación utilizados. Inicialmente se aclaran una serie de conceptos fundamentales comunes a todos los métodos. Posteriormente se pormenorizan los detalles de cada método, de los que ya se ha realizado una breve introducción. En particular se hace especial desarrollo del RRT. En él se detalla su génesis, el modelo primitivo, sus diferentes variantes y las características más relevantes del método. El capítulo resulta un sucinto compendio de lo publicado hasta el momento.

El capítulo 3 presenta el nuevo método de planificación denominado VODEC. En primer lugar se describe el método. A continuación sus características como diagrama de Voronoi alternativo, y como planificador ante escenarios dinámicos.

El capítulo 4 particulariza el problema de la planificación a sistemas no holónomos. Comienza con la definición de no holonomía y continúa con la descripción de

los dos tipos de robots no holónomos más habituales: el vehículo diferencial y el tipo “Ackerman”. Seguidamente se explica el concepto de maniobra restringida y se describen las adecuadas para cada tipo de robot. Además se explica el nuevo método de planificación que las usa, consistente en un algoritmo sin restricciones holónomas y otro de depuración de sus resultados denominado “postprocesado".

El capítulo 5 describe las diferentes implementaciones que pueden derivarse de la combinación de uno de los planificadores previamente descritos con las maniobras restringidas y su comparativa. En especial se contrasta el nuevo algoritmo desarrollado (VODEC) con respecto a los demás.

El capítulo 6 explora las diferentes formas de evaluar las trayectorias obtenidas y su aplicación. En concreto se utilizan los métodos de decisión multicriterio. Al final del mismo se propone el uso conjunto del algoritmo RRT con un algoritmo genético.

El capítulo 7 se dedica a los experimentos realizados sobre un robot real. Además se estudia un sistema particular de planificación orientado a la explotación de las redes de sensores que permite contender con escenarios dinámicos.

El capítulo 8 describe las diferentes conclusiones obtenidas y las futuras líneas de investigación.

El apéndice A ofrece información acerca de la aplicación informática desarrollada describiendo sus capacidades y mostrando el interfaz de usuario.

Por último, se anexan las referencias bibliográficas.

CAPÍTULO 2 Algoritmos de planificación

2.1 Introducción

Desde su aparición en 1998 los algoritmos RRT han generado enorme interés con numerosas publicaciones que lo sitúan como tema de actualidad en el campo de la planificación de trayectorias. En esta tesis se expone el resultado de un trabajo de búsqueda, clasificación y sumario de los artículos de mayor impacto publicados hasta la fecha sobre el tema. Se ha pretendido ofrecer un resumen didáctico de su contenido, detallando su fundamento, sus diversas adaptaciones y una relación de virtudes y defectos con respecto a otros métodos. Por ello resulta necesario exponer las bases sobre planificación de trayectorias.

En este capítulo se describen los métodos de planificación más significativos, escogidos entre los más conocidos en la comunidad científica. Cada uno de ellos tiene sus ventajas e inconvenientes, siendo seleccionados en función del problema de planificación a tratar. Los factores que influyen en su idoneidad son muy variados. Por ejemplo un planificador muy eficaz en escenarios con pocos obstáculos (como el diagrama de visibilidad) puede ser demasiado lento cuando éstos aumentan. Hay planificadores que requieren una fase preparatoria ardua, pero que una vez ejecutada

les resulta inmediato el cálculo de una trayectoria válida, incluso aunque se varíen los puntos de partida y fin. Si el robot presenta restricciones cinemáticas o dinámicas el problema se vuelve más complejo. Hay planificadores que no pueden contender fácilmente con estas condiciones. A veces el objetivo no es buscar una trayectoria, sino varias. O incluso no alcanzar una configuración, sino determinar las regiones a las que se tiene acceso. En ocasiones los obstáculos varían su posición con el tiempo, exigiéndose del planificador adaptar la trayectoria inicialmente calculada.

Todos estos factores hacen interesante conocer las cualidades de los planificadores, por lo cual se exponen a continuación. En concreto interesa conocer las características comparativas con el planificador RRT que se explicará más adelante, justificando en lo posible su elección como base de esta investigación.

Comunes a los planificadores hay una serie de elementos que conviene definir. Por ejemplo, aunque es conocido el concepto de robot, en el ámbito de la planificación suele ser considerado como un conjunto de sólidos rígidos cada uno de los cuales ocupa una posición y orientación en el espacio. Cuando se habla de robot móvil se refiere a aquel dotado de un sistema de locomoción que le permite desplazarse en un entorno dado (Ollero, 2001). Los más comunes entran en la categoría de vehículos rodados, pudiendo desplazarse en dos dimensiones. A continuación se detallan una serie de conceptos que también merecen aclaración (Latombe, 1991):

- Configuración de un robot: conjunto mínimo de parámetros capaz de identificar de forma única la posición y orientación de todos y cada uno de los sólidos rígidos que componen el robot. Por ejemplo, en un robot manipulador consistiría en el conjunto de ángulos que forman cada una de sus articulaciones.

- Configuración origen: Es la configuración que presenta el robot antes de iniciar cualquier acción. En el problema de la planificación es el punto de partida.

- Configuración destino: Es la configuración que se desea alcanzar partiendo de la anterior.

- Espacio de configuraciones: Es el conjunto de todas las configuraciones que puede adoptar el robot independientemente del escenario en el que se ubique. La dimensión de este espacio será igual al número de parámetros necesario para describir una configuración. Habitualmente se denota como C. Obsérvese además, que el espacio de configuraciones no coincide con el espacio físico. Incluso en robots semejantes a un

automóvil, su configuración consiste en dos variables coincidentes con la posición cartesiana de su centro (o de otro punto de referencia del mismo) y un ángulo que define su orientación. Se estaría hablando de un espacio de dimensión 3 cuando el vehículo sólo se mueve en el plano.

- Colisión: Se dice que un robot entra en colisión cuando su configuración sitúa uno o varios de sus elementos intersecando el espacio ocupado por obstáculos o los límites del escenario de estudio.

- Obstáculo: Para el ámbito de la planificación, un obstáculo se considera como un conjunto cerrado y acotado de puntos del espacio físico. Es decir, sólo interesa del obstáculo el volumen que ocupa y no su naturaleza. Así, un obstáculo también puede ser un lugar que por alguna razón no se desea que el robot ocupe.

- Obstáculo-C: Sea q un elemento de C. Denotamos como $A(q)$ el espacio físico que ocupa el robot cuando adopta la configuración q. Sea B_i un obstáculo. Se define el obstáculo-C asociado a B_i (abreviadamente CB_i), como el conjunto de configuraciones cuyo espacio físico colisiona con B_i. Es decir, CB_i es el efecto de B_i en el espacio de configuraciones:

$$CB_i := \{ q \in C : A(q) \cap Bi \neq \varnothing \} \qquad (2\text{-}1)$$

La existencia de colisión del robot con un obstáculo implica que el espacio que ocupa, $A(q)$, es compartido con el que ocupa el obstáculo, B_i, o lo que es lo mismo, la intersección de $A(q)$ y B_i no es nula (2-1) porque existen puntos de colisión.

- Región de obstáculos-C: Es la unión de todos los obstáculos C. Se denota como CB:

$$CB := \bigcup_{i=1}^{i=n} CB_i \quad n = \text{número de obstáculos} \qquad (2\text{-}2)$$

- Espacio de configuraciones libres de colisión: Es el subconjunto del espacio de configuraciones cuyos elementos no presentan colisión. Habitualmente se denota como C_{free}.

$$C_{free} := C \setminus CB \qquad (2\text{-}3)$$

Igualmente el espacio de configuraciones libres de colisión deja de ser coincidente con el espacio físico libre de obstáculos. Por ejemplo, en el caso de un robot móvil, para una misma posición cartesiana de su centro, puede haber orientaciones en colisión, y orientaciones permitidas; de modo que el aspecto de dicho espacio puede parecer poco intuitivo.

- Camino o trayectoria: Sucesión continua de configuraciones adoptadas por un sistema robótico cuando se traslada desde una configuración origen a otra de destino. Para que una trayectoria sea admisible ha de tener todos sus elementos dentro del conjunto de configuraciones libres de colisión y cumplir con las restricciones propias del problema. En la práctica se define una determinada métrica en el espacio de configuraciones. La trayectoria se identifica con un subconjunto discreto y finito de la sucesión, generándose de modo que sus elementos disten entre sí un valor determinado. De este modo resultan estructuras de datos útiles para su uso en algoritmos.

Aunque la aplicación natural de los métodos de planificación sean los robots móviles, también es necesario su uso en otro tipo de robots. Así pues, todos estos conceptos no discriminan entre uno y otro, y la mayoría de los métodos que se describen a continuación son igual de válidos en cualquier caso.

Muchos de los métodos que se describen a continuación, utilizan en una primera etapa el modelo de un robot puntual (constituido por un único cuerpo de tamaño nulo) cuya configuración consiste en la posición cartesiana del mismo. De este modo se logra hacer coincidentes el espacio de configuración con el espacio euclídeo y el espacio de configuraciones libres de colisión con el espacio libre de obstáculos. Aunque los diagramas y ejemplos que a continuación se detallan utilizan este modelo, esto no quiere decir que las capacidades de estos métodos se limiten al mismo. Muy al contrario la mayoría pueden extrapolar sus capacidades a espacios de configuración de dimensión elevada, aunque por supuesto entonces dejan de coincidir con el espacio físico, y los ejemplos resultarían menos intuitivos.

Con objeto de situar los siguientes métodos dentro de un marco más general y lograr una visión estructurada de los sistemas de planificación, se ofrece la siguiente clasificación (Latombe, 1991):

-Métodos “roadmap”: Son aquellos que construyen una descripción del espacio de configuraciones libres de colisión en forma de una red de curvas unidimensionales, que formarán parte del espacio de soluciones. Entre estos métodos se encuentran: los grafos de visibilidad, los diagramas de Voronoi y otros.

-Métodos de descomposición en celdas: Se basan en la descomposición del espacio de configuraciones libres de colisión en regiones convexas denominadas celdas. Cada una de estas celdas es representada por un nodo en el denominado grafo de adyacencia. Cuando dos de dichas celdas tienen un límite en común, se agrega un arco al grafo que une sus correspondientes nodos. Este grafo se utilizará para hallar la solución. La descomposición vertical y las técnicas quadtree son ejemplos de esta categoría.

-Métodos de campos de potencial: Se genera una función escalar sobre el espacio de configuraciones de modo que su gradiente pueda ser utilizado como guía para trazar la trayectoria. La semejanza que éste sistema ofrece con respecto a una partícula deslizándose merced al campo de potenciales artificial (función escalar) creado a partir de los obstáculos y las configuraciones inicial y final, da nombre a dichos métodos.

-Métodos probabilísticos: Son aquellos que utilizan generación de configuraciones aleatorias para identificar su entorno. Ejemplos de estos métodos son los RRT y los PRM.

2.2 Grafos de visibilidad

Tal y como se describía en la introducción, éste y otros métodos se basan en la construcción de un grafo que representa en sus nodos configuraciones factibles del robot, y en sus arcos el coste de conexión directa entre dos vértices. Una vez construido dicho grafo, denominado grafo de conectividad, hallar un camino entre el origen y el destino se reduce a conectar dichos puntos con el grafo y hallar una secuencia de nodos dentro de él (Latombe, 1991). El término de visibilidad alude a que en este método se enlazan los vértices entre los que existe visión directa, es decir, sin obstáculos que lo impidan.

El método de grafos de visibilidad como otros planificadores, precisa de una descripción poligonal de los obstáculos-C. Si se asume el modelo de un robot puntual, tal y como se expresaba en la introducción, esto implica que los obstáculos coincidan con los obstáculos-C. Así, el entorno de estudio queda definido por los límites del plano en el que se desplaza el robot y un conjunto de polígonos que identifican los obstáculos. Dentro del espacio de configuraciones libres de colisión se sitúan las posiciones de partida y meta, quedando completamente definido el problema.

El proceso comienza generando una lista de los vértices de los polígonos a los que se añaden las posiciones inicial y final (Muñoz, 1995). Con estos puntos se genera una matriz cuadrada con tantas filas como puntos. A continuación se evalúa para cada par de puntos su posibilidad de enlace directo, es decir, si el robot ubicado en uno de los puntos puede alcanzar el otro desplazándose en línea recta. Si un obstáculo no lo impide, esto será posible, y la longitud del tramo a recorrer se almacena en la matriz indicando el coste de realizar dicho enlace. Si por el contrario existe obstáculo, en la matriz se anotará un valor que indique tal eventualidad. Esta matriz no es más que la descripción de un grafo que une los puntos entre los que es posible el enlace directo (ver figura 2.1).

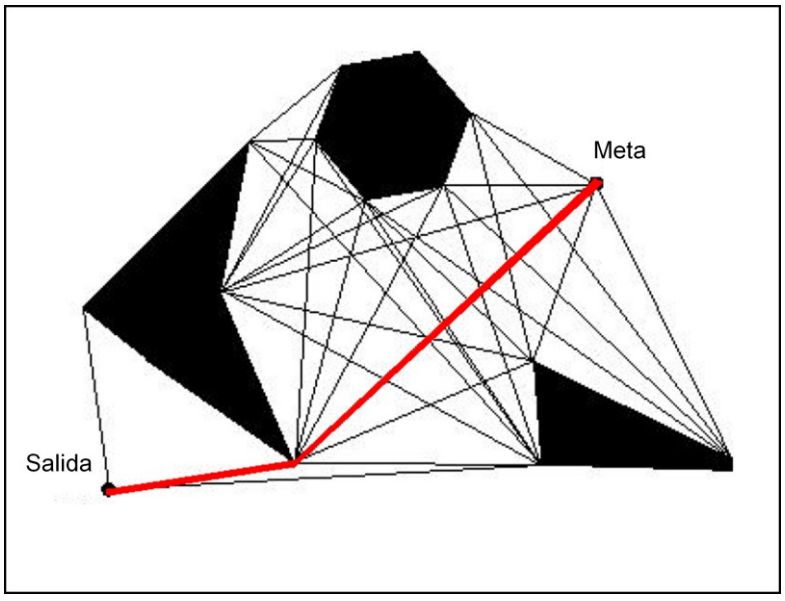


Figura 2.1: Grafo de visibilidad

Recorriendo este grafo se puede alcanzar cualquier vértice. Así pues, una solución, es decir, una trayectoria admisible que una la configuración inicial con la final,

se consigue escogiendo una sucesión de vértices conexos dentro del grafo en la que el primero sea el punto de partida y el último la meta. Entre los muchos caminos a escoger se suele utilizar el criterio de mínima distancia. Para ello el algoritmo de Dijkstra es el más utilizado aunque hay otros (Overmars y Welzl, 1988; Angelier, 2002).

El coste computacional de este algoritmo se eleva ostensiblemente con el número de vértices. Sobre este particular existen ciertas mejoras como por ejemplo la eliminación de vértices cóncavos en el cálculo del grafo. Ello se debe a que la trayectoria que utilice un vértice cóncavo, será siempre de mayor longitud que otra que pase por los mismos puntos y sustituya dicho vértice por un punto próximo de su bisectriz (ver figura 2.2). Esto demuestra que no es la trayectoria óptima y por tanto, todos los arcos del grafo pertenecientes a dicho vértice jamás estarán en la solución (Latombe, 1991).

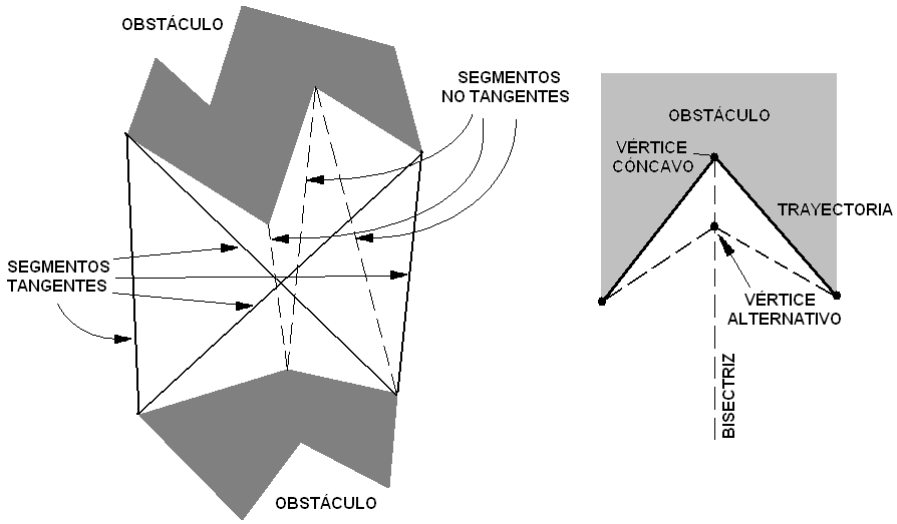


Figura 2.2: Segmento tangente y vértice cóncavo.

Otro concepto interesante que aporta una reducción significativa en el grafo, es el de segmento tangente a un polígono. Sean dos polígonos *A* y *B*, *X* e *Y* dos vértices de los mismos ($X \in A$, $Y \in B$), *L* la línea que pasa por *X* e *Y*. Se dice que el segmento *XY* es tangente si *L* no interseca *A* ni *B*. Se demuestra que la solución dentro del grafo de visibilidad, si existe, está compuesta por segmentos tangentes (Latombe, 1991). Así se eliminan bastantes arcos del grafo disminuyendo el coste computacional.

Existen ciertas peculiaridades asociadas a la aplicación práctica de este algoritmo. No se abundará en ellas pero sí se comentará alguna en particular que resulta de especial interés en este trabajo.

Para empezar la designación puntual del robot es una aproximación intuitiva a la descripción de la configuración del robot como un punto en el espacio de configuraciones. Tal y como se describe en la introducción, un obstáculo, aunque tenga la morfología de un polígono, supondrá una sustracción al conjunto de configuraciones libres de colisión determinada, cuya proyección en el plano no tiene por qué coincidir con el obstáculo. Para simplificar esta cuestión (facilitar la implementación informática), se aumenta el área del polígono de modo que inscriba el espacio dentro del cual el robot, al menos en alguna orientación, puede colisionar. Es habitual tomar un punto de referencia para el robot, de modo que la descripción de su posición (su configuración) se establezca con las coordenadas cartesianas de este punto de referencia y su orientación (ángulo del eje central con respecto al eje de abscisas). Llegados a este punto se puede concretar mejor el concepto de obstáculo-C.

Sea *B* el área poligonal que representa un obstáculo dado. Sea *A(θ)* el área ocupada por el robot con su centro de referencia ubicado en el origen y con la orientación *θ*. Con esta orientación fija, el conjunto de configuraciones que entran en colisión con *B*, que denotaremos como CB_θ, se hallaría variando la posición *x* en el plano del centro de referencia del robot, y sería:

$$CB_\theta = B \ominus A(\theta) = \{x \in \mathbb{R}^2 \,/\, \exists b \in B, \exists a \in A(\theta) : x = b - a\} \qquad \textit{(2-4)}$$

Donde Θ es el operador de Minkowski de diferencia. Para todas las orientaciones posibles se obtiene el obstáculo-C completo asociado CB:

$$CB = \cup_{\forall \theta} CB_\theta \qquad \textit{(2-5)}$$

A efectos prácticos suele tomarse la dimensión desde el punto de referencia del robot a su vértice más alejado como la distancia con la que recrecer los polígonos de los obstáculos evitando así la colisión. No obstante es habitual encontrar escenarios donde existen pasillos estrechos de paso obligado. Con esta medida, los obstáculos recrecidos pueden solaparse en los pasillos impidiendo hallar la solución.

Por último hay que significar que si se desea utilizar directamente la solución aportada por este algoritmo como camino para ser seguido por un robot, un aspecto relevante consiste en el hecho asumido de que el robot es capaz de girar sobre sí mismo. Si observamos la solución del algoritmo de visibilidad, ésta consiste en una secuencia de segmentos unidos por vértices. El robot, para seguirla, debería avanzar

recto por cada segmento y orientarse en cada vértice sin abandonarlo. Esto, como veremos más adelante, es posible en algunos robots como los vehículos de conducción diferencial, sin embargo existen otros en los que resulta del todo imposible por tener acotada su curvatura.

2.3 Diagramas de Voronoi

Los Diagramas de Voronoi son en realidad los grafos que dan el nombre al método. En este caso el grafo resultante tiene la ventaja de que maximiza la distancia entre el robot y los obstáculos. Este método se utiliza bien directamente (Aurenhammer, 1991) o bien como base a otros métodos (Wilmarth y otros, 1998). Al igual que en el anterior método, la aplicación de éste a la planificación exige el modelado de los obstáculos como obstáculos-C poligonales. Lo cual permite adoptar la analogía del robot puntual, es decir, reducir la búsqueda de una solución a identificar el camino seguido por un punto en el plano (dentro del espacio de configuraciones libres de colisión) que representa de la posición del robot.

Un diagrama de Voronoi generalizado (denotación utilizada para diferenciarlo del original, en el que los obstáculos se reducían a puntos) representa los puntos que equidistan de elementos vecinos entre sí. Estos elementos pueden ser obstáculos o bien bordes del escenario. El atributo de vecindad quiere expresar que no hay ningún otro elemento entre dos que sean vecinos.

Dado que el lugar geométrico de los puntos del plano que equidistan de un punto y una recta es una parábola, y el lugar geométrico de los puntos del plano que equidistan de dos rectas es a su vez una recta, si los obstáculos son polígonos, el diagrama estará compuesto de una sucesión de parábolas y segmentos rectos interconectados. Si se toman los puntos de interconexión como vértices, y los fragmentos de parábolas y rectas como arcos, se dispone de un grafo. No obstante los diagramas de Voronoi también pueden extraerse de escenarios donde existan obstáculos con lados curvos, aunque en estos casos la forma del diagrama dependerá de estos elementos sin poder acudir a estructuras predefinidas.

Una vez generado el grafo, se procede a conectar los puntos de partida y meta al mismo (Muñoz, 1995). Si el punto se halla en la vecindad de dos segmentos, se halla la línea recta ortogonal que une dicho punto al segmento más próximo. El segmento de dicha recta definido por su intersección con el grafo y el punto considerado, constituirá la unión de este punto al grafo. Si por el contrario el punto se halla en la vecindad de un vértice y otro elemento, se halla la recta que pasa por el vértice y el punto, determinándose de la misma forma que en el caso anterior el arco del grafo necesario (ver figura 2.3). A partir de aquí el problema resulta como en el caso anterior, es decir, se aplica el algoritmo de Dijkstra por ejemplo y se halla una secuencia de vértices en el grafo que una sendos puntos.

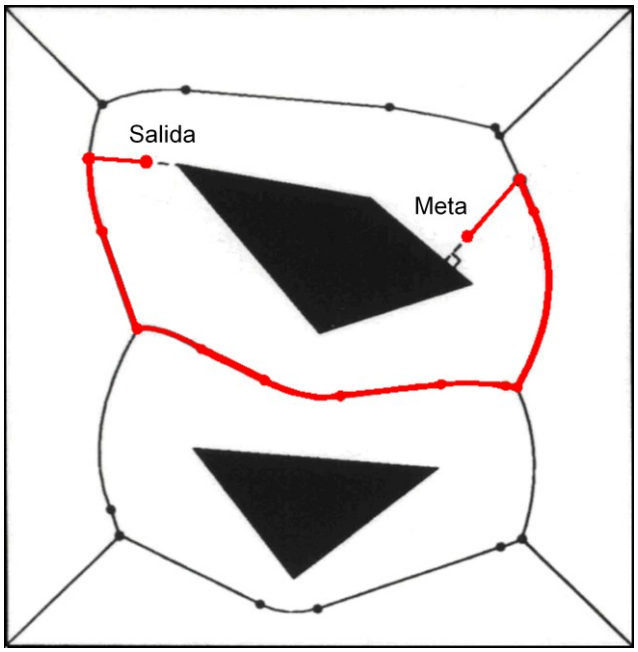


Figura 2.3: Diagrama de Voronoi

Al igual que en el caso anterior, si se desea utilizar el grafo como camino para ser seguido por un robot, es necesario asumir la capacidad de maniobra del mismo para girar abruptamente en algunos vértices. Por otro lado, la ventaja de este método radica en su capacidad para alejarse de los obstáculos, eligiendo siempre el camino más despejado posible. Esto puede ser muy interesante en escenarios con abundantes obstáculos y pasos estrechos, sin embargo resulta un inconveniente en entornos con escasos obstáculos. En este caso, dependiendo de la ubicación de los mismos, el

diagrama arrojará trayectorias que pueden dar rodeos largos e innecesarios para llegar a la meta.

2.4. Descomposición en celdas.

La idea general de estos métodos consiste en dividir el problema en dos niveles: uno global que resuelve la trayectoria por zonas y otro, más simple, a nivel local. Se trata de dividir el entorno de estudio en un conjunto de celdas (Latombe, 1991). Éstas han de tener la peculiaridad de que resulte sencilla la conexión de dos puntos cualesquiera de su interior, bien porque sean muy próximos, porque la celda sea convexa, etc. Una vez discretizado el espacio de configuraciones libres de colisión en un conjunto de celdas completo, se procede a resolver su conectividad a un nivel superior, por ejemplo construyendo un grafo de adyacencia. Es decir, a cada celda se le asigna un nodo, y éstos se unirán o no en función de que las celdas que representen sean adyacentes (de nuevo un grafo de conectividad). Existen dos subtipos principales:

- Descomposición exacta: Se utiliza este término cuando la descomposición genera un conjunto de celdas cuya unión resulta idéntica al espacio de configuraciones libres de colisión. Es decir, las celdas han de estar de tal modo definidas que se adapten a la configuración de los obstáculos. Por ejemplo, en un problema en el que el escenario y los obstáculos tengan límites poligonales (ver figura 2.4), se pueden trazar rectas verticales en cada vértice. De esta forma, todo el espacio de configuraciones libres de colisión queda fragmentado en celdas trapezoidales o triangulares en su totalidad. A partir de aquí se genera el grafo de conectividad y se halla la secuencia de celdas (secuencia de vértices del grafo) en donde ha de estar contenida la trayectoria. Para ésta, basta con elegir los puntos medios de los segmentos adyacentes como puntos de paso. Trazando segmentos rectos entre los mismos se obtiene la solución. Este tipo de fragmentación en celdas se denomina “Descomposición Vertical” (Chazelle, 1987). Precisa una ordenación previa de los vértices por abscisa, lo cual implica un tiempo de cómputo de O(*nlogn*) siendo *n* el número de vértices. A partir de aquí, el trazado de líneas verticales, la designación de celdas y la identificación de las que contienen la posición origen y la de destino, todo puede realizarse de modo concurrente en un tiempo O(*nlogn*) también. El número de celdas y el de sus enlaces coinciden en O(*n*). Para un

grafo de estas dimensiones, hallar un camino entre dos nodos puede realizarse en un tiempo O(*n*) aunque no sería el más corto. Si se desea esta última propiedad el algoritmo de búsqueda tardaría O(*nlogn*) (Latombe, 1991).

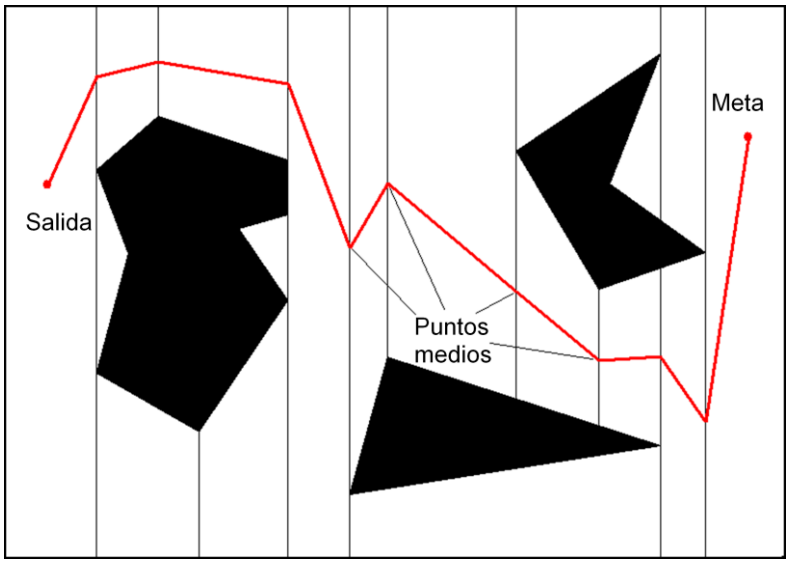


Figura 2.4: Descomposición en celdas exacta.

Otro ejemplo de descomposición en celdas exacta lo constituye la llamada triangulación de Delaunay. Ésta considera inicialmente el conjunto de puntos formado por los vértices de los obstáculos. A continuación une en un segmento los pares de puntos tales que es posible hallar una circunferencia que los contenga sin inscribir ningún otro punto. Esta exclusión implica que los puntos del par son los más cercanos al centro de la circunferencia hallada. Además, dicho centro equidista de los puntos a unir, puesto que para dos puntos cualesquiera de una circunferencia, la bisectriz del segmento que definen contiene siempre al centro de la misma. De ambos resultados se deduce que este centro pertenecerá al diagrama de Voronoi que definen los vértices de los obstáculos. No obstante el método no resuelve el diagrama de Voronoi, sino que únicamente define una descomposición de C_{free} en celdas triangulares delimitadas por los segmentos definidos con la propiedad mencionada. Sobre estas celdas se construye un grafo de adyacencia, y a partir de aquí se resuelve de la misma forma que en el apartado anterior

Si bien éste método resulta algo más complejo, los puntos medios de los segmentos de estas celdas se distancian con mayor eficacia de los obstáculos, y la

solución obtenida ofrece por tanto mayor margen de seguridad en orden a evitar colisiones (ver fig. 2.5).

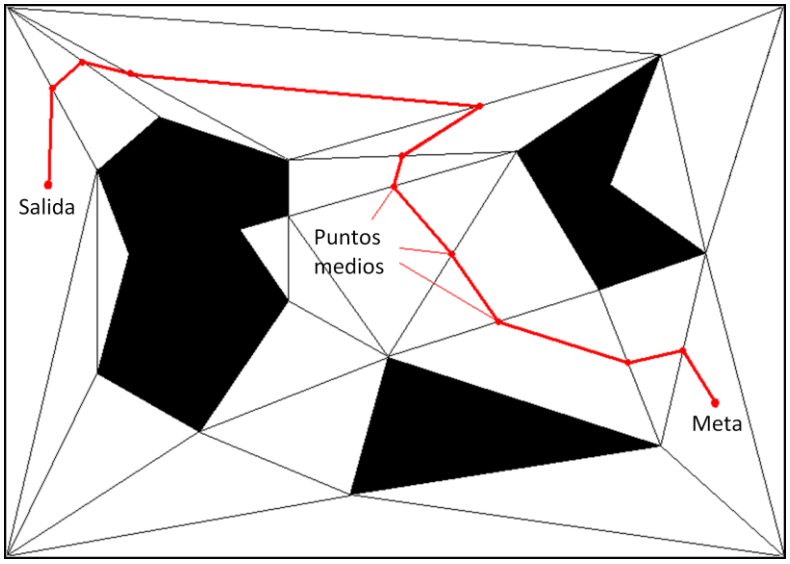


Figura 2.5: Descomposición de Delaunay

- Descomposición aproximada: En este tipo de descomposición se utilizan celdas de diseño predefinido, habitualmente rectangulares. Al descomponer todo el escenario en dichas celdas, algunas solaparán bien completamente bien en parte algún obstáculo. Éstas serán descartadas, las demás serán las que conformen el grafo de adyacencia. Por lo demás el proceso es bastante similar. La ventaja de este método con respecto al anterior reside en su sencillez de implementación. Como puede deducirse este tipo de descomposición no cumple con el requisito de la anterior debido a esas celdas descartadas que contienen en parte configuraciones libres de colisión. A continuación se describen dos ejemplos: el método de división en cuadrantes, también conocido como descomposición "*quadtree*", y el método de frente de ondas, aludido también como "*fast marching method*".

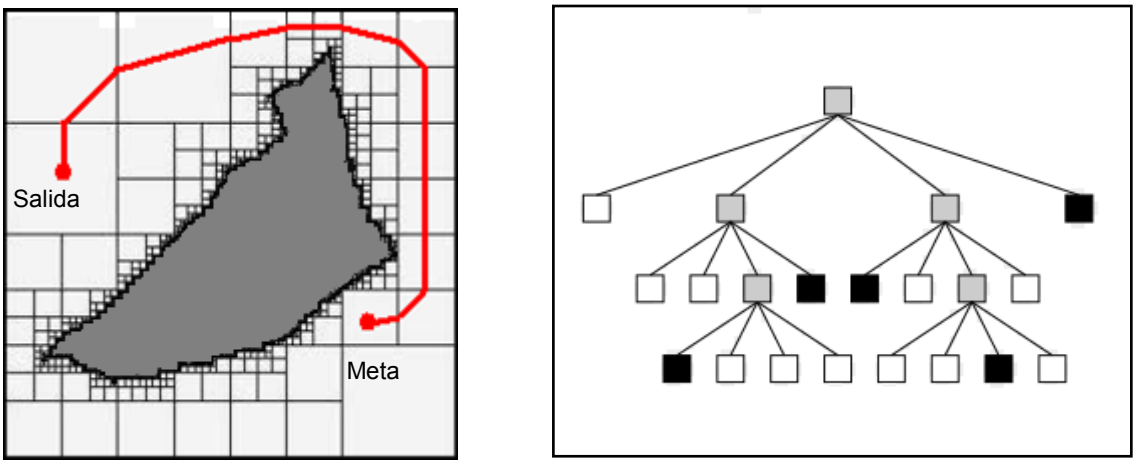
El método de descomposición "quadtree" parte de un escenario contenido en un rectángulo (si los límites del escenario no coincidieran con ninguno bastaría con inscribirlo en uno y añadir exceso como obstáculo). A continuación se divide el escenario en rectángulos semejantes mediante el procedimiento de trazar mediatrices en la base y la altura del original (ver fig. 2.6a). Así, se obtienen cuatro rectángulos. Se clasifican los rectángulos según su área esté contenida en *CB* (se les llamará "llenos"), en C_{free} (rectángulos "vacíos"), o tenga parte en ambos ("mixtos"). Éstos últimos volverán a sufrir

el proceso de fragmentación utilizando el mismo sistema. De nuevo serán etiquetados como llenos (si están dentro de CB), vacíos (si están en C_{free}) y mixtos, siendo los mixtos los sujetos del siguiente proceso.

En cada una de estas etapas se actualiza una estructura de datos de tipo árbol de grado 4. Es decir, de cada nodo del árbol descienden otros cuatro. El nodo raíz representa el rectángulo inicial que contiene el escenario. Sus cuatro nodos descendientes representan los rectángulos de la primera subdivisión. De la misma forma están relacionados los demás rectángulos, teniendo en cuenta que aquellos nodos que están etiquetados como llenos o vacíos no tienen nodos descendientes, sólo los mixtos.

En la fig. 2.6b se ha representado un ejemplo de este árbol marcando en gris los nodos mixtos, en blanco los vacíos y en negro los llenos. Utilizando este árbol, se comprueba en cada iteración si existe una secuencia de celdas vacías que conecte la celda origen con la celda destino. Si la hay, se procede a generar un camino utilizando los puntos medios de los rectángulos vacíos. Si no la hay, se vuelve a iterar. Estas iteraciones están limitadas a un determinado tamaño de celda, con lo que si se alcanza sin encontrar un camino, se devuelve que no ha sido posible hallar una solución.

Este método es adecuado para el procesamiento de escenarios definidos como imágenes, donde cada pixel pertenecerá o bien a C_{free} o bien a *CB*. En este caso se define la celda mínima con el tamaño de un pixel. De este modo, al llegar a ese nivel de descomposición, no quedarán celdas mixtas y el método se detiene de forma natural.



a) Partición del escenario *b) Estructura en árbol de las particiones*

Figura 2.6: Descomposición quadtree

Este método, que ha sido definido en un espacio de dos dimensiones, puede ser generalizado a varias. En el caso de tres dimensiones se habla de descomposición "octree", debido a que el árbol pasa a ser de grado 8. Por ejemplo, si el escenario está contenido en un cubo, se subdivide utilizando planos paralelos a cada cara y que lo corten por la mitad. De este modo se obtienen ocho cubos del mismo tamaño que descienden del primero. El resto del procedimiento es similar al de dos dimensiones.

La definición de estos métodos surgió en el campo de modelado de sólidos y visión por computador (Jackins y Tanimoto, 1980; Samet, 1980). Su uso en tareas de planificación fue posterior (Hayward, 1986; Herman, 1986). Incluso se ha utilizado el tiempo como una dimensión más para hallar trayectorias en escenarios dinámicos (Fujimura y Samet, 1989).

Existen otros métodos que directamente parten de una descomposición en celdas del mismo tamaño, es decir, utilizan una discretización regular del espacio de configuraciones. Uno de estos métodos es conocido como "*fast marching method*" (Sethian, 1996). Se basa en descubrir la trayectoria de un rayo de luz emitido desde la posición origen hasta la de destino, aplicando la teoría de ondas. Para ello se define un frente de onda mediante una función $\Phi(x,y,t)$, que corresponde a una curva cerrada en el plano. Por ejemplo, para un destello luminoso iniciado en el origen, sin obstáculos, esta curva correspondería a un círculo centrado en el origen y cuyo radio aumenta de forma progresiva a medida que transcurre el tiempo. En general, la evolución temporal de esta curva se desarrolla avanzando con una velocidad $F(x,y)$ en dirección normal a la propia curva.

Para resolver este problema de forma numérica, Sethian en Sethian (1996), define la función de nivel $T(x,y)$ que devuelve para un punto (x,y) del plano el instante t para el cual el frente de onda alcanza dicho punto. Así, el frente de onda sigue la ecuación:

$$|\nabla T|F = 1 \qquad (2\text{-}6)$$

Esta ecuación es una expresión de la conocida ecuación *eikonal* y viene a decir que la velocidad de propagación del frente de onda es inversamente proporcional al gradiente de la función de nivel. Para resolverla se utiliza una aproximación numérica a esta ecuación (Sethian, 1996).

$$D_{ij}^{+x}T = (T_{i+1,j} - T_{i,j})/\Delta x \qquad D_{ij}^{-x}T = (T_{i,j} - T_{i-1,j})/\Delta x$$

$$D_{ij}^{+y}T = (T_{i,j+1} - T_{i,j})/\Delta y \qquad D_{ij}^{-y}T = (T_{i,j} - T_{i,j-1})/\Delta y$$

$$max\left(max\left(D_{ij}^{-x}T, 0\right), -min\left(D_{ij}^{+x}T, 0\right)\right)^2 + \\ + max\left(max\left(D_{ij}^{-y}T, 0\right), -min\left(D_{ij}^{+y}T, 0\right)\right)^2 = \frac{1}{F_{ij}^2} \qquad (2\text{-}7)$$

donde $T_{i,j}$=*T(i,j)*, siendo *i* el elemento *i*-ésimo de la discretización del eje *x,* y *j* el correspondiente al eje *y.* Utilizando esta aproximación numérica se puede establecer el siguiente método (fig. 2.7).

1. Inicializa

celdas tras la onda=0 (lista A)

celdas en el frente=valores de T(i,j) iniciales (lista B)

resto de celdas=infinito (lista C)

2. Avanza

1. Extrae de B la celda de menor valor cel_{ij} *y agrégala en A*

2. vecinos=$\{cel_{i-1,j}, cel_{i+1,j}, cel_{i,j-1}, cel_{i,j+1}\}$

3. Si algún vecino está en C, trasládalo a la lista B.

4. Recalcula los vecinos que estén en B usando (2-7)

5. Vuelve a 1.

Figura 2.7: Método de frente de ondas.

Como puede observarse se utilizan tres listas, una para las celdas ya calculadas (lista A), otra para las celdas que están en el frente (lista B), y otra para las demás. El algoritmo calcula *T* mediante el avance del frente de onda, celda por celda. Cuando el algoritmo finaliza, se halla T_{ij} para todas las celdas del escenario.

No obstante, a efectos del uso de este método como planificador, resulta innecesario calcular todas las celdas. En el instante en que la celda que contiene la

posición destino es alcanzada por el frente de onda, el algoritmo se detiene. A continuación, utilizando el vector gradiente sobre la función *T* se halla la secuencia de celdas que lleva al origen.

La solución de este método se aproxima a la que ofrecen los grafos de visibilidad, siendo el nivel de discretización el responsable de dicha proximidad. Este tipo de soluciones suele discurrir pegada a los obstáculos, lo cual no es deseable a efectos prácticos. Sin embargo, este método ha sido combinado con otros ofreciendo soluciones de mayor calidad. Por ejemplo, la función de velocidad *F* puede relacionarse con la cercanía a los obstáculos (Garrido y otros, 2007; Garrido y otros, 2008). Para ello se calcula un diagrama de Voronoi previo, asignando en el proceso a cada celda su distancia al obstáculo más cercano. Usando estos valores como índices de refracción (inversa de la velocidad de propagación *F*), se ejecuta el algoritmo (fig. 2.7). El efecto es similar al que se produce en la fibra óptica de índice gradual, donde los rayos de luz se curvan evitando los límites del canal. Así, las trayectorias obtenidas mediante este método se alejan de los obstáculos y resultan además bastante suaves, facilitando su implementación en un algoritmo de control.

2.5. Campos de potencial

Este método trata al robot representado como un punto en C_{free} como una partícula sometida a diferentes fuerzas (Latombe, 1991). Por un lado existe una fuerza de atracción hacia la configuración destino, y por otro una serie de fuerzas de repulsión destinadas evitar la colisión con los obstáculos. Para definir estas fuerzas se considera un campo de potencial asociado al escenario, cuyo valor depende de su proximidad a los obstáculos y de la distancia a la configuración destino (ver figura 2.8).

El planificador genera una trayectoria en virtud de un vector gradiente derivado del campo potencial. Se calcula dicho vector en el origen y se agrega a la trayectoria un segmento de tamaño fijo en la dirección marcada por el gradiente. En el extremo de dicho segmento (final de la trayectoria actual), se vuelve a calcular el gradiente. Nuevamente se añade otro segmento en la dirección del último vector gradiente calculado. De este modo se procede iterativamente construyendo una trayectoria o bien guiando al robot continuamente según el sentido del vector gradiente.

Esta técnica permite que el robot 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.

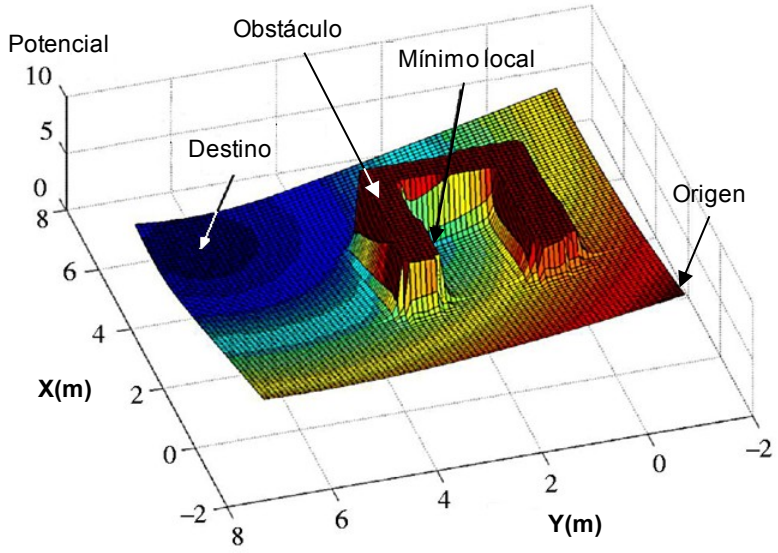


Figura 2.8: Campo de potenciales creado por un entorno dado.

La definición de este campo de potencial (*U*) puede realizarse de diversas formas. Una de ellas consiste en diseñar el campo como la suma de dos: uno que genere una fuerza de atracción sobre la configuración destino (U_{atr}), y otro que genere fuerzas de repulsión en torno a los obstáculos (U_{rep}). Así el campo de potencial en un punto $q \in C_{free}$ resulta:

$$U(q) = U_{atr}(q) + U_{rep}(q) \qquad (2\text{-}8)$$

El campo de potenciales responsable de la fuerza de atracción suele definirse como una función parabólica:

$$U_{atr} = k\rho^2_{goal}(q) \qquad (2\text{-}9)$$

Donde k es un factor de escala, q_{goal} es la configuración destino y $\rho_{goal}(q)$ es la distancia euclídea $\|q - q_{goal}\|$.

La fuerza de atracción derivada de este campo de potencial será (Latombe, 1991):

$$\vec{F}_{atr} = -\vec{\nabla} U_{atr}(q) = -2k \cdot (q - q_{goal}) \qquad (2\text{-}10)$$

Como puede observarse la fuerza de atracción consiste en un vector cuyo módulo es proporcional a la distancia entre el punto considerado y el objetivo, y cuya dirección apunta siempre hacia la configuración final. En cuanto al campo de repulsión suele definirse como la suma de los campos generados por cada obstáculo:

$$U_{rep}(q) = \sum_{k=1}^{r} U_{CB_k}(q) \qquad (2\text{-}11)$$

Un ejemplo de campo de potencial de repulsión asociado a un obstáculo podría ser la siguiente (Latombe, 1991):

$$U_{CB_k}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_k(q)} - \frac{1}{\rho_0}\right)^2 & si \quad \rho_k(q) \le \rho_0 \\ 0 & si \quad \rho_k(q) > \rho_0 \end{cases} \qquad (2\text{-}12)$$

Donde η es un factor de escala positivo, $\rho_k(q)$ resulta ser la distancia de q al obstáculo-C identificado con el valor k. Por ejemplo:

$$\rho_k(q) = \min_{q' \in CB_k} \|q - q'\| \qquad (2\text{-}13)$$

Por último ρ_0 es una constante positiva denominada “distancia de influencia del obstáculo-C”. Así el campo de potencial repulsivo asociado al obstáculo-C correspondiente es siempre positivo o nulo, tiende a infinito conforme se acerca al obstáculo-C y se hace nulo más allá de ρ_0. La fuerza de repulsión asociada a cada obstáculo resulta pues (Latombe, 1991):

$$\vec{F}_{CB_k}(q) = -\vec{\nabla} U_{CB_k}(q) = \begin{cases} \eta\left(\frac{1}{\rho_k(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho_k^2(q)}\vec{\nabla}\rho_k(q) & si \quad \rho_k(q) \le \rho_0 \\ 0 & si \quad \rho_k(q) > \rho_0 \end{cases} \qquad (2\text{-}14)$$

El conjunto de todas estas fuerzas de repulsión sería:

$$\vec{F}_{rep}(q) = \sum_{k=1}^{r} \vec{F}_{CB_k}(q) \qquad (2\text{-}15)$$

Y por tanto el vehículo se moverá en función de la siguiente fuerza resultante:

$$\vec{F}_{res} = \vec{F}_{rep} + \vec{F}_{atr} \qquad (2\text{-}16)$$

El planificador por tanto generará una trayectoria propia de una partícula sometida a $\boldsymbol{F_{res}}$. Sin embargo, el campo resultante puede tener una serie de mínimos locales (véanse las figuras 2.8 y 2.9). El problema aparece cuando el sistema alcanza un mínimo local. En este caso la resultante de todas las fuerzas es nula, pero la configuración no es la que se desea alcanzar. 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 consiste en crear una sucesión de desplazamientos aleatorios. En éstos, cada coordenada puede experimentar un incremento positivo o negativo de una longitud constante dada. Seguidamente, se comprueba la existencia o no de colisión y en función de ello se agrega al camino. Este proceso se repite un número de iteraciones determinado y, a partir del último punto, se vuelve al algoritmo de campo de potencial original.

En la figura 2.9 se ha representado esquemáticamente este proceso, donde las curvas grises representan el valor del campo de potenciales para el conjunto de configuraciones que definen un mínimo, las líneas quebradas el camino creado por los desplazamientos aleatorios, y las rectas largas el producido al seguir el gradiente de potencial.

Como puede observarse, este método presenta algunos inconvenientes. El principal consiste en estimar el tiempo empleado (o el número de iteraciones asignadas) en el desarrollo del camino aleatorio. Éste debe ser lo suficientemente largo como para abandonar cualquier mínimo local, pero no tanto como para consumir excesivo tiempo de cómputo. La solución adoptada consiste en calcular dicho tiempo a su vez con una función aleatoria. Su valor máximo será aquel que posibilite recorrer la mayor longitud rectilínea (en el espacio de configuraciones) contenida en el escenario dado. De este

modo se mantiene la capacidad de escape de cualquier mínimo, al tiempo que se asignan esfuerzos de generación aleatoria más pequeños y habitualmente suficientes para resolver los mínimos más probables.

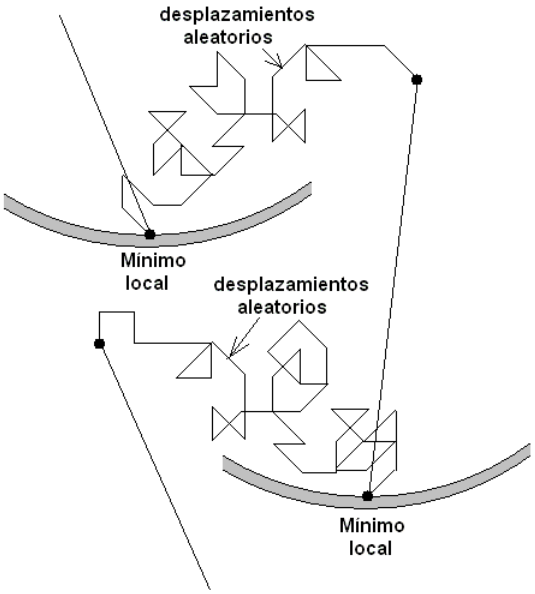


Figura 2.9: Caminos aleatorios.

Otro de los inconvenientes se deriva de la misma naturaleza de los caminos aleatorios particularmente aquí definidos. Son muy sencillos de calcular, pero por el contrario generan una densidad de probabilidad no uniforme.

Por ejemplo, si el espacio de configuraciones C es de dimensión n, la configuración $q \in C$ tendrá n componentes ($q=(q_1,q_2,..,q_i,..q_n)$). Si en cada unidad de tiempo t la componente q_i experimenta un incremento finito de valor $-\varepsilon$ o $+\varepsilon$, entonces, dado un tiempo t (discreto), el valor de esta variable responderá a una función de probabilidad p_i, cuyo valor es (Latombe, 1991):

$$p_i(q_i) = \frac{1}{\varepsilon\sqrt{2\pi t}} \cdot \exp(-\frac{q_i^2}{2t\varepsilon^2}) \qquad (2\text{-}17)$$

La gráfica de la figura 2.10 representa el valor de p_i en función de q_i para diferentes iteraciones ($t \in [10,100]$). Tal y como se puede observar en dicha gráfica, existe una alta probabilidad de permanecer junto al punto de partida, intención opuesta a la

deseada, que sería explorar el espacio circundante para abandonar el mínimo. Es decir, la función de probabilidad ideal consistiría en una recta horizontal. Más adelante se verá que ésta es una de las mejores cualidades del RRT.

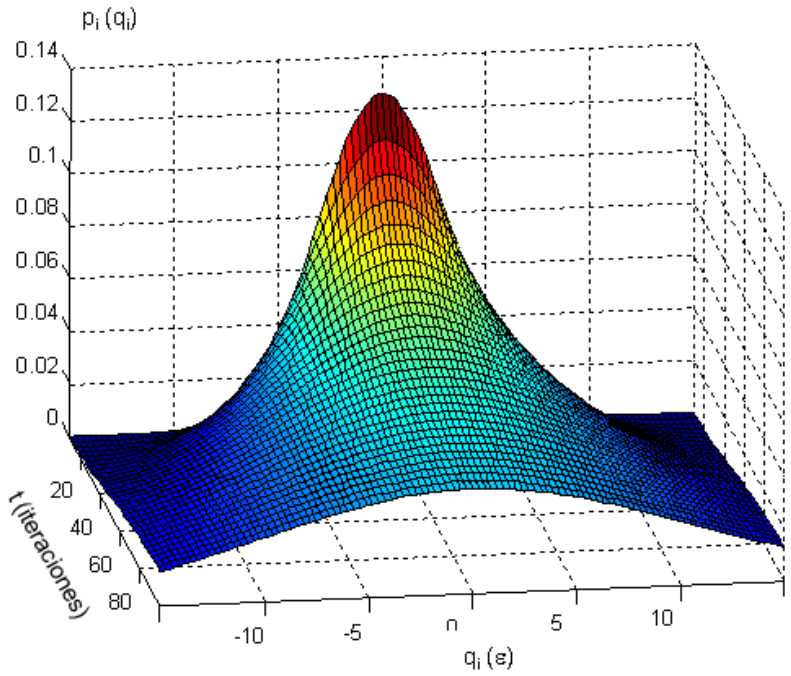


Figura 2.10: Función de probabilidad $p_i(q_i)$.

Lo expuesto hasta ahora muestra un método basado en expresiones analíticas. Sin embargo, el uso de una rejilla regular ("*grid*") para descomponer el escenario en celdas, facilita la implementación del método (Barraquand y Latombe, 1989). Estas celdas tendrán asignado un valor de campo de potencial. En lugar de gradiente y segmento de avance, basta con elegir la celda de menor valor de entre las que rodean a una dada, comenzando a partir de la que contiene a la configuración origen. De este modo se obtiene una secuencia continua de celdas que marcarán la trayectoria a seguir.

2.6 Mapas probabilísticos (PRMs)

Otro de los algoritmos de planificación de actualidad se conoce como *"Probabilistic roadmap"*, mapas probabilísticos o PRMs. Una de sus principales virtudes consiste en su eficacia en el cálculo de trayectorias de robots con muchos grados de

libertad. Existen dos tipos fundamentales de mapas probabilísticos: Los PRM de consulta única y los de múltiple consulta.

En los PRM de múltiple consulta pueden coexistir varios grafos, estos métodos inician el proceso de planificación generando múltiples puntos aleatorios dentro de C_{free}. Después, a partir de éstos, se ejecuta un algoritmo de planificación local entre pares de puntos próximos entre sí. Si la maniobra devuelta presenta alguna colisión se desecha, si no, se incorpora a un grafo que representa todos los puntos entre los que se ha logrado conectividad. Por último, cuando el grafo contiene un camino entre la configuración inicial y la final se obtiene la trayectoria (ver figura 2.11).

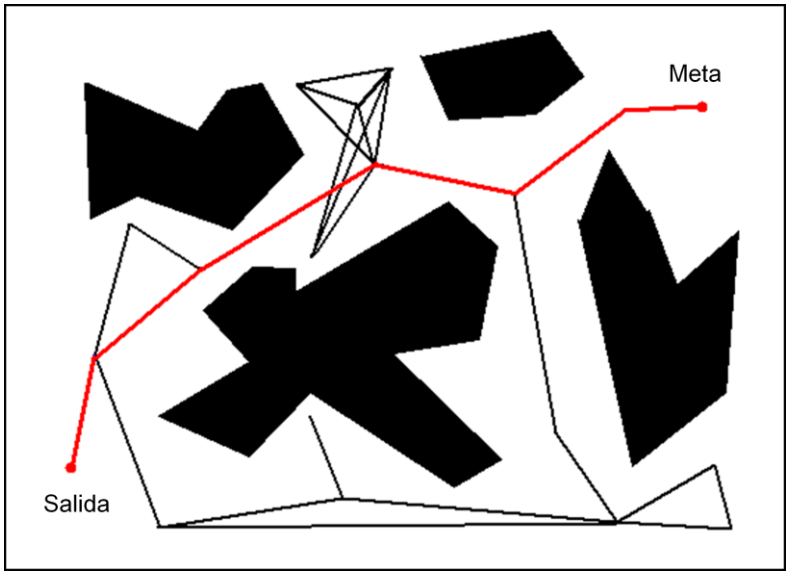


Figura 2.11: Grafo generado por un PRM de múltiple consulta.

La principal ventaja de los PRM reside en que no es necesario calcular los límites de los C-obstáculos, y por ende de C_{free}, tarea nada fácil cuando el número de dimensiones del problema es elevado; sino que basta con utilizar un algoritmo que compruebe si existe o no colisión. En la mayoría de los casos esto resulta bastante más práctico. Otra de las propiedades que hacen atractivo este método se denomina "completitud probabilística" y consiste en que la probabilidad de que un PRM encuentre un camino, si es que existe, tiende a uno exponencialmente con el número de nodos agregados (Kavraki y otros, 1996a).

Un planificador de consulta única mantiene un único grafo (o dos como mucho), y restringe la búsqueda a aquellas configuraciones que sean accesibles desde los nodos

de este grafo (o par de grafos). Esto puede hacerse construyendo un árbol de configuraciones (grafo en el que el planificador local va agregando nuevos nodos y arcos computando trayectorias en direcciones aleatorias) desde la configuración inicial o la final hasta que se puede establecer una conexión con la otra configuración.

También pueden construirse dos árboles simultáneos, cada uno de los cuales tendrá como raíz a una de las configuraciones que se desean unir con una trayectoria, y cuyo crecimiento se detendrá en el momento en que sea posible establecer una conexión entre los dos árboles.

En el primer caso (árbol único) se dice que se hace una búsqueda unidireccional, mientras que en el segundo caso (dos árboles) se trata de una búsqueda bidireccional.

Los PRM tienen ciertas ventajas e inconvenientes con respecto a los campos de potencial. Una virtud evidente es que no hay que resolver problemas de mínimos locales, de hecho, ésta es una de las causas por las que a veces este método es preferido con respecto al anterior. La desventaja más ostensible se centra en el proceso del planificador local. Éste ha de ejecutarse con elevada frecuencia y su coste computacional no es leve (mucho menos si ha de atenderse algún tipo de restricción cinemática o dinámica). Además, muchos de los arcos del grafo no aportan nada a la solución, con lo que su resolución agrega tiempo superfluo al proceso.

Como resultado los PRM son planificadores relativamente lentos pero con ellos se obtiene un grafo que proporciona más información sobre el entorno (varias trayectorias entre los puntos origen y destino además de acceso a otras regiones del plano) que otros métodos.

2.7 Rapidly Exploring Random Tree

2.7.1 Introducción

Steven M. LaValle en 1998 publicó el primer artículo en el que se describe la forma básica de un algoritmo de planificación bastante prometedor bautizado con el nombre de “Rapidly Exploring Random Tree”, abreviadamente RRT (LaValle, 1998).

Inicialmente el RRT se ideó tanto como planificador independiente, como estructura algorítmica a utilizar en otros métodos que precisan algún tipo de proceso de exploración aleatoria (mínimos locales en campos de potencial, y PRMs). Inmediatamente surgieron nuevos artículos poniendo de manifiesto las virtudes del RRT como método de planificación autónomo. En 1999 el autor del método y J. Kuffner publican una adaptación del método para sistemas con restricciones cinemáticas y dinámicas (LaValle y Kuffner, 1999). Al año siguiente, los mismos autores presentaron una mejora que consigue hacer más eficaz al método (Kuffner y LaValle, 2000). Otros autores también han publicado diferentes adaptaciones para los distintos escenarios en los que se aplicaba (LaValle y Kuffner, 2000), e implementaciones prácticas (Xiaoshan, 2005), sus resultados competitivos (Bruce y Veloso, 2002), y evaluaciones sobre su eficacia (Ranganathan, 2003). El RRT tiene por tanto diferentes formas (LaValle, 2006), entre las que se puede distinguir la básica (la original descrita en LaValle, (1998)) y sus adaptaciones (LaValle y Kuffner, 1999; Kuffner y LaValle, 2000; LaValle y Kuffner, 2000; Xiaoshan, 2005). Entre las aportaciones realizadas en esta tesis, también se incluyen ciertas modificaciones (López y otros, 2006; Gómez-Bravo y otros, 2007a; Gómez-Bravo y otros, 2008; Gómez-Bravo y otros, 2009; Martín y otros; 2010), que permiten mejorar los resultados, la aplicabilidad y el tiempo de cómputo. En las siguientes secciones se describen las adaptaciones más relevantes.

2.7.2 Algoritmo RRT básico

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

ρ = 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 final que se desea alcanzar.

q_{rand} = Es una configuración aleatoria que genera el algoritmo dentro del espacio de configuraciones ($q_{rand} \in C$).

q_{near} = Es la configuración más próxima a $q_{rand,}$ en el sentido definido por ρ, de entre las existentes en un árbol.

q_{new} = Es la configuración que se va a añadir al árbol.

ε = 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* consiste en construir un árbol de exploración que cubra uniformemente todo el espacio de configuraciones libres de colisión (C_{free}). Para ello, se desarrolló el algoritmo que se muestra en la figura 2.12 (LaValle, 1998). Dicho algoritmo tiene como misión seleccionar un punto (q_{rand}) de forma aleatoria y extender hacia él el árbol de configuraciones.

```
Algoritmo_basico RRT( q_ini)
    Arbol[0]= q_ini
    Para k = 1 hasta Kmax
        q_rand= Configuración_Aleatoria();
        Extiende(Arbol, q_rand);
    Siguiente k
    Devuelve Arbol
```

Figura 2.12: Algoritmo RRT básico.

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}. El esquema de dicha función se presenta en la figura 2.13.

El algoritmo comienza inicializando la tabla asociada al árbol con la configuración origen. Seguidamente, 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 detener 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.).

```
Extiende(Arbol, q_rand)
    q_near= VecinoMásPróximo(q_rand, Arbol);
        Si NuevaConfiguración(q_rand, q_near, q_new)
    Entonces
        AñadeVértice(Arbol,qnew);
        Si q_new=q_rand Entonces
            Devuelve "alcanzado"
        Si no
            Devuelve "avanzado"
    Si no
        Devuelve "rechazado"
```

Figura 2.13: Función Extiende.

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

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 ρ 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 2.14). 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.

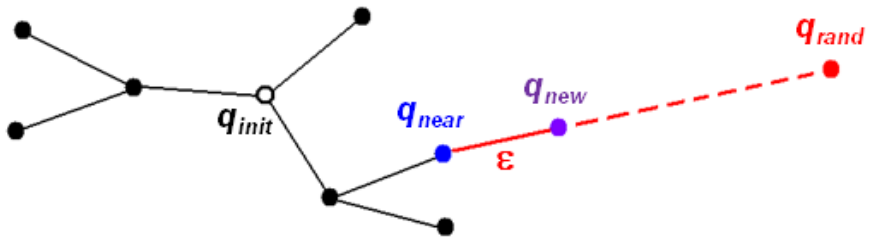


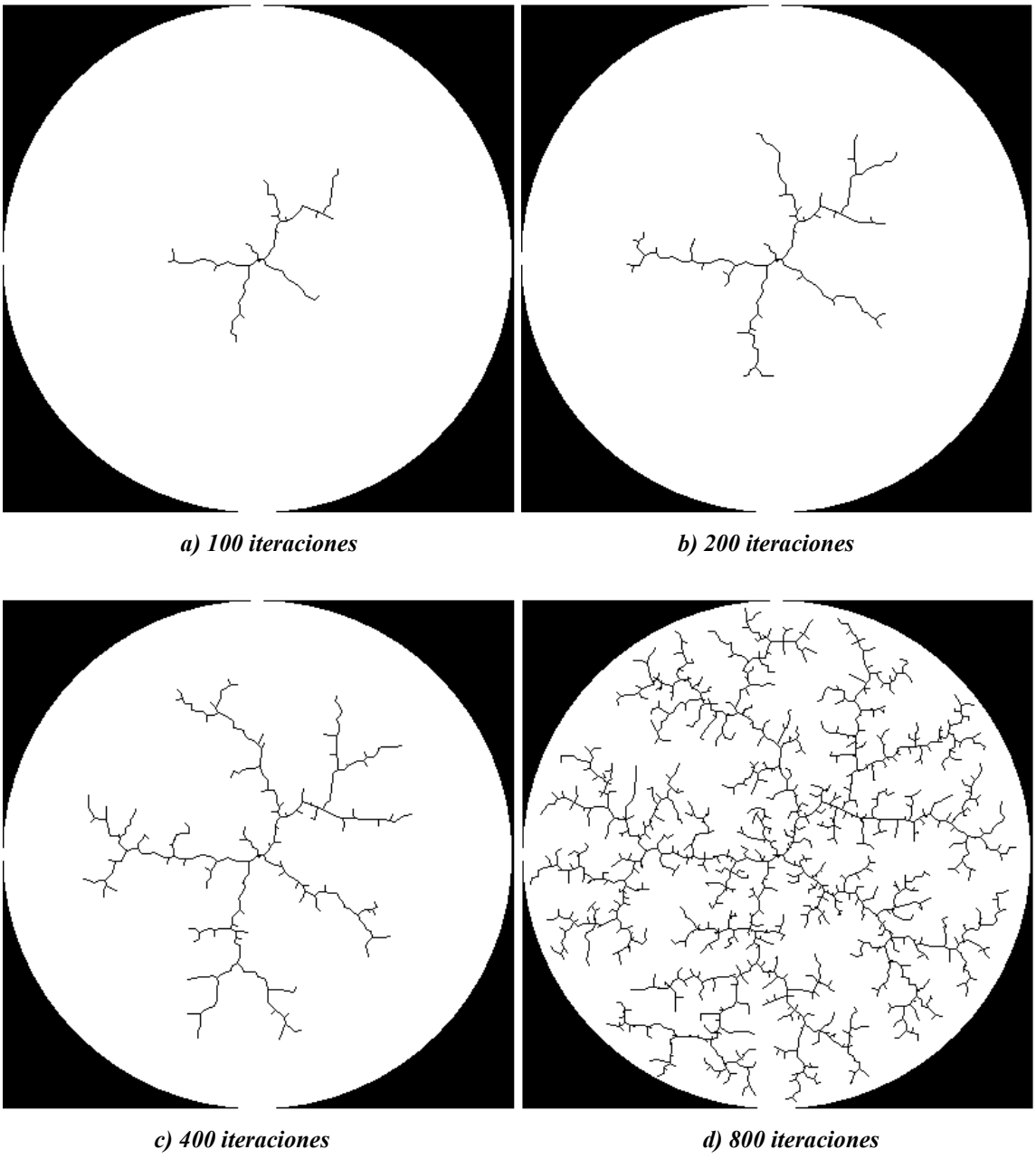
Figura 2.14: Esquema de expansión del RRT.

Si no se ha detectado colisión, se agrega el nuevo punto al árbol distinguiendo entre dos casos. Si el punto aleatorio pertenecía a un círculo de centro q_{near} y radio ε, entonces q_{new} coincidirá con q_{rand} y, por tanto, dicho punto ha sido "*alcanzado*". El algoritmo notificará entonces tal circunstancia. Si por el contrario (caso más general), 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 que une q_{near} con 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 exige al principio avanzar con más avidez hacia zonas inexploradas, pues es allí donde hay más posibilidad de que se defina q_{rand}. Esto puede comprenderse observando el crecimiento del árbol en distintos entornos. Obsérvese, por ejemplo, la progresión del algoritmo representado a través de la figura 2.15.

En concreto, para que un punto determinado del árbol agregue una rama, es necesario que q_{rand} quede más próximo a éste que a ningún otro. Esto equivale a decir que q_{rand} pertenezca a la región de Voronoi asociada a dicho punto. Si q_{rand} se genera de forma equiprobable en todo el espacio de configuraciones, y éste se divide en regiones de Voronoi, la probabilidad de que q_{rand} pertenezca a una de estas regiones será igual al cociente entre el tamaño de la región y el del espacio de configuraciones.

En el caso de la fig. 2.15, 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.



a) 100 iteraciones *b) 200 iteraciones*

c) 400 iteraciones *d) 800 iteraciones*

Figura 2.15: Evolución del Algoritmo RRT en un entorno circular.

Otro hecho a observar es la inexistencia de una alta densidad de ramas en el origen. La naturaleza del RRT evita precisamente esto (que es lo que ocurría en el método de escape de mínimos locales en los campos de potencial). Es más, al principio, las regiones de Voronoi de los puntos periféricos del árbol son las más extensas, por lo que el crecimiento se invierte en la prolongación de unas pocas ramas. Conforme el árbol va llenando el espacio libre, este proceso decae, y las ramas aumentan su número

emergiendo por doquier. Se puede contemplar que el árbol tiende a lograr una densidad uniforme de ramas sobre el espacio libre.

Obsérvese ahora las figuras 2.16a, b, y c. El escenario en este caso es cuadrado, y se muestra tanto el árbol RRT como las regiones de Voronoi asociadas a sus puntos. En este caso las ramas tienden a crecer hacia las esquinas, dado que en estas direcciones hay más configuraciones que en otras, y ello se traduce en una mayor probabilidad de establecer q_{rand}.

En las fig. 2.16d, e, f y g se representa la progresión del algoritmo RRT sobre un espacio donde el desequilibrio en regiones de Voronoi es más acusado. Para ello se sitúa el origen del árbol RRT a un lado del escenario. En la evolución se aprecia 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 a la derecha del origen del árbol. Estos puntos dirigen el crecimiento, en detrimento de otras ramas anexas a dicho punto origen. En este caso se ha utilizado una variante en la que $q_{\text{rand}} \in C_{free}$, con lo que este comportamiento resulta más acusado.

Al igual que en el caso de la figura 2.15, tras las suficientes iteraciones, puede comprobarse que la densidad de ramas resulta razonablemente homogénea a pesar del desequilibrio inicial existente entre los dos rectángulos.

A continuación se expone una lista de las características de los algoritmos basados en RRT (LaValle, 1998), cuyos dos primeros elementos se acaban de explicar:

- La expansión de los algoritmos basados en RRT se inclina decididamente hacia los espacios inexplorados.

- La distribución de sus vértices aproxima la de la función de probabilidad utilizada.

- Son simples, facilitando el análisis de comportamiento y la implementación en cualquier escenario.

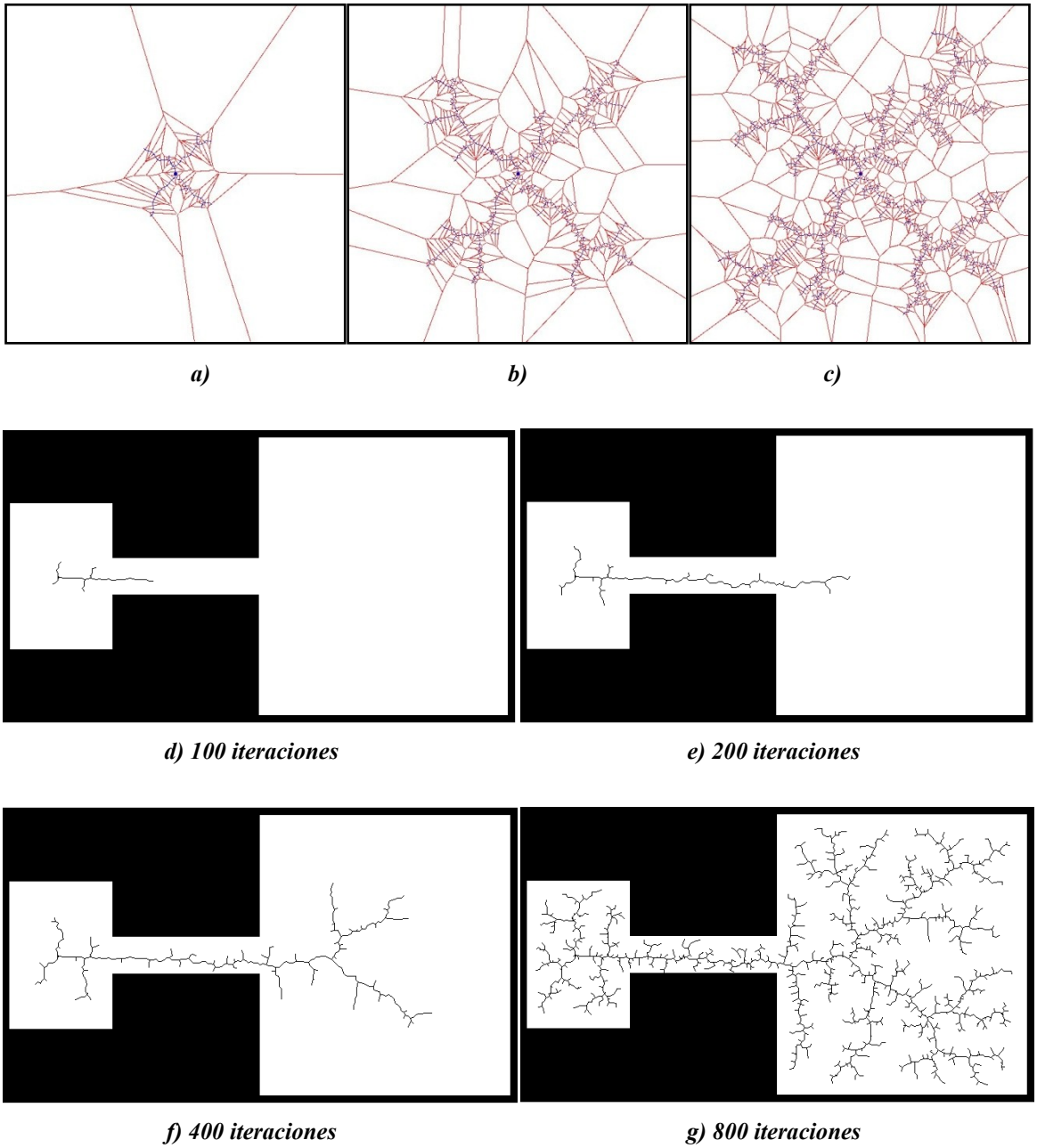
- Siempre permanecen conexos, aún con pocos vértices.

- Un RRT es un módulo que puede ser implementado en otros planificadores.

- No requieren la existencia del par origen-destino, con lo que su ámbito de aplicación se diversifica.

- No requieren una definición explícita de C_{free}, sólo utiliza una función que comprueba la existencia de colisión.

- Es probabilísticamente completo (la probabilidad de encontrar un camino si existe tiende a 1 exponencialmente con el número de nodos).



a) *b)* *c)*

d) 100 iteraciones *e) 200 iteraciones*

f) 400 iteraciones *g) 800 iteraciones*

Figura 2.16: Evolución del RRT en un entorno asimétrico.

Por último, se advierte que los resultados indicadores de la función *Extiende* no se utilizan en este algoritmo, que sólo se ocupa de generar un árbol capaz de explorar de modo equiprobable el espacio libre. Esta peculiaridad atiende al diseño inicial del RRT como estructura de datos a implementar sobre otros algoritmos de planificación (campos de potencial y PRMs). Sin embargo, en una primera aproximación del RRT como planificador autónomo, surgieron los algoritmos RRT-GoalZoom y RRT-GoalBias (LaValle y Kuffner, 2000), que utilizan este algoritmo RRT básico con pequeñas modificaciones en la función de probabilidad generadora de q_{rand}. En las siguientes secciones se detallan algunas de estas extensiones así como otras mejoras del RRT que permiten establecer un camino entre configuración origen y destino.

2.7.3 Algoritmos RRT de consulta única

Las primeras aplicaciones del RRT como planificador autónomo consisten en utilizar la generación de un árbol RRT en torno al punto de partida. Con las sucesivas iteraciones el árbol crece ocupando el espacio vacío del escenario, aproximando eventualmente sus vértices cada vez más a la configuración final deseada. Dado un número de iteraciones lo suficientemente elevado alguna de las ramas agregadas se acercará al objetivo lo suficiente como para obtener la trayectoria deseada. Sin embargo las prestaciones de este algoritmo dejan mucho que desear, puesto que resulta demasiado lento.

Una de las mejoras que se pueden introducir para acelerar el proceso consiste en sustituir q_{rand} por q_{fin} una fracción de las iteraciones, o mejor, asignar una función de probabilidad que sortee el candidato a q_{rand}. Así, unas veces q_{rand} será un punto aleatorio y otras el mismo punto objetivo. De esta forma queda definido el “RRT-GoalBias” (LaValle y Kuffner, 2000). Este algoritmo, incluso con muy baja probabilidad asignada al punto objetivo (0.05), resulta mucho más rápido que el RRT sin modificar. Este parámetro se revela determinante en el comportamiento del planificador, pues si es demasiado elevado actúa de forma muy parecida al de campos de potencial cayendo con facilidad en mínimos locales.

Para evitar tal contingencia surgió la idea de no imponer exactamente la configuración final, sino puntos próximos a esta. Así la función de probabilidad anterior

queda definida de la siguiente forma: se asigna una probabilidad *p* a que q_{rand} sea un punto aleatorio del espacio de configuraciones, y una probabilidad (*1-p*) a que sea un punto que diste de la configuración final un valor inferior a R_{max}. Así, en vez de un punto, se tiene una esfera en el espacio de las configuraciones capaz de atraer el crecimiento del árbol. El radio de dicha esfera (R_{max}) se determina en cada iteración siendo la distancia entre el punto objetivo y el vértice del árbol más próximo al mismo. El efecto es que conforme el árbol crece los puntos aleatorios surgen cada vez más cerca del objetivo, mejorando el rendimiento del algoritmo. Este planificador es conocido como "RRT-GoalZoom" y aunque resulta mejor que el anterior, no evita del todo caer en mínimos locales.

2.7.4 Algoritmo RRT-Bidirectional

La siguiente adaptación del algoritmo *RRT* para conseguir conectar una configuración inicial con una final ofrece mejores resultados que los algoritmos anteriores. El denominado *RRT_Bidirectional* (LaValle y Kuffner, 2000) queda descrito en la figura 2.17. Éste y los siguientes se basan en la generación simultánea de dos árboles cuyas raíces son las configuraciones inicial y final. Dichos árboles crecen explorando el espacio vacío y buscándose entre sí hasta conectar, obteniendo entonces una trayectoria como resultado. El método utilizado para ello varía, dando lugar a distintos algoritmos.

Como se puede comprobar este algoritmo comienza con la generación de dos árboles que parten de los puntos origen y destino, tal y como se acaba de comentar. El algoritmo concluye en cuanto dichos árboles conectan. Si alcanzado el valor de *Kmax* ambos árboles no han coincidido, se devuelve un mensaje de error.

En cada iteración uno de los árboles agregará una nueva rama si es posible en dirección al punto aleatorio generado. De esta forma dicho árbol tendrá una función de exploración del espacio vacío, dado que q_{rand} es una configuración que puede surgir en cualquier punto del escenario, marcando una dirección de crecimiento para el nodo más próximo.

```
RRT_Bidirectional (q_ini, q_fin)
    Arbol_a[0]= q_ini
    Arbol_b[0]= q_fin
    Para k = 1 hasta Kmax
        q_rand= Configuración_Aleatoria( );
        Si Extiende(Arbol_a, q_rand)≠ "rechazado" Entonces
            Si (Extiende(Arbol_b,q_new)="alcanzado" Entonces
                Devuelve Camino(Arbol_a, Arbol_b)
        Intercambiar(Arbol_a, Arbol_b)
    Siguiente k
    Devuleve Error
```

Figura 2.17: Algoritmo RRT-Bidirectional.

Si ha habido una nueva rama, entonces existe un vértice nuevo q_{new}. Ahora le toca el turno al segundo árbol que tomará como meta no el punto aleatorio q_{rand}, sino el nuevo vértice generado q_{new}. De esta forma dicho árbol no explora el espacio vacío, sino que intentará encontrar a su homólogo si es posible.

Obsérvese cómo se hace uso de la función *Intercambiar*, que alterna el orden de los árboles de manera que el reparto de sendas funciones sea equilibrado. Así pues, las acciones resultantes de dos iteraciones consecutivas serían:

1. Árbol A crece hacia q_{rand} (1ª iteración).
2. Árbol B crece hacia q_{new} de A.
3. Árbol B crece hacia q_{rand} (2ª iteración).
4. Árbol A crece hacia q_{new} de B.

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.

El algoritmo RRT-Bidirectional admite ciertas mejoras, como se verá a continuación. Para diferenciarlos de los otros también ha sido bautizado con el sinónimo

de RRT-ExtExt, nomenclatura que responde a cierta lógica como se detalla a continuación.

2.7.5 Algoritmo RRT-ExtCon

En este algoritmo (ver figura 2.18) se sustituye la función "Extiende", que agregaba un nuevo segmento al árbol en la fase de conexión, por otra denominada "Conecta" (ver figura 2.19), más ambiciosa, que va agregando segmentos consecutivos hasta alcanzar la configuración objetivo o, si no es posible, hasta que un obstáculo lo impida.

```
RRT_ExtCon ( q_ini, q_fin)
    Arbol_a[0]= q_ini
    Arbol_b[0]= q_fin
    Para k = 1 hasta Kmax
        qrand= Configuración_Aleatoria( );
        Si no (Conecta(Arbol_a,q_rand )="rechazado") Entonces
            Si Conecta(Arbol_b, q_new) = "alcanzado" Entonces
                Devuelve Camino(Arbol_a, Arbol_b)
        Intercambiar(Arbol_a,Arbol_b)
    Siguiente k
    Devuleve "trayectoria no encontrada
```

Figura 2.18: Algoritmo RRT_ExtCon

Se logra así un esfuerzo de conexión mucho más acusado que en el algoritmo anterior logrando la unión de los árboles en cuanto encuentran una vía sin obstáculos rectilínea entre sus vértices sin menoscabar las funciones de exploración de la fase previa. Esto introduce un mayor tiempo de cómputo en cada iteración, consumido en la

elongación de las nuevas ramas tanto como sea posible. A cambio se ahorran las instrucciones del algoritmo dedicadas a escudriñar el vértice más próximo para cada nuevo segmento.

```
Conecta(Arbol, q)
    Repite
        S=Extiende(Arbol, q);
    Mientras (S="avanzado")
    Devuleve S
```

Figura 2.19: Esquema de la función conecta

Dependiendo del escenario, la alternativa presentada puede mejorar o no los resultados. En la práctica parece observarse un mejor comportamiento de este algoritmo con respecto al "ExtExt" cuando el entorno no está congestionado de obstáculos (Kuffner y LaValle, 2000).

2.7.6 Otras modalidades de RRT

Si la idea anterior consiste en sustituir la función Extiende por la función Conecta en la segunda fase, ahora se plantea la posibilidad de hacerlo en la primera o en ambas. Así surge el algoritmo RRT-ConExt y RRT-ConCon. En el primero las virtudes de la función Conecta se aplican ahora en la primera fase, con lo que la exploración gana protagonismo. Al igual que en el caso anterior, no todos los escenarios son susceptibles de ofrecer mejores resultados con este algoritmo. Por ejemplo, en un escenario con escasos obstáculos donde no sea precisa una amplia exploración antes de que sendos árboles puedan verse, este algoritmo consumiría inútilmente bastante tiempo computacional tratando de crecer y agregando por tanto nuevas e inútiles ramas cuando la conexión ya está "a la vista". En dicha circunstancia resultaría más apropiada la versión anterior del algoritmo. Sin embargo en escenarios más complicados donde sea preciso avanzar más tiempo explorando antes de poder conectar, será éste el que proporcione mayor eficiencia.

Por último, es posible hacer lo mismo que en el algoritmo anterior pero usando la función *Conecta* en ambas fases, con lo cual se obtendría el algoritmo RRT-ConCon. La experiencia (LaValle y Kuffner, 2000) ha demostrado que el RRT-ExtCon resulta especialmente apropiado en problemas holónomos. El RRT-ExtExt en cambio sigue siendo el más eficaz para los no holónomos.

También es posible utilizar varios árboles en lugar de sólo dos. Aquí surge el problema de decidir qué árboles se intentará conectar y cuánto tiempo habrá de utilizarse en la exploración y cuánto en la conexión. Incluso, es posible plantearse la opción de generar un árbol con cada punto aleatorio y ejecutar la función Conecta desde los demás. Resultaría entonces que los mapas probabilísticos (PRMs) serían un caso límite de los algoritmos RRT.

Otro de los aspectos sobre los que se ha trabajado recientemente ha sido en el método de generar los q_{rand}. En lugar de la función aleatoria habitual cuyo único requisito consiste en que todos los puntos de *C* presenten la misma probabilidad de ser elegidos, se ha estudiado introducir una secuencia de q_{rand} con propiedades más atractivas para la planificación (LaValle, 2006). Algunas de estas propiedades son:

-Densidad, relativa al término de densidad de conjuntos. Se dice que una secuencia es densa en *C*, si la clausura (interior más frontera) del conjunto de *n* puntos de la secuencia tiende a *C* con *n*. Un ejemplo de secuencia infinita que cumple con este requisito, y con buenas propiedades para su uso en RRTs, es la secuencia de Van der Corput (1935). La idea que subyace en esta propiedad es que la secuencia cubra en lo posible a *C*. Dado que es imposible en la práctica llegar a todos los elementos de *C* (por tener infinitos puntos), al menos se procura que la secuencia proporcione puntos arbitrariamente cercanos a cualquier punto de *C* lo cual está relacionado con esta propiedad (LaValle, 2006).

-Dispersión, definida como la distancia máxima que puede hallarse entre un punto cualquiera de *C* y el punto de la secuencia más próximo. Gráficamente coincide con el radio del mayor círculo vacío (sin puntos de la secuencia en el interior) que puede hallarse en *C.* Este círculo vacío constituye un área inexplorada, por tanto se desea minimizar el valor de la dispersión, con objeto de disminuir en lo posible estas áreas inexploradas. Para ello, Sukharev definió una secuencia que ubica los puntos en forma de una rejilla regular (Sukharev, 1971) que ofrece el valor óptimo de dispersión.

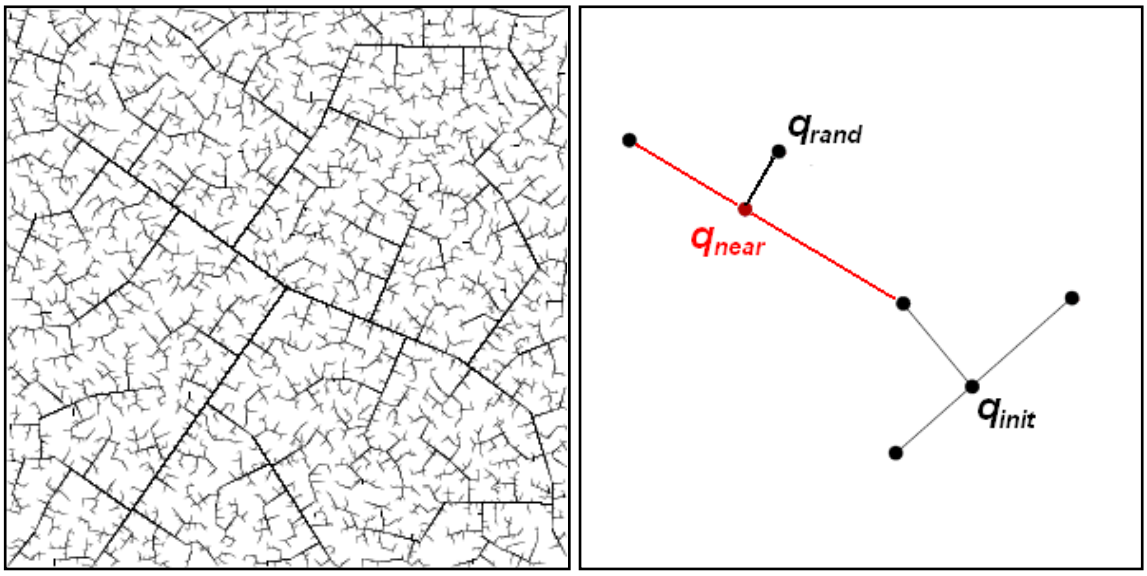
-Discrepancia, está relacionada con el grado de alineación de los puntos de la secuencia. Se desea que no exista tal alineación en lo posible, ya que ello implicaría una seria limitación en cuanto a las direcciones de búsqueda del RRT. Por ejemplo, no es lo mismo que una serie de puntos alineados coincida con un pasillo (C_{free}) o con una pared (CB). El primer caso podría acelerar el algoritmo al dirigir el crecimiento por C_{free}, mientras que el segundo retardarlo. Las secuencias de Halton (1960) y Hammersley (1960) son ejemplos de secuencias orientadas a optimizar el valor de discrepancia.

Además de las secuencias mencionadas con cada propiedad existen otras que cumplen en mayor o menor grado con estas deseables características. Por ejemplo se pueden usar rejillas obtenidas a partir de vectores unitarios no ortogonales (para mejorar la discrepancia). En general, ya sean secuencias densas en C o funciones equiprobables (que también tienden a cubrir C con lo cual son densas en C), el algoritmo pasa a denominarse RDT ("*Rapidly Exploring Dense Trees").*

Otros aspectos diferentes del algoritmo han sido considerados. Por ejemplo, en lugar de buscar sólo en los vértices del grafo más cercanos, sería posible añadir a la búsqueda los puntos contenidos en las propias ramas del grafo. Procediendo de esta forma el aspecto del árbol varía (ver fig. 2.20).

Con éste método las ramas dejan de tener una longitud fija ε, pudiéndose añadir en una única iteración ramas de gran longitud con el reducido coste de información de un único vértice. La desventaja aparece en el proceso de búsqueda del punto más cercano, que se complica. Este proceso ha de incluir las ramas necesariamente o de lo contrario (buscando sólo los vértices más cercanos) podrían agregarse ramas cruzadas.

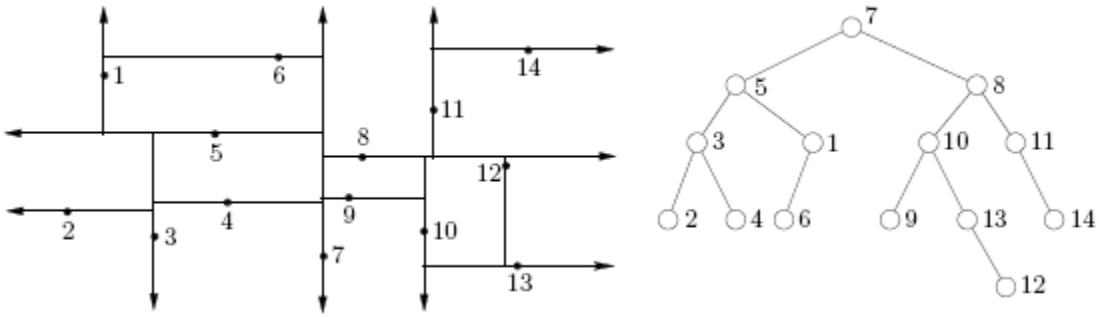
Hay dos formas de acometer el problema. En la primera se resuelve de manera exacta. Es decir, en cada iteración se comprueban todas las ramas del grafo, calculando la distancia de cada una de ellas a la configuración q_{rand}. Este proceso tiene una duración proporcional al número de vértices. En cada iteración pueden agregarse hasta dos vértices a la vez (q_{near} y q_{rand} en la fig. 2.20b), toda vez que el punto más próximo pertenezca al interior de una rama (q_{near}). Este nuevo punto subdivide la rama en dos. Para no elevar excesivamente el tiempo de cómputo, a efectos de cálculo de distancia se mantienen los segmentos originales.



a) Aspecto tras varias iteraciones *b) Incorporación ortogonal de una rama*

Fig. 2.20: Representación del Algoritmo RDT

La otra forma de resolver esta distancia es subdividiendo las ramas en segmentos pequeños de longitud fija. Este sería el caso del RRT original. De hecho, su autor los clasifica como un subtipo de RDT (LaValle, 2006). En este caso la búsqueda del punto más próximo se reduce a los vértices del árbol y no a sus ramas. La desventaja radica en que el número de vértices aumenta enormemente en comparación al método anterior.



a) Ubicación de los vértices *b) Estructura de árbol binario*

Fig. 2.21: Kd-Tree

Una de las soluciones propuestas es utilizar los denominados “Kd-tree” (de Berg y otros, 2000). Éstos consisten en estructuras de tipo árbol binario (fig. 2.21b), en la que

cada nodo representa un vértice del RDT (en la fig. 2.21a se representan los vértices de un RDT sin sus ramas). Para construirla se ordenan todos los vértices por abscisa creciente y se asigna al primer nodo del Kd-tree el vértice medio de esta clasificación (vértice 7 en la fig. 2.21a y nodo raíz con el mismo número en la 2.21b). De este modo la mitad de los vértices quedarán a la izquierda (vértices 1-6) y la otra mitad a la derecha (vértices 8-14). Con cada uno de estos conjuntos se realiza una segunda jerarquización, está vez por ordenada creciente. El vértice medio de cada una (5 en el lado izquierdo y 8 en el derecho) se escoge como nodo descendiente del primero (7). Ahora se tienen cuatro conjuntos: los vértices situados por encima y por debajo del vértice 5, y a la izquierda del 7; y los vértices situados por encima y por debajo del vértice 8 y a la derecha del 7. Estos conjuntos vuelven a ordenarse por abscisa y el proceso continúa hasta que el Kd-tree tiene un nodo por vértice.

Una vez obtenido el Kd-tree, la búsqueda del vértice más cercano a un q_{rand} dado se realiza de una forma más rápida. Se procede a comparar q_{rand} con el nodo raíz del árbol determinando si está a su izquierda o a su derecha. Seguidamente se compara su ordenada con el nodo hijo situado al mismo lado, y así sucesivamente. De este modo se recorre el Kd-tree hasta llegar al último nodo. Una vez recorrido el árbol se calcula la distancia a los vértices de los nodos recorridos, puesto que el más próximo se contará entre éstos.

2.7.7 Postprocesado

A la vista de los árboles obtenidos en la aplicación de los distintos algoritmos RRT, los caminos que se obtienen se alejan de lo que sería el camino más sencillo, a veces dando excesivos rodeos y contorneando los límites de los obstáculos. Además la calidad de estas trayectorias deja mucho que desear, sobre todo desde el punto de vista del control. Ello se debe a que es muy difícil hallar sucesiones de vértices alineados, es decir, una evolución suave de las variables de configuración a lo largo de la trayectoria. En vez de eso el camino resulta difícilmente practicable; tanto más cuanto mayor sea la precisión (menor tamaño de segmento) utilizada en el RRT: su ejecución implicaría ordenar al robot avanzar unos centímetros para luego pararlo y hacerlo girar, en muchas ocasiones, sin necesidad. El coste de la acción se eleva enormemente.

Por consiguiente, en todas las aplicaciones prácticas donde interviene el método RRT resulta conveniente aplicar al camino obtenido una herramienta que permita reducir la irregular topología de los árboles. A estas herramientas, comunes por otro lado a otros planificadores, se las denomina algoritmos de postprocesado (Latombe, 1991) y consisten en algoritmos de simplificación del camino, rápidos y sencillos, que parten de la información suministrada por el planificador y generan de forma iterativa un camino más simple.

En concreto, para la aplicación presentada en este trabajo se propone la función que se muestra en la figura 2.22, basada en un algoritmo de post-procesado conocido (Latombe, 1991). Su efecto se representa en la figura 2.23. 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, *Numero_de_vertices* es una función que devuelve el número de vértices que componen el árbol obtenido e *i* representa el número de vértices que se pretende sustituir. En un principio, el algoritmo prueba a sustituir todo el árbol por un trayectoria recta. Si esto no es posible, *i* va adquiriendo un valor cada vez menor (típicamente se escoge el punto medio del tramo de trayectoria con el que se prueba).

En cada iteración, la función *CuerdaVálida*(1, *j*) establece una ruta recta entre el primer vértice y el *j-ésimo,* generando un conjunto de puntos alternativos y alineados en el espacio de configuración (*C*) sobre los que se comprueba la posibilidad de colisión. Si la ruta presenta colisiones, la función *CuerdaVálida* devolverá falso, y se deja a la siguiente iteración el intento de establecer una nueva ruta recta, pero ahora entre el primer vértice y el punto medio del fragmento de trayectoria con el que se acaba de probar.

Si en una de las iteraciones la ruta es válida, es decir, todas sus configuraciones están incluidas en C_{free}, entonces se da paso a la función *EliminaArco*(1, *j*), que sustrae del árbol original los vértices que van desde el 1 al *j*, disminuyendo, por tanto, el valor que devuelve *Numero_de_vertices* e incorporando la nueva ruta al camino definitivo final. Por tanto, el árbol original queda configurado con un nuevo vértice inicial y el camino definitivo con un nuevo tramo recto.

```
Postprocesado(camino_original)
    fin = falso;
  Mientras no fin
      colision = verdadero;
      i =Numero_de_vertices
    Mientras colision
      Si CuerdaVálida(1, i) Entonces
        EliminaArco(1,i,camino_original,camino_final)
        colision=falso;
      Si no
        i = i/2
      Si i <= 2 Entonces
        EliminaArco(1,2,camino_origi nal,camino_final)
        colision=falso;
    Fin Mientras
    Si Numero_de_vertices<=2 Entonces
      fin=verdadero;
  Fin mientras
  Devuelve camino_final
```

Figura 2.22: Algoritmo de Postprocesado

En la figura 2.23a 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 de la trayectoria solución. Una vez obtenida se acomete el postprocesado.

En la misma figura 2.23b se muestran, sobre el escenario original, las cuerdas utilizadas para simplificar el camino inicial. Las líneas en color rojo muestran las cuerdas que presentan colisión y que, por tanto, no pueden ser utilizadas para ese fin. Por contra,

la línea de trazo azul representa la primera cuerda que permite simplificar el árbol. En concreto, la simplificación del tramo de árbol que abarca desde la configuración inicial A hasta la configuración B.

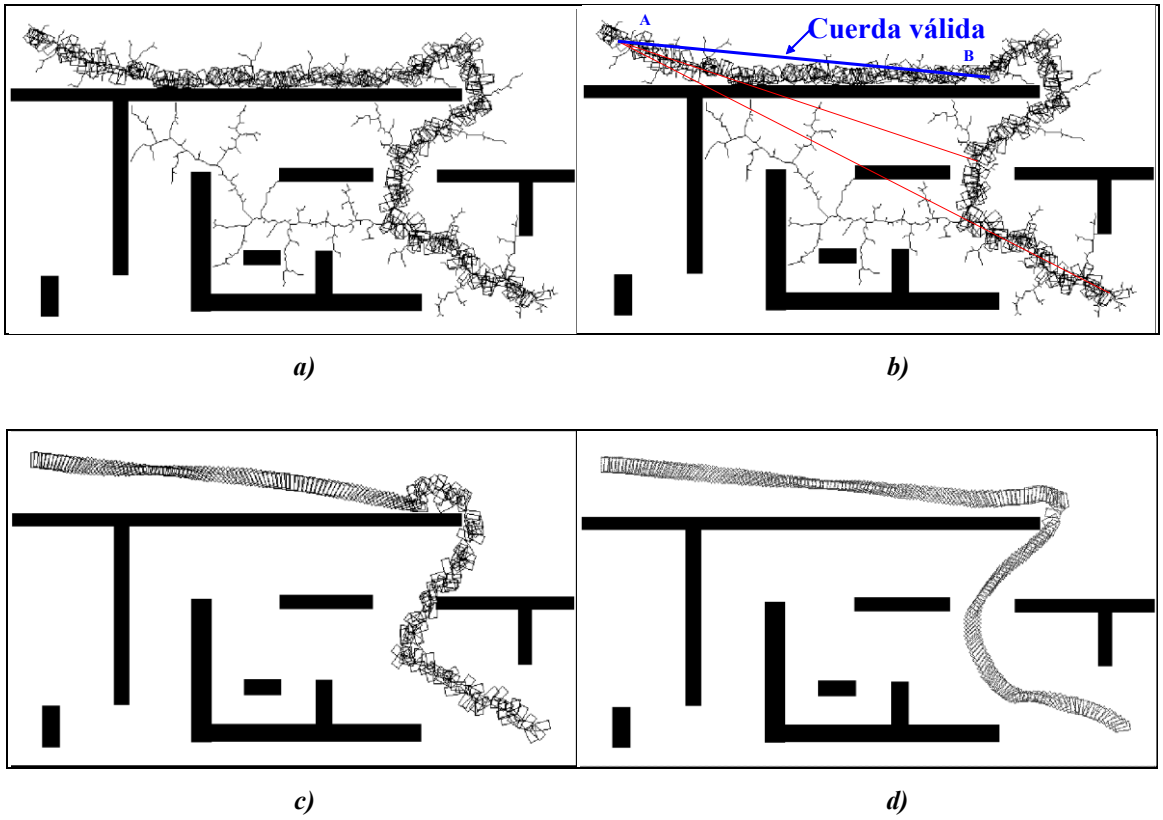


Figura 2.23: Evolución del algoritmo de post-procesado.

La figura 2.23c presenta el resultado de sustituir dicho tramo del árbol original por las configuraciones contenidas en la cuerda anteriormente validada. A partir de este instante, el algoritmo es aplicado al resto del camino original, según lo expuesto a lo largo de esta sección, hasta alcanzar un camino simplificado. Una vez obtenido éste, el algoritmo de postprocesado se vuelve a aplicar para asegurar la obtención de un camino lo suficientemente sencillo.

En el último cuadrante (figura 2.23d) se muestra el camino final una vez se ha completado todo el postprocesado.

2.8. Conclusiones

Entre los diferentes métodos de planificación existen diferencias que los sitúan más o menos adecuados a cada tipo de escenario. La mayoría, como se ha visto, requieren algún tipo de pre-procesamiento que puede limitarse a la definición poligonal de los obstáculos-C (visibilidad, Voronoi, descomposición en celdas), o llegar a la subdivisión en celdas de todo el escenario y el cómputo de un determinado valor en cada una de ellas (campos de potencial). Otros en cambio evitan esta labor previa y permiten atacar de forma directa el problema (PRMs), en cambio tienen un coste computacional más pesado en su proceso. Según los requerimientos del problema, puede ser más interesante un método u otro en función de su velocidad, y la calidad de la solución obtenida.

Entre los algoritmos más rápidos se cuenta el método de campos de potencial, pero conlleva el problema de los mínimos de potencial. Para solventarlo se idearon los métodos de exploración aleatoria. Éstos tuvieron su mayor difusión dentro de los PRMs, que basaban su expansión en un muestreo probabilístico de C_{free}. El éxito de los métodos aleatorios sugirió la posibilidad de usar dichas técnicas de forma exclusiva, eliminando el coste del pre-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 RRT.

El RRT no precisa el establecimiento de un campo de potencial con el consecuente ahorro del pre-procesamiento. Además es inmune al número de vértices que tengan los obstáculos (al contrario que los diagramas de visibilidad) o a la forma y orografía de los mismos (que complica la tarea de elaborar los diagramas de Voronoi), pues tan sólo necesita una función que compruebe la existencia o no de colisión para una configuración concreta. Al mismo tiempo el RRT asegura una exploración equiprobable de todo el espacio de configuraciones, compitiendo en esta ventaja con los PRM al tener un coste computacional menor. Mientras los PRM pueden generar grafos inconexos, el RRT siempre mantiene sus vértices conectados. Por añadidura es sencillo de implementar, rápido y de fácil extensión a sistemas con elevado número de grados de libertad. Tiene características propias como la extensión natural hacia los espacios inexplorados, y la diversificación del algoritmo en diferentes tipos adaptados a cada tipo de escenario.

Por el contrario, los RRT precisan invariablemente de un post-procesado que transforme sus resultados en algo práctico. Además, la adaptación del RRT a sistemas con restricciones no holónomas que había sido propuesta en la bibliografía, resulta comparativamente lenta respecto a otros algoritmos. Estos dos aspectos, han constituido el punto de partida del postprocesado con maniobras restringidas (capítulo 4) y del método VODEC desarrollado en esta tesis y que se presenta en el capítulo siguiente.

CAPÍTULO 3 VODEC

3.1 Introducción

Tal y como se comentó en la introducción, uno de los objetivos de esta tesis ha sido la integración del algoritmo RRT con las maniobras restringidas para la planificación de robots con restricciones no holónomas. Esto se ha llevado a cabo en dos etapas: en la primera el RRT actúa sin restricciones proporcionando una trayectoria, en la segunda un algoritmo de postprocesado incorpora las maniobras restringidas hasta transformar la trayectoria inicial en una admisible para el robot.

El estudio del algoritmo de postprocesado reveló que su comportamiento mejoraba a medida que las trayectorias de la primera etapa se alejaban de los obstáculos. Esto motivó la consideración de otros planificadores en lugar del algoritmo RRT. Se trabajó con varios de los expuestos en el capítulo 2. En concreto hubo dos que merecieron especial atención: los diagramas de Voronoi y la descomposición vertical. El primero, el más obvio, cumple a la perfección con el criterio de alejarse de los

obstáculos, pero resulta lento y complejo. El segundo es comparativamente más sencillo y rápido, pero no es óptimo en cuanto a la separación con respecto a los obstáculos.

Surgió entonces la idea de VODEC (diagrama de VOronoi con Descomposición Exacta de Celdas), un algoritmo que mejoraba los resultados obtenidos en la descomposición vertical en cuanto a proximidad de obstáculos. Por tanto VODEC se diseñó basándose en dicha descomposición. En un principio VODEC pretendía alterar la trayectoria separándola de los obstáculos en el sentido horizontal, ya que en el vertical la naturaleza del método lo procuraba (uso de los puntos medios, es decir, de los puntos equidistantes en el sentido vertical). Para ello, VODEC se apoyó en la estructura de datos creada en la descomposición vertical, denominada celda. VODEC recorre las fronteras de cada celda buscando los obstáculos que la tocan, y en base a éstos corrige la trayectoria alejándola de los mismos. Dado que estas celdas tienen un ancho arbitrario, no resultaba efectivo en celdas delgadas (baja distancia de separación) y fue necesario incorporar un sistema de propagación de información de distancia horizontal desde unas celdas a otras.

Conforme VODEC se fue desarrollando resultó obvia la proximidad de su objetivo con los diagramas de Voronoi. De hecho VODEC devuelve un diagrama de Voronoi en el que se ha utilizado un métrica no euclídea, conocida como métrica L_1 o métrica "Manhattan". Con dicha métrica el diagrama se simplifica, al estar compuesto por segmentos rectilíneos y eliminar los fragmentos parabólicos propios de los diagramas de Voronoi clásicos.

Así pues, VODEC surge como mejora sobre la descomposición vertical, o como medio de llegar a un diagrama de Voronoi desde la descomposición vertical. No obstante, el nuevo procedimiento aquí planteado, puede encuadrarse entre los algoritmos de cálculo de los diagramas de Voronoi como uno más. Sus diferencias con respecto a los demás generan una serie de ventajas e inconvenientes.

Como inconveniente ha de señalarse que VODEC, en su aplicación para hallar el diagrama de Voronoi completo, no es un algoritmo óptimo en tiempo de cómputo. Como se detallará en el apartado 3.5.1, tiene una duración del orden de $O(n^2)$ donde n es el número de vértices de todos los obstáculos agregados; mientras que el óptimo se encuentra en $O(nlogn)$. Sin embargo, en las condiciones más habituales (dispersión relativamente homogénea de los vértices), al igual que sucede con el método de

inserción de obstáculos, este tiempo puede reducirse drásticamente. Además, VODEC utiliza la descomposición vertical, que devuelve una trayectoria válida en *O(nlogn).*

Las principales ventajas de VODEC se derivan de considerar el espacio libre (C_{free}) en lugar de los obstáculos. Son las celdas, subconjuntos de C_{free}, la base del algoritmo. Así, como primera ventaja VODEC no tiene la necesidad de procesar todo el escenario. Puede utilizar la secuencia de celdas en la que se inscribe la trayectoria inicial ofrecida por la descomposición vertical como punto de partida. El algoritmo puede tener la necesidad de procesar, además de éstas, otras celdas vecinas para obtener precisión en la trayectoria deseada, pero sólo las indispensables, como se verá en el apartado 3.4.1. Al poder realizar este cálculo parcial del escenario, el algoritmo puede ofrecer tiempos de cómputo inferiores, adecuados para la planificación en línea.

Al igual que otros métodos (Papadopoulou y Lee, 2001), VODEC genera un Diagrama de Voronoi de métrica no euclídea, que cumple la condición de alejarse de los obstáculos. Esto otorga sencillez al eliminar las formas cuadráticas propias del Voronoi común, lo cual afecta positivamente en la determinación del grado del algoritmo tal y como es definido en Liotta y otros (1998).

Otra virtud es la adaptabilidad ante cambios en el escenario. Los métodos habituales de cálculo del Diagrama de Voronoi son del orden de O(*nlogn*) (Fortune, 1987; Kirkpatrick, 1979; Lee, 1982; Yap, 1987). Sin embargo, cualquier cambio en los obstáculos implica la necesidad de calcular de nuevo la totalidad del escenario (exceptuando los métodos de inserción incremental cuyo tiempo de cómputo pertenece a $O(n^2)$; Boissonnat y otros, 1992; Klein y otros, 1993). VODEC en cambio, consume ese tiempo adicional en almacenar información en los límites de las celdas, de modo que los cambios implican calcular únicamente las celdas afectadas.

Con respecto al método de descomposición en celdas exactas del que parte, pueden eliminarse sus condiciones de aplicación (engrosamiento previo de obstáculos), abarcando todas las soluciones posibles. Por otra parte proporciona información adicional como la detección de pasos estrechos o *"gaps"* (Xiaoshan, 2005), posibilitando un tratamiento especial en el postprocesado. Su ámbito de uso excede el de la planificación pudiendo aplicarse en los campos tradicionales de los diagramas de Voronoi en los que no se exija precisión, tales como el cálculo de área crítica en el diseño de chips VLSI (Papadopoulou y Lee, 2001).

3.1.1. Antecedentes sobre algoritmos de Voronoi

Los diagramas de Voronoi han sido objeto principal de atención dentro de la geometría computacional. El clásico caso de estudio consiste en un conjunto de puntos (que en algoritmos de planificación representarán a los obstáculos) dispersos sobre el plano. Aurenhammer (1991) recopiló los algoritmos más relevantes y describió las características y ámbitos de aplicación de los mismos. Entre los métodos considerados de coste computacional óptimo, los cuales se ejecutan en tiempo O(*nlogn*), se hallan los clásicos como el de línea de barrido (Fortune, 1987) y los que utilizan la estrategia de "divide y vencerás" (Shamos y Hoey, 1975). Además, en tiempo no óptimo pero rápidos a efectos prácticos (cuando hay una distribución relativamente homogénea de los obstáculos), se dispone de los de inserción incremental de obstáculos (Sugihara e Iri, 1992; Ohya y otros, 1984).

El de línea de barrido (Fortune, 1987) consiste en ordenar los puntos (u obstáculos puntuales) por abscisa creciente y luego proceder como si una línea vertical imaginaria recorriera el escenario de izquierda a derecha. En un instante dado los puntos situados a la izquierda de la línea ya han sido procesados, los de la derecha aún no, y con la línea se mantiene una estructura de datos que se altera con la aparición de cada nuevo punto o cada nodo del diagrama que se está calculando. El algoritmo va desplazando la línea y actualizando la estructura hasta que no quedan nodos o puntos a la derecha, dejando a su izquierda la solución.

El método de divide y vencerás (Shamos y Hoey, 1975) parte también de una clasificación ordenada de sus puntos. A continuación se subdivide el conjunto de puntos de forma binaria hasta obtener pares de puntos en cada subconjunto. Seguidamente se halla el diagrama de Voronoi de cada subconjunto (lo cual es la simple bisectriz del segmento que define el par). De nuevo, de forma recursiva se van calculando los diagramas de Voronoi correspondientes a la mezcla de cada par subconjuntos hasta lograr el completo.

El método de inserción incremental (Green y Sibson, 1978) trabaja con la triangulación de Delaunay, que es el dual del diagrama de Voronoi. En esta triangulación están unidos por segmentos aquellos puntos que son vecinos en el diagrama de Voronoi. El método parte del triángulo formado por los tres primeros puntos como una

aproximación a la descomposición triangular. En cada iteración, el método añade un nuevo punto a la construcción triangular previa. Para ello busca el triángulo al que pertenece y desde ahí comienza un proceso de comprobación de distancia a los vértices de los triángulos vecinos. Desde el nuevo punto se trazan los lados de los nuevos triángulos de Delaunay, dejando lista la estructura para la siguiente etapa.

Si en lugar de puntos, los obstáculos son segmentos, arcos circulares o polígonos, el resultado pasa a denominarse “diagrama de Voronoi generalizado”. La adaptación de los métodos clásicos anteriores a estos tipos de obstáculos dio lugar a una nueva familia de algoritmos que conservaba la eficiencia computacional anterior, es decir, un coste de O(*nlogn*) donde *n* es el número de vértices entre los que puede haber segmentos rectos o curvos de radio constante. Kirkpatrick (1979), Lee (1982) y Yap (1987) optaron por utilizar la técnica de “divide y vencerás”; Fortune (1987) la línea de barrido y Boissonnat (1992) y Klein (1993) el procedimiento de inserción incremental de obstáculos. Para obstáculos móviles existen también algunos trabajos (Guibas y otros, 1992; Roos, 1993).

Las desventajas de los diagramas de Voronoi con respecto a otros planificadores suelen ser su complejidad y su velocidad. Como ya se expresó en el capítulo 2 apartado 2.3, la arista de Voronoi entre un punto y un segmento es una parábola. Así, un escenario con obstáculos poligonales presentará un fragmento de este tipo de forma cuadrática toda vez que un vértice de un polígono sea el lugar más próximo a un segmento de otro. Esta abundancia de tramos parabólicos en el grafo resultante complica innecesariamente la trayectoria final, máxime cuando dicha trayectoria sólo pretende servir de guía aproximada a un algoritmo de postprocesado. Además, y de forma general, suponen una desventaja en cuanto a la representación gráfica y construcción (Aurenhammer, 1991) e incrementan ostensiblemente el grado del algoritmo (Papadopoulou y Lee, 2001).

En este sentido se han desarrollado abundantes trabajos para linealizar y simplificar el diagrama de Voronoi (Canny y Donald, 1988; De Berg y otros, 1993; Kao y Mount, 1991). De entre éstos destacan los denominados “diagramas de Voronoi compactos” (McAllister y otros, 1996), que han sido ampliamente estudiados. Otra alternativa es definir una nueva estructura, como por ejemplo la intersección producida por ondas rectas emergentes de los obstáculos, llamada “stright skeleton” (Aichholzer y Aurenhammer, 1995) que se aproxima al diagrama de Voronoi, pero que contiene

segmentos rectos. Otros recurren a una descomposición en celdas no exacta proporcionando algoritmos muy sencillos aunque en vez de un grafo simplificado devuelven un conjunto de puntos (Vleugels y Overmars, 1995; Maurer y Raghavan, 2003), que pueden agruparse después en una lista (Boada y otros, 2002).

Por otra parte, la métrica Manhattan proporciona soluciones sencillas, sin formas cuadráticas. El resultado de un diagrama de Voronoi con este tipo de métrica es un grafo formado por segmentos rectos, y la trayectoria obtenida es por tanto una línea poligonal. Existen diferentes algoritmos de Voronoi para tipos de métrica no euclídea entre las que se encuentra ésta (Hwang, 1979; Lee, 1980a; Lee y Wong, 1980b). Específicamente la utilización de la métrica Manhattan se detalla en Jünger y otros, (1993) y en Papadopoulou y Lee (2001).

La métrica Manhattan tiene una singular propiedad, que ha constituido uno de los pilares fundamentales de este trabajo. Dicha propiedad otorga la capacidad de descomponer el proceso de computación métrica en celdas separadas. Esto no significa que no pueda realizarse con la métrica euclídea también, pero no de modo tan simple como se verá en este capítulo (uso de funciones de flanco). Esta propiedad, unida a la sencillez de la descomposición vertical y su aplicabilidad a la planificación de trayectorias auspició el desarrollo del método VODEC que se muestra a continuación.

3.1.2. Desarrollo del capítulo

Aunque el método resulta sencillo en su implementación y ejecución, no resulta obvio el hecho de que su resultado sea exactamente un diagrama de Voronoi de métrica Manhattan. Se precisan una serie de justificaciones para llegar a esta conclusión, debido a lo cual el contenido de los siguientes apartados puede resultar denso.

En primer lugar se exponen las definiciones clásicas relativas a los diagramas de Voronoi. Después se detalla la métrica Manhattan y su propiedad más relevante, que será utilizada como piedra angular de este método. Seguidamente aparecen los conceptos relacionados con la descomposición vertical, identificados de la misma forma que en la literatura publicada. Partiendo de éstos, se aborda la definición de dos nuevas estructuras de datos íntimamente relacionadas con el método VODEC, que son la función de flanco y la línea de cresta. Todo ello queda incluido en el siguiente apartado.

Una vez establecidos los conceptos, se pasa a describir el algoritmo. Aunque los pasos son sencillos, requieren una serie de teoremas y propiedades que serán justificadas debidamente a lo largo de la sección 3.3.

Hasta aquí, VODEC se presenta como un método para obtener el diagrama de Voronoi. En el apartado siguiente, se exponen sus adaptaciones más relevantes y que dan sentido al algoritmo: su idoneidad para la planificación, y para escenarios dinámicos.

Por último se añaden las propiedades del método, como el tiempo de cómputo y las diferencias con respecto a la métrica euclídea.

3.2 Definiciones y propiedades básicas

3.2.1. Diagramas de Voronoi

Aunque los diagramas de Voronoi pueden establecerse en tres o más dimensiones, en adelante y para las definiciones próximas, sólo se aludirá a un entorno en dos dimensiones. Por otra parte, los diagramas de Voronoi se establecen con respecto a un conjunto de puntos. En este capítulo se aludirá a éstos como obstáculos (elementos de *CB*), ya sean de tamaño puntual ("*sites*" en la literatura científica) o sean conjuntos de puntos en forma de segmentos rectos, segmentos curvos o polígonos ("*generalized sites*"). En la fig. 5.1 se muestran obstáculos poligonales (*A*) y puntuales (*B,C,D*…). Se aludirá como distancia de un punto *p* a un obstáculo *A* como la comprendida entre *p* y el punto de *A* más próximo a *p* (ver fig. 5.1). A partir de aquí las siguientes definiciones son comunes en la literatura científica (Aurenhammer, 1991).

DEFINICION 1: La **región de Voronoi** correspondiente a un obstáculo *A* es el conjunto de puntos más cercano a *A* que a cualquier otro obstáculo.

DEFINICION 2: Una **arista de Voronoi** es el conjunto de puntos que limitan entre dos regiones de Voronoi dadas.

En la fig. 5.1 se muestra sombreada la región de Voronoi correspondiente al obstáculo puntual *F*. Además se señalan las aristas de Voronoi correspondientes a las fronteras de la región del obstáculo puntual *E*.

DEFINICION 3: Un **diagrama de Voronoi** es el que define y representa las aristas de Voronoi de un conjunto de obstáculos dado (fig. 5.1).

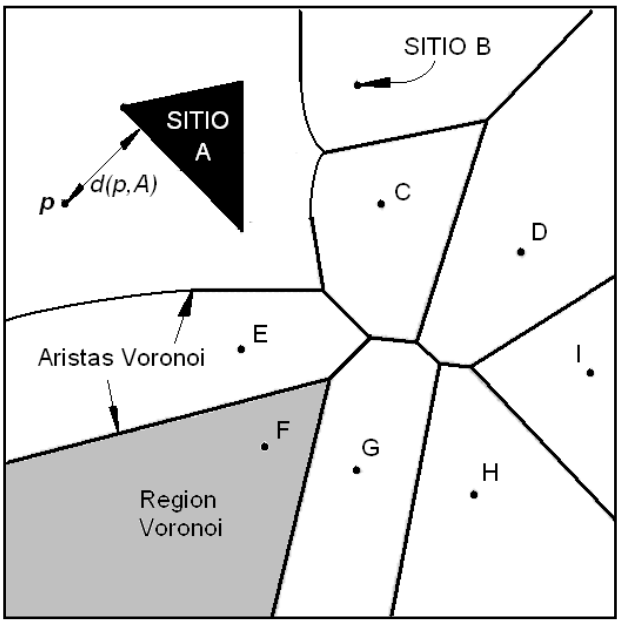


Figura 5.1: Diagrama de Voronoi.

3.2.2. Métrica Manhattan

El concepto de distancia entre dos puntos *a* y *b* está asociado normalmente a la métrica euclídea definida como:

$$d(a,b) = \sqrt{|a_x - b_x|^2 + |a_y - b_y|^2} \qquad (3\text{-}1)$$

Donde (a_x,a_y) son las coordenadas *x* e *y* de un punto *a* del plano; y (b_x,b_y) las correspondientes al punto *b*. Esta métrica pertenece a la familia conocida como "normas de Minkowski" definidas por la ecuación:

$$N_p(a,b) = \sqrt[p]{|a_x - b_x|^p + |a_y - b_y|^p} \qquad (3\text{-}2)$$

Que particularizada en p=2 resulta la norma euclídea. Para p=1, N_1 es conocida como "distancia Manhattan" de *a* a *b*, porque coincide con la mínima distancia a recorrer desde *a* hasta *b* siguiendo las líneas de una malla rectangular:

$$N_1(a,b) = |a_x - b_x| + |a_y - b_y| \qquad (3\text{-}3)$$

A continuación el uso del término “distancia” se referirá a la obtenida aplicando esta norma, y se denotará como $N_1(a,b)$. La figura 3.2 muestra la función de distancia desde un punto *P* hasta cualquier punto del plano *xy* en el eje *z*. Obsérvese que los puntos equidistantes se hallan cortando la superficie por un plano horizontal. En la figura 3.2 se muestran varios de estos cortes (distintos planos horizontales), representando sobre el plano *xy* uno de ellos. Se puede comprobar que en torno al punto *P* trazan la figura de un rombo.

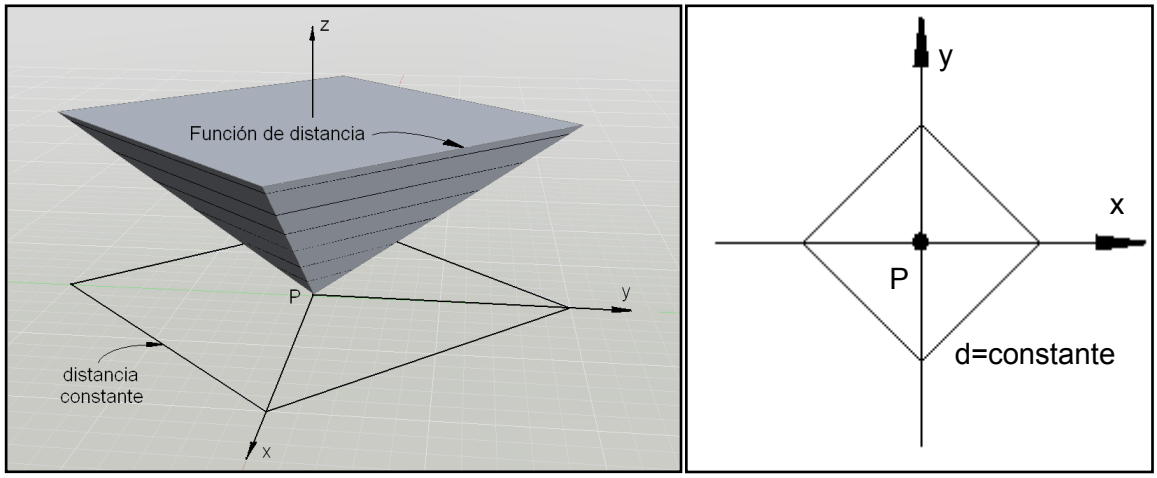


Figura 3.2: Función de distancia desde un punto P en norma N_1.

A continuación se exponen el lema y el corolario fundamentales de VODEC. Gracias a ellos es posible dividir el proceso de cálculo en celdas independientes.

Lema 1 ***Descomposición métrica***

Sean $a=(a_x,a_y)$ y $b=(b_x,b_y)$ dos puntos del plano tales que $a_x \le b_x$. Sea m un valor del intervalo $[a_x,b_x]$. Se define a’ como el punto (m,a_y), es decir, la proyección de a sobre recta x=m (ver fig. 3.3). Entonces:

$$N_1(a,b) = N_1(a,a') + N_1(a',b) \qquad (3\text{-}4)$$

Demostración: *dada la definición de m, $a_x \le m \le b_x$. Por tanto $(b_x-a_x)=|b_x-a_x|$, $(b_x-m)=|b_x-m|$, y $(m-a_x)=|m-a_x|$. Aplicando la definición de distancia:*

$$N_1(a,b) = |a_x - b_x| + |a_y - b_y| = (b_x - a_x) + |a_y - b_y| =$$
$$= (b_x - m) + (m - a_x) + |a_y - b_y| =$$
$$= |b_x - m| + |m - a_x| + |a_y - b_y| = N_1(a,a') + N_1(a',b)$$

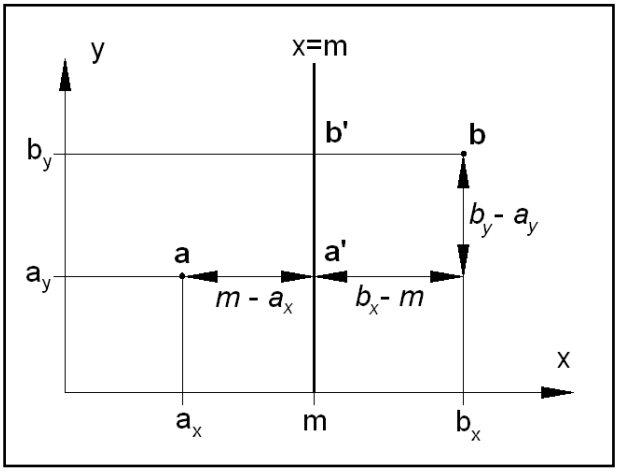


Figura 3.3: Explicación del Lema 1.

Corolario 1

Sea b' la proyección de b sobre la recta x=m; b'=(m,b_y). Entonces:

$$N_1(a,b)=N_1(b,b')+N_1(b',a) \qquad (3\text{-}5)$$

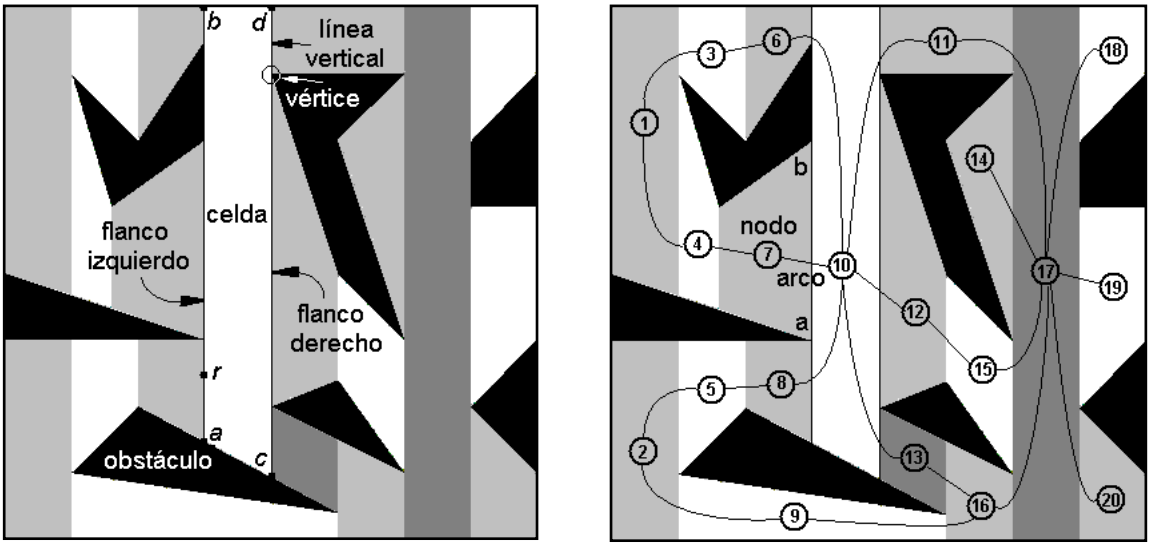
Demostración: *Se demuestra del mismo modo que el lema anterior, utilizando en esta ocasión la propiedad $b_x \geq b'_x \geq a_x$.*

3.2.3. Descomposición vertical: Celda y grafo de adyacencia

La descomposición vertical es un subtipo de la descomposición en celdas exacta (Latombe, 1991), tal y como se esbozó en la sección 2.4. Consiste en un método que divide el escenario en subconjuntos convexos del espacio de configuraciones libres de colisión, denominados celdas, y relacionados entre sí por su adyacencia (existencia de fronteras comunes). Para este método se define un tipo específico de celda convexa (ver fig. 3.4a), que es el correspondiente a la descomposición vertical (Chazelle, 1987).

DEFINICION 4: Sean *a* y *b* dos puntos del plano tales que $a_x=b_x=m$. Sean *c* y *d* otros dos puntos tales que $c_x=d_x=n>m$. Si $\overline{ab}$ y $\overline{cd}$ contienen al menos un vértice de un polígono cada uno, $\overline{bd}$ y $\overline{ac}$ son segmentos coincidentes con los lados de un polígono o los bordes del escenario, y la región del plano delimitada por el polígono *abcd* resulta ser un conjunto convexo del espacio de configuraciones libres de colisión, entonces dicha región se denomina ***celda***.

DEFINICION 5: Los límites verticales de una celda (en el caso anterior, los segmentos $\overline{ab}$ y $\overline{cd}$) se denominan ***flancos*** de la celda, y se distinguen por su ubicación respecto a la misma como ***flanco izquierdo*** y ***derecho*** respectivamente (ver fig. 3.4a).



a) Definición de celdas *b) Grafo de adyacencia*

Figura 3.4: Descomposición Vertical.

DEFINICION 6: El grafo definido por las celdas como nodos y los segmentos comunes a dos celdas como arcos entre los nodos que las representan se denomina ***grafo de adyacencia*** (ver fig. 3.4b).

Dado un escenario rectangular en el que se definen una serie de obstáculos poligonales, el método de descomposición vertical comienza ordenando sus vértices por abscisa creciente. Desde cada uno se traza una línea vertical que se prolongará hacia arriba y hacia abajo mientras permanezca dentro del espacio libre (C_{free}).Estas líneas servirán de límite lateral entre celdas. Los límites superior e inferior de éstas, consistirán

en los bordes de los obstáculos o las fronteras del escenario (fig. 3.4a). La principal propiedad de estas celdas es que son conjuntos convexos, es decir, cualquier par de puntos interiores a la misma puede ser unidos en un segmento recto sin intersecar sus fronteras. Al mismo tiempo que se van definiendo estas celdas, se construye paralelamente el grafo de adyacencia. Hasta aquí el método es exacto al tradicional definido en Chazelle (1987). De hecho, y es una de las ventajas de VODEC, se puede establecer una primera trayectoria a partir del grafo de adyacencia en un tiempo del orden de O(nlogn), al igual que el método tradicional. En la figura 3.4 se representa un escenario con obstáculos poligonales donde se ha procedido a efectuar este método.

3.2.4. Función de flanco

El siguiente concepto que se va a definir consiste en una función de distancia cuyo dominio es el flanco de la celda. Gracias a esta función y al lema 1, VODEC puede aislar en celdas el proceso de cálculo del diagrama de Voronoi.

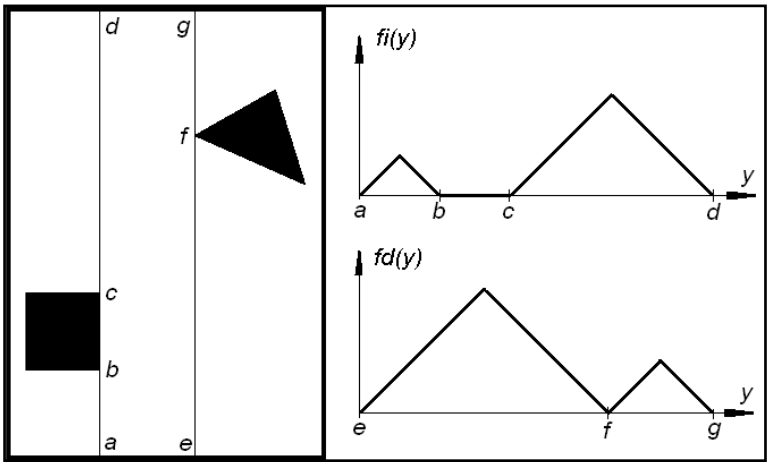
DEFINICION 7: Sobre cada punto de cada flanco de la celda (por ejemplo el punto r del flanco izquierdo en la fig. 3.4a) se puede definir una función de distancia calculando la misma hasta el obstáculo más cercano. Esta función se denominará ***función de flanco***, y se denotará bien como *fi(r)* cuando *r* sea un punto del flanco izquierdo de la celda, bien como *fd(r)* cuando *r* pertenezca al flanco derecho de la celda.

En el método VODEC, la función de flanco se construye de forma progresiva. Inicialmente sólo se tienen en cuenta los obstáculos situados sobre el propio flanco. Después, el algoritmo modifica la función si es necesario, es decir, si existen otros obstáculos más próximos que los considerados en un principio.

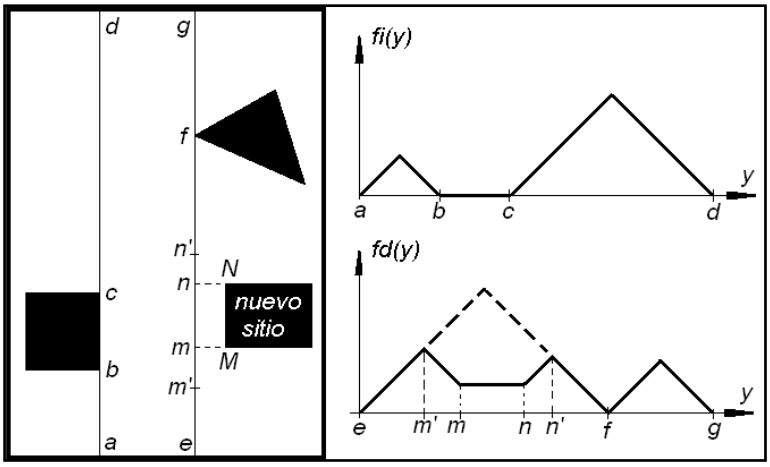
En la figura 3.5a se muestra a la izquierda un escenario con dos obstáculos, y a la derecha las funciones de flanco (*fi(r)* y *fl(r)*). Recorriendo el flanco izquierdo ($r \in [a,d]$) desde *a* hasta *d*, inicialmente (*r*=*a*), el obstáculo más próximo resulta ser el borde del escenario (*a*). En el propio punto *a* la distancia es nula, luego la función de flanco en *a* es cero *(fi(r=a)=0)*. Conforme se avanza hacia *b*, y mientras *a* siga siendo el punto más próximo (*r-a*<*b-r*), la distancia al obstáculo más cercano será la distancia hasta el punto *a* (*fi(r)*=*r-a*), siendo la función de flanco una recta de 45 grados. A partir del punto que

equidista de *a* y *b*, los siguientes (*b-r<r-a*) tendrán como obstáculo más cercano el punto *b*. Por tanto la función de flanco se representará con una recta diferente (*fi(r)=b-r*), que ahora pasa por *b*, y que de nuevo será nula al llegar a b (*fi(r=b)=0*). Obsérvese que sendas rectas y el propio flanco forman un triángulo isósceles, de 45 grados en la base.

Este triángulo isósceles se forma en todos los segmentos de espacio libre de la celda ($\overline{ab}$, $\overline{cd}$, $\overline{ef}$, $\overline{fg}$), dado que en todos ellos se cumple la misma circunstancia, es decir, los obstáculos más cercanos son los extremos de dichos segmentos. Los otros, o bien son vértices (*a, d, e, f, g*) o bien son lados pertenecientes a un obstáculo ($\overline{cb}$); con lo cual, la función de flanco valdrá cero. Como se puede observar, el cálculo de la función de flanco es sencillo y rápido (proporcional al número de vértices sitos sobre el flanco).



a)



b)

Figura 3.5: Función de flanco.

Posteriormente, en un proceso iterativo, las funciones de flanco van siendo modificadas para integrar las distancias a otros obstáculos. En la figura 3.5b se muestra cómo se ve afectada una función de flanco por un obstáculo no adyacente al mismo. Los puntos *m'* y *n'* definen el intervalo de influencia del nuevo obstáculo, es decir, el intervalo en el cual los puntos de *CB* más próximos han dejado de ser *f* y *g,* porque hay otros más cercanos en el nuevo obstáculo. Entre *m'* y *m* el punto más cercano resulta ser *M*, con lo que la función será una recta inclinada. Entre *m* y *n* los puntos más próximos serán en cada caso los ubicados con la misma ordenada en el segmento $\overline{MN}$, por tanto se obtiene una recta horizontal en la función de flanco. Finalmente, entre *n* y *n'*, el punto más cercano es *N*. Obsérvese como el fragmento de la función de flanco punteada de la fig. 3.5b es sustituida por una línea que representa valores inferiores.

Si el obstáculo y su intervalo de influencia están identificados, el proceso de modificación de la función de flanco resulta sencillo. Lo que no resulta evidente es cómo realizar esta identificación. Para ello VODEC utiliza la denominada línea de cresta que se verá más adelante.

Como se ha comentado anteriormente, el objeto de la función de flanco es circunscribir a la celda todas las operaciones necesarias para el cálculo del diagrama de Voronoi. Para ello ha de demostrarse que basta con las funciones de flanco y los límites de la celda para poder determinar la distancia desde cualquier punto de la misma a su obstáculo más próximo. Ello se logra aplicando el lema 1, como se verá en el siguiente teorema.

Teorema 1 ***Independencia métrica***

Si m es un punto de una celda, entonces la distancia desde m hasta el punto $Q \in CB$ más próximo, sólo depende de la distancia a los límites de la celda y/o de las funciones de flanco.

Demostración: *Por la definición de celda (pertenece a C_{free}), Q ha de ser exterior o estar en la frontera de la misma. Es decir, Q debe estar en una de las siguientes cuatro regiones: el semiplano definido por el flanco izquierdo en la dirección homónima, denominado "semiplano izquierdo" (ver fig. 3.6); el semiplano formado por el flanco derecho extendiéndose en*

dicha dirección, denominado "semiplano derecho" (ver fig. 3.6); los puntos de ordenada mayor que los del límite superior de la celda o "región superior"; y por último la región de ordenada menor que el límite inferior de la celda o "región inferior". Se demuestra a continuación la validez del teorema para cada caso.

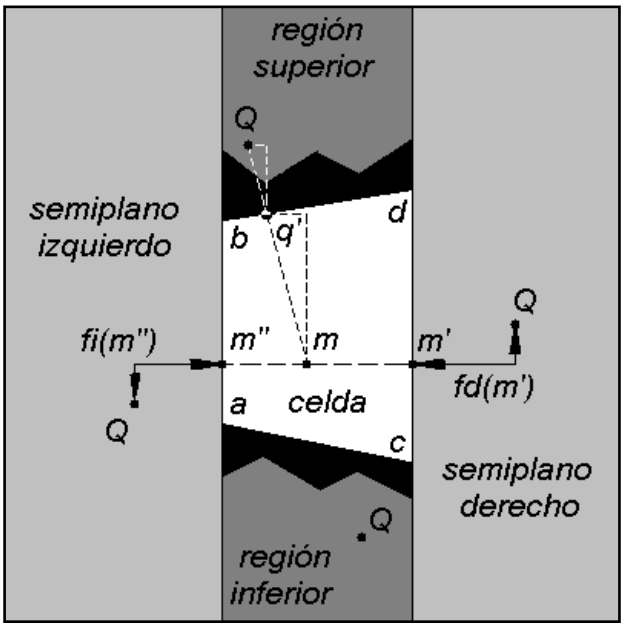


Figura 3.6: Demostración del Teorema 1.

Si Q se halla en el semiplano derecho, entonces la distancia podrá ser inferida de acuerdo al lema 1 como suma de dos distancias: la existente entre m y su proyección sobre dicho flanco m', y la distancia entre m' y Q:

$$N_1(m,Q)=N_1(m,m')+N_1(m',Q) \qquad (3\text{-}5)$$

Q es el punto de CB más cercano a m', ya que si existiera un punto $R \in CB$ más cercano entonces:

$$N_1(m,R)=N_1(m,m')+N_1(m',R)< N_1(m',Q) \qquad (3\text{-}6)$$

Lo cual contradice la condición del teorema. Por tanto (3-5) puede escribirse como:

$$N_1(m,Q)=N_1(m,m')+N_1(m',Q)= N_1(m,m')+fd(m') \qquad (3\text{-}7)$$

Lo que demuestra el enunciado para este caso.

El mismo razonamiento puede llevarse a cabo con el flanco izquierdo utilizándose para ello la proyección de m sobre el mismo, m''.

Para la región superior, la celda será colindante (segmento $\overline{bd}$), bien al límite del escenario, bien a un obstáculo. Si colindase con el límite del escenario, la región superior se reducirá al segmento $\overline{bd}$, es decir, Q sólo puede existir en la propia frontera de la celda, por lo que en este caso la distancia a Q coincide con la distancia al límite de la celda. Así se demuestra el teorema para este caso.

Si por el contrario colindase con un obstáculo, considérese el segmento recto $\overline{Qm}$. Dado que la región superior, a la que pertenece Q está separada de la celda por la recta bd, el segmento $\overline{Qm}$ tendrá que intersecarla en algún punto q' ($q' \in \overline{bd}$ y $q' \in \overline{Qm}$). Este punto, por pertenecer al segmento recto $\overline{Qm}$, o bien es el propio Q o bien no lo es. En el primer caso Q pertenecerá a la frontera de la celda ($\overline{bd}$) y el teorema quedaría justificado. En el segundo, el segmento $\overline{Qq'}$ no sería nulo, con lo cual $N_1(m,q')<N_1(m,q')+N_1(q'Q)=N_1(m,Q)$, es decir, q' sería más próximo a m que Q. Y dado que $q' \in \overline{bd} \in CB$, se niega la premisa del teorema. Por tanto Q no puede pertenecer más que a la frontera $\overline{bd}$, donde el enunciado queda demostrado.

El mismo principio puede aplicarse a la región inferior.

Este teorema es uno de los más importantes sobre los que se sustenta el método VODEC, ya que le permite dividir el problema del diagrama de Voronoi global al cálculo de sus fragmentos en cada celda de forma particular.

Por último, nótese que la función de flanco está definida, a intervalos, sobre la línea divisoria entre dos celdas. Dada la definición, en estos intervalos la función de flanco habrá de ser la misma para ambas celdas.

3.2.5. Línea de cresta

La línea de cresta es un elemento importante para VODEC pues tiene dos cometidos: por un lado permite establecer el fragmento de diagrama de Voronoi

contenido en la celda, y por otro determina los intervalos en los que la función de flanco ha de ser redefinida cuando es necesario.

La línea de cresta se calcula a partir de las funciones de flanco como se verá a continuación. En la figura 3.7 se muestra una celda con sus funciones de flanco abatidas a sendos lados. Sobre esta figura se procede a describir la construcción de la línea de cresta.

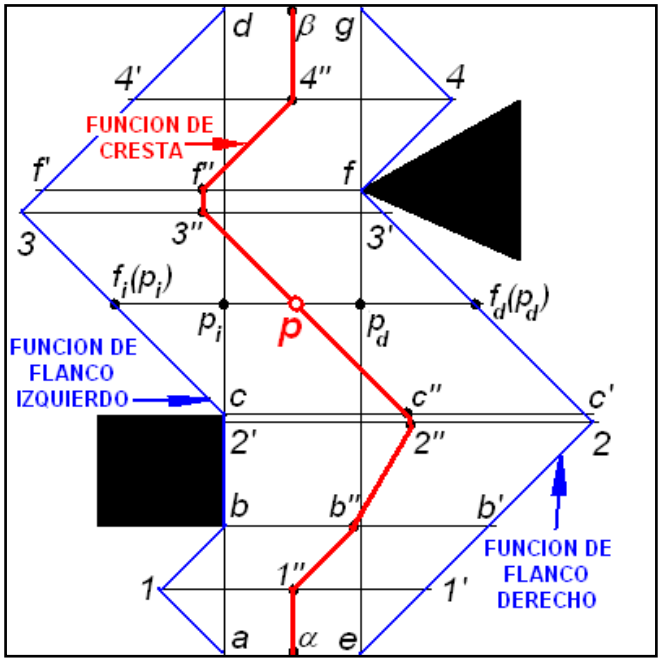


Figura 3.7: Línea de cresta.

Sea *p* un punto interior de la celda (figura 3.7). Se definen las funciones de distancia relativa a cada flanco como:

$$d_d(p) = d(p, p_d) + fd(p_d) \qquad (3\text{-}8)$$

$$d_i(p) = d(p, p_i) + fi(p_i) \qquad (3\text{-}9)$$

Donde p_d es la proyección de *p* sobre el flanco derecho, y p_i sobre el flanco izquierdo. El término $d(p,p_d)$ es la distancia entre *p* y p_d, y $d(p,p_i)$ es la distancia entre *p* y p_i. Hay que significar que esta definición también puede extenderse a puntos exteriores a la celda, con la salvedad de que $d(p,p_d)$ será negativo para puntos que excedan el flanco derecho, y lo mismo para $d(p,p_i)$ si *p* excede el flanco izquierdo. Siguiendo la notación

anterior $fd(p_d)$ es la función de flanco derecha sobre el punto p_d y $fi(p_i)$ la función de flanco izquierda sobre el puno p_i. Aplicando el lema 1, $d_d(p)$ es la distancia de p al obstáculo del semiplano derecho (fig. 3.7) más cercano si la función de flanco está definida con respecto a esta región. Del mismo modo $d_i(p)$ será la propia con respecto a los obstáculos del semiplano izquierdo.

DEFINICION 8: Se define **línea de cresta** como el conjunto *R*:

$$R = \left\{ x \in \Re^2 \middle| d_i(x) = d_d(x) \right\} \qquad (3\text{-}10)$$

Como se verá más adelante, en el caso de que la función de flanco derecho dependa de los obstáculos situados en el semiplano derecho, y ocurra lo propio con la función de flanco izquierdo, la línea de cresta es una aproximación al diagrama de Voronoi generado por los obstáculos situados a cada lado de la celda.

Las funciones de flanco se pueden representar como líneas poligonales abiertas, es decir, secuencias de segmentos tales que el extremo de uno de ellos coincide con el origen del segmento siguiente, lo cual se demostrará con posterioridad. Gracias a esta propiedad puede inferirse un método gráfico sencillo para el cálculo de la línea de cresta. Tal y como se muestra en la figura 3.7, en primer lugar se abaten las funciones de flanco sobre sus respectivos semiplanos, es decir, se hace coincidir la abscisa de la función con el flanco de la celda y el eje de ordenadas se extiende hacia el exterior de la misma. Seguidamente se halla el punto medio (α) del segmento formado por los primeros puntos de cada función de flanco (*a* y *e*). A continuación se procede a recorrer sendas funciones de flanco a lo largo de su dominio, localizando cada punto de inflexión (*1,b,c,3,d* para la función del flanco izquierdo; y *2,f,4,g* para el derecho). Estos puntos se ordenan en común, generando una secuencia en la que todos quedan clasificados de acuerdo a su posición vertical en la celda. En la figura 3.7 el primer punto de la secuencia sería el punto *1* en el flanco izquierdo, seguido de *b* en el mismo flanco, alternando a *2* en el derecho, *c* en el izquierdo, *3, f, 4* y los puntos finales *d* y *g*. Por cada uno de estos puntos de inflexión se traza una línea horizontal hasta atravesar la función de flanco opuesta, determinando sobre la misma su intersección: *1'* para el punto *1, b'* para el punto *b, 2'* para el punto *2*, y así sucesivamente. Quedan así definidos los segmentos $\overline{11'}, \overline{bb'}, \overline{2'2}$, etc. Se determinan ahora los puntos medios de cada uno de estos segmentos (α,*1",b",2",c",...,*β). La unión de dichos puntos de forma consecutiva con segmentos rectos constituirá la línea de cresta *R*.

La terminología de “cresta” deriva del aspecto que tiene la función de distancia asociada a los puntos de *R*. En la figura 3.8 se ha representado en el eje *z* el valor $min\{d_i(p),d_d(p)\}$ para todos los puntos *p* que pertenecen a la celda. Es decir, la distancia mínima, o distancia de cada punto de la celda al obstáculo más próximo. Puede observarse que esta función es una superficie (ver figura 3.8) que se construye deslizando las funciones de flanco izquierda y derecha en las direcciones definidas por los vectores (1,0,1) y (-1,0,1). Esta acción equivale a añadir, a las funciones de flanco (representadas sobre el eje *z*), el valor del trecho que en el sentido del eje x separa los puntos interiores de la celda a sus flancos ($d(p,p_d)$ y $d(p,p_i)$).

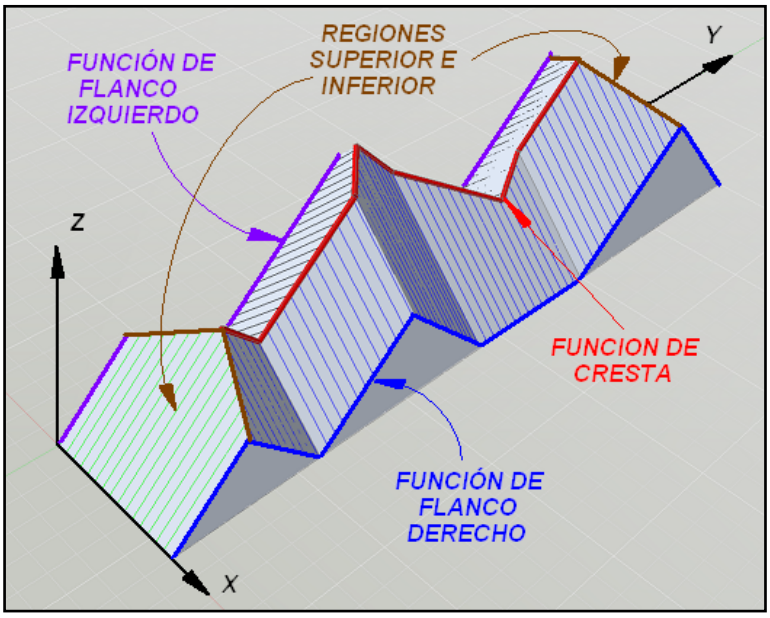


Figura 3.8: Línea de cresta.

La línea de cresta de las figuras 3.7 y 3.8 separa los puntos de la celda en dos regiones: los situados a su izquierda corresponderán a puntos cuyo obstáculo más próximo se encuentra en el semiplano izquierdo, y los situados a la derecha lo mismo con el semiplano derecho. Es decir, las regiones de Voronoi de los obstáculos de la izquierda de la celda están separadas de las de la derecha con la línea de cresta. Por tanto, **las aristas de Voronoi que pasen por la celda, correspondientes a dos obstáculos ubicados a cada lado de la misma, habrán de estar contenidas en la línea de cresta.** Así pues, la línea de cresta es una primera aproximación al fragmento del diagrama de Voronoi contenido en la celda. Sin embargo, faltan aún por añadir algunas aristas de Voronoi como por ejemplo las correspondientes a las regiones de Voronoi de los obstáculos superior e inferior. En la figura 3.8, el obstáculo inferior que

define la celda consiste en el propio eje x. La función de distancia asociada a dicho eje consiste en un plano apoyado en el mismo e inclinado 45º. La fracción del mismo representado en la figura 3.8 indica también la distancia mínima de esos puntos de la celda, y por tanto define la región de Voronoi del obstáculo inferior. Las aristas de Voronoi consistirán en los límites de esa región y serán agregadas posteriormente a la línea de cresta. Su cálculo se realiza utilizando los vectores directores de los planos implicados, que son, por un lado el plano que parte del límite inferior o superior de la celda, y por otro los sucesivos planos que parten de las funciones de flanco.

Hasta ahora se han introducido dos conceptos fundamentales: función de flanco y línea de cresta. La función de flanco permite hallar la distancia de cualquier punto de la celda al obstáculo más próximo. La línea de cresta sirve para encontrar las aristas de Voronoi. Por ejemplo, si las aristas de Voronoi entre los obstáculos situados a la derecha y a la izquierda de la celda se hallan en el interior de la misma, entonces la línea de cresta coincidirá con dicho límite. Estos conceptos plantean varias cuestiones: cómo se redefinen las funciones de flanco, qué ocurre cuando las regiones de Voronoi de un obstáculo lateral atraviesa completamente la celda, y cómo se determinan las aristas de Voronoi entre dos obstáculos situados a un mismo lado de la celda y que puedan invadirla. A todas ellas se les da respuesta en la explicación del algoritmo.

3.3 El algoritmo

VODEC comienza con el algoritmo de descomposición vertical, generando un conjunto de celdas y su grafo de adyacencia asociado. En el proceso de definición de las celdas se incluyen las funciones de flanco, calculadas a partir de los obstáculos adyacentes. A continuación, se entra en un proceso iterativo de cálculo de las funciones de cresta. Éstas, a tenor del procedimiento descrito para su cálculo (figura 3.7), podrán transgredir, o no, los límites de sus celdas. Supóngase que esto ocurre en el flanco derecho. Es decir, que existe un intervalo sobre la línea del flanco derecho para el cual la línea de cresta queda fuera de la celda invadiendo otras. Para que esto suceda, y siguiendo la notación de (3-8) y (3-9), en un punto *p* del flanco en dicho intervalo de invasión ha de cumplirse que $d_i(p)<d_d(p)=fd(p)$. Por tanto, hay un obstáculo en el lado izquierdo de la celda más próximo a *p* que el reflejado por la función de flanco derecho en *p*. Consecuentemente la función de flanco derecho está mal definida en ese punto y

ha de volver a calcularse. Al ser la función de flanco común a dos celdas, sus modificaciones afectarán a la celda adyacente. Al hacerlas, se estará incluyendo la influencia de los obstáculos de unas celdas a otras.

Así pues el algoritmo entra en una fase iterativa (fase 2 de la fig. 3.9) en la que se procesan las líneas de cresta, se determina si existe invasión y si la hay se redefinen las funciones de flanco. Ésta es la forma en que las celdas interaccionan entre sí. Es decir, el resultado del procesamiento de cada celda, que pueda afectar a otras, se transmite a esas otras a través de las modificaciones de las funciones de flanco. Dado que las funciones de flanco son comunes a dos celdas en el dominio definido por sus segmentos verticales de adyacencia, sólo es necesario guardar, una vez redefinidas las funciones de flanco, las celdas afectadas para un postprocesado posterior. La lista *A* (figura 3.9) almacena las celdas pendientes de ser procesadas. Esta fase concluye cuando *A* está vacía, y por tanto las funciones de flanco están todas correctamente definidas.

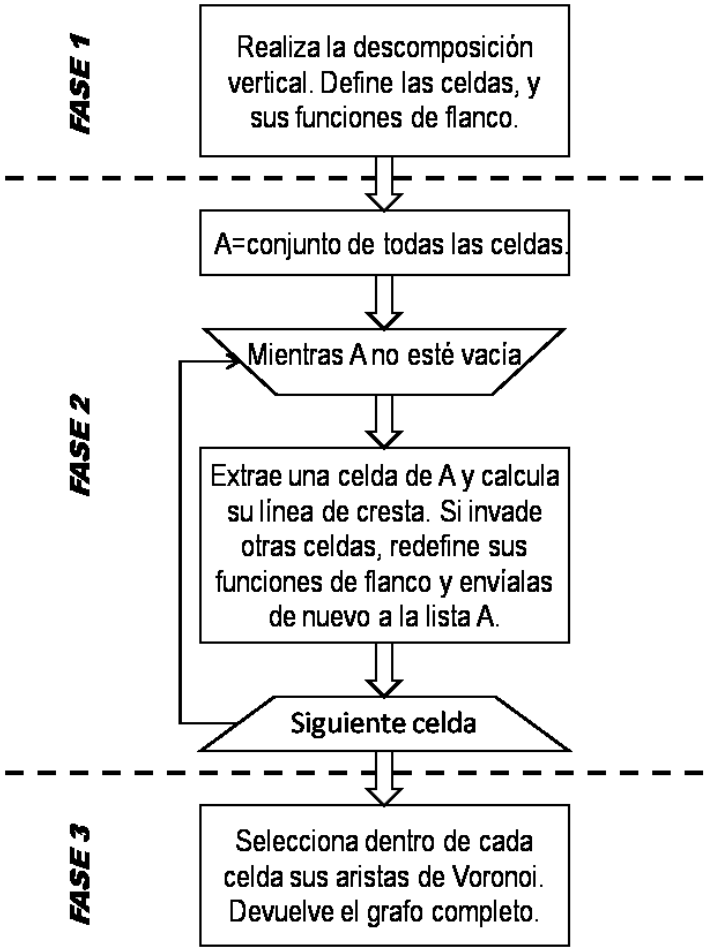


Figura 3.9: Algoritmo de VODEC.

Cuando estas invasiones acaban, sobre cada celda se calcula el fragmento del diagrama de Voronoi que queda delimitado por la misma. Para ello, de la línea de cresta se extrae aquello que no pertenece al diagrama de Voronoi (como los fragmentos de invasión), se añaden las aristas de Voronoi correspondientes a los límites inferior y superior de la celda, y por último se añaden aristas de Voronoi horizontales (como las que se dan entre obstáculos situados a un mismo lado de la celda). Uniendo todos los fragmentos se obtiene la solución completa.

La explicación exhaustiva de estas fases se verá a continuación, no obstante, a modo introductorio puede observarse que la base del algoritmo consiste en el trato de las celdas como unidades independientes de información. Cada una de ellas, además de su contorno geométrico, tiene asociadas dos funciones de flanco y una línea de cresta que se calcula a partir de las anteriores. Como colofón habrá una fracción del diagrama de Voronoi que atraviesa la celda, sin embargo esta última no se calculará hasta el final, cuando en la celda se dispone de toda la información métrica necesaria.

3.3.1. Fase 2: Propagación de la información métrica

El objetivo de la fase 2 del algoritmo es propagar la información métrica de unas celdas a otras, de forma que todas las funciones de flanco queden correctamente definidas. Tal y como se indica en el apartado 3.2.4, inicialmente las celdas definen sus funciones de flanco atendiendo tan sólo a los puntos de los obstáculos sitos en el propio flanco. Para cualquier obstáculo poligonal, sus lados o bien tienen una pendiente finita y constituyen el límite superior o inferior de una celda, o bien son verticales y por lo tanto adyacentes a uno de sus flancos. Es decir, siempre existe una celda en la cual dicho lado es un segmento presente.

El método se basa en determinar dentro de cada celda si la región de Voronoi asociada a cada lado de obstáculo la excede, y si es así, trasladar esa información a las celdas afectadas. **Esto ocurre cada vez que la línea de cresta atraviesa los flancos de la celda** (figura 3.10). Entonces se procede a redefinir las funciones de flanco de modo que contengan las distancias al nuevo elemento detectado.

En el ejemplo de la figura 3.10a, la línea de cresta penetra el flanco derecho de la celda *1* entre los puntos *a* y *b*. Por tanto, la región de Voronoi del obstáculo *K* atraviesa

la celda afectando a la contigua. La función de flanco derecho de la celda *1* ha de ser redefinida para que contenga las distancias al obstáculo más cercano, que entre los puntos a y b será el obstáculo *K*. Esta distancia será igual a $d_i(x)$ para todo *x* del intervalo *[a,b]*. Es decir, es igual a la función de flanco izquierdo incrementada en el ancho de la celda (figura 3.10b).

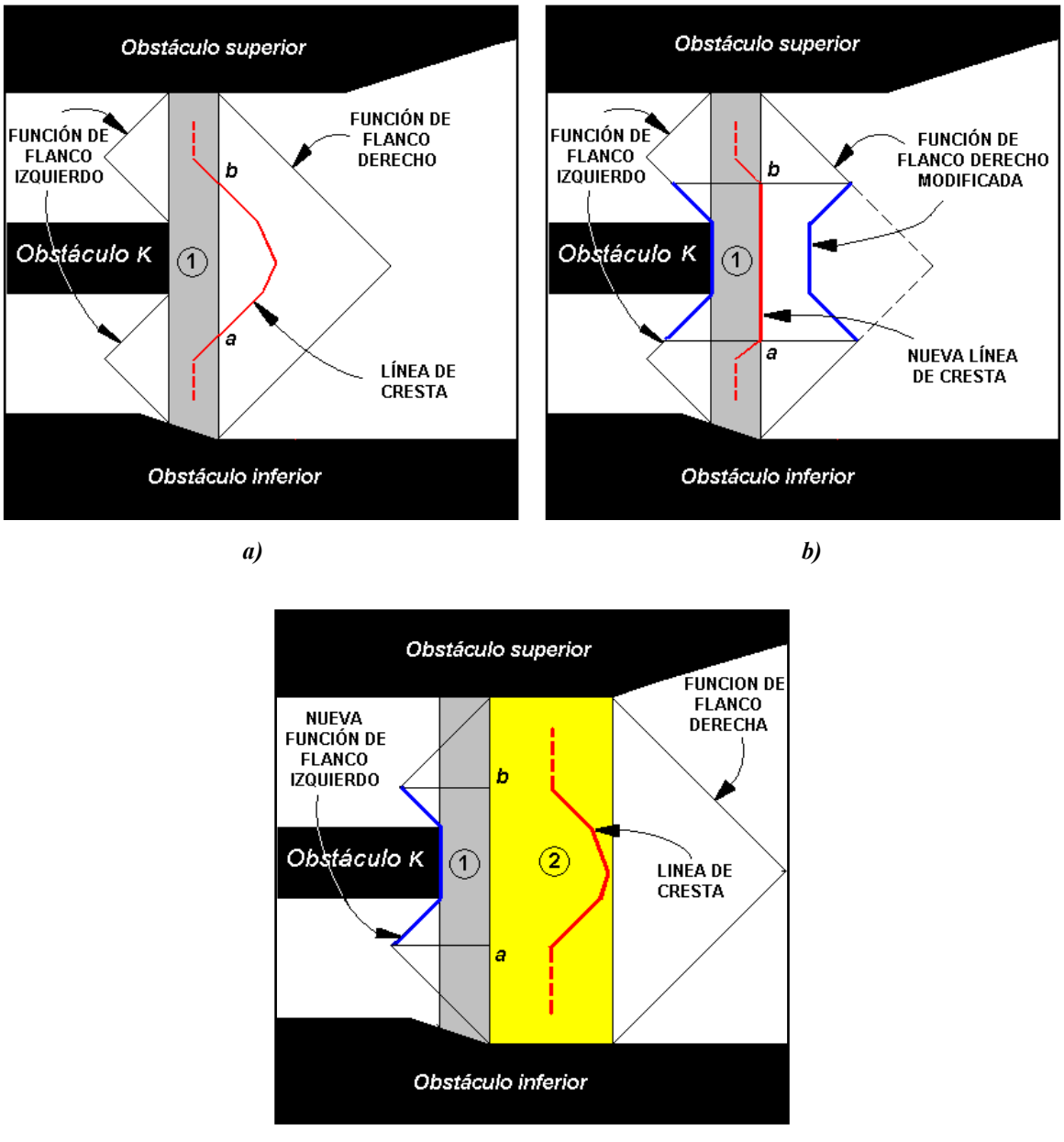


Figura 3.10: Invasión de flancos por líneas de cresta.

La función de flanco izquierdo de la celda *2* es la misma función de flanco derecho de la celda *1* (fig. 3.10c), y con ella se construirá la nueva línea de cresta. Ésta por tanto reflejará las aristas de Voronoi del obstáculo *K*. En este caso, la línea de cresta queda contenida dentro de la celda, con lo que no será necesario volver a repetir el proceso.

Si por el contrario el flanco derecho de la celda *2* nuevamente se viese invadido, se procedería a calcular las funciones de flanco pertinentes y las líneas de cresta de las celdas afectadas. De este modo las funciones de flanco pasan de considerar sólo los obstáculos del flanco (tal y como se definen inicialmente) a los obstáculos situados a una o varias celdas de distancia.

Obsérvese que si se calcula de nuevo la línea de cresta de la celda *1,* queda adyacente al flanco en el intervalo invadido. Es decir, si la invasión fue procesada, la línea de cresta queda inscrita en la celda. La utilidad de esta propiedad es que la inscripción de la línea de cresta dentro de la celda sirve de indicador de que las funciones de flanco están correctamente definidas, lo que se conoce con el término de celda equilibrada.

DEFINICION 9: Una celda está **"equilibrada**" cuando su línea de cresta queda inscrita en la propia celda.

Ahora puede comprenderse mejor el algoritmo descrito en la figura 3.9. En una primera fase se definen las celdas. Para ello se asume una lista de vértices ordenada previa. En cada celda se define inicialmente su función de flanco atendiendo a los puntos de obstáculo sitos en dicho flanco.

En una segunda fase se define una lista que contendrá todas las celdas. A partir de aquí tiene lugar un bucle que no cesará hasta que todas las celdas están equilibradas. Dentro del bucle se calculan las líneas de cresta. Si no existe invasión, la celda queda equilibrada, se elimina de la lista y se procede con la siguiente. Si existe invasión, la celda invadida se reincorpora a la lista (si no lo estaba) para ser procesada de nuevo. Cuando el bucle finaliza todas las celdas están equilibradas, y por tanto todas las funciones de flanco quedan correctamente definidas.

Una vez equilibradas, en la última fase se puede construir el fragmento del diagrama de Voronoi que cruza cada celda a partir de las funciones de flanco. Con estos fragmentos se construye el grafo final.

3.3.2. Fase 3: Obtención del Diagrama de Voronoi

Como ya se ha mencionado con anterioridad, no todas las partes de las líneas de cresta forman parte del grafo general de Voronoi. Por ejemplo, los tramos que yacen en el lateral de cada celda pueden corresponder a invasiones tratadas y por tanto no entran en el grafo general. Es necesario además agregar las aristas de Voronoi entre dos obstáculos situados a un mismo lado de la celda. También han de añadirse las aristas de Voronoi asociadas a los obstáculos inferior y superior de la celda.

DEFINICION 10 Se dice que una celda está **conclusa**, cuando se han determinado todas las aristas de Voronoi contenidas en ella.

El objetivo en esta fase por tanto es transformar las celdas equilibradas de que parte, en celdas conclusas. Para ello se calculan en primer lugar las aristas de Voronoi de los obstáculos superior e inferior. En esta operación entran en juego únicamente las funciones de flanco y los segmentos superior e inferior que delimitan la celda. En segundo lugar se suprimen de la línea de cresta los fragmentos que corresponden a invasiones o quedan dentro de las regiones de Voronoi de los obstáculos superior e inferior recién calculadas. Por último se añaden las aristas de Voronoi entre obstáculos situados a un mismo lado de la celda. Éstas, como se demostrará en los siguientes lemas y teoremas, son horizontales en el interior de la celda. Gracias a esta propiedad, para agregar estas aristas al fragmento del diagrama de Voronoi de una celda, se inicia un segmento horizontal en el punto de flanco equidistante de los dos obstáculos, y se detiene al tocar la primera de las aristas de Voronoi presentes (de la línea de cresta o de los obstáculos superior e inferior) o el límite opuesto de la celda.

DEFINICIÓN 11 Se denomina **entrada** de una celda al punto del flanco que equidista entre dos puntos situados a un mismo lado de la celda.

Si el flanco de la celda se divide en intervalos asociados a su obstáculo más próximo, las entradas se localizan en los extremos de estos intervalos. La función de flanco es una línea poligonal como se demostrará próximamente, y los vértices de la función de flanco son etiquetados como entradas o no. A medida que tienen lugar las invasiones, esta información se actualiza, para que al llegar a esta fase final, la construcción del diagrama de Voronoi sea sencilla. Se precisa por tanto demostrar que la función de flanco es una línea poligonal. Este hecho, derivado de la naturaleza poligonal de los obstáculos, es necesario para justificar el procedimiento de cálculo de las líneas

de cresta y del diagrama de Voronoi. Con esta intención se exponen los siguientes enunciados. Los lemas 2 y 3 junto al corolario 2 sirven para determinar qué punto de un obstáculo dado (que pertenecerá a un segmento genérico $\overline{AB}$ de su frontera) es el más próximo al punto de flanco considerado. El teorema 2 se apoya en estos resultados para limitar la pendiente de la función de flanco. El teorema 3 culmina los razonamientos previos para demostrar la morfología poligonal de la función de flanco. Por último, el teorema 4 demuestra la horizontalidad de las aristas de Voronoi de los obstáculos situados a un mismo lado de la celda.

Tal y como se enunciaba en la introducción, la distancia de un punto a un obstáculo es la distancia hasta el punto más próximo de dicho obstáculo. Si el obstáculo es un segmento $\overline{AB}$ definido por los puntos $A=(A_x,A_y)$ y $B=(B_x,B_y)$; y $P=(P_x,P_y)$ es un punto cualquiera; la distancia de P al segmento $\overline{AB}$ será $min\{N_1(P,C)\}\ \forall C\in\overline{AB}$.

Lema 2 Distancia a un segmento de baja pendiente

Sea $r=(B_y-A_y)/(B_x-A_x)$ la pendiente del segmento $\overline{AB}$ ($A_x<B_x$). Sea $P=(P_x,P_y)$ un punto tal que $P_x\in[A_x,B_x]$ y $P_y\leq min(A_y,B_y)$. Sea $P'=(P'_x,P'_y)$ un punto de $\overline{AB}$ tal que $P'_x=P_x$.

$$Si\ |r|\leq 1\ entonces\ N_1(P,\overline{AB}) = N_1(P,P')$$

Demostración: *Se desea demostrar que P' es el punto de $\overline{AB}$ más cercano a P (ver fig. 3.11 izquierda). Para ello, sea Q otro punto cualquiera del segmento. Se probará que $N_1(P,P')< N_1(P,Q)\ \forall Q\in\overline{AB}$*

$$\begin{aligned} N_1(P,Q) &= |Q_x-P_x|+|Q_y-P_y| = |Q_x-P'_x|+Q_y-P_y = \\ &= |Q_x-P'_x|+Q_y-P'_y+P'_y-P_y = \\ &= |Q_x-P'_x|+Q_y-P'_y+N_1(P',P) \end{aligned} \qquad (3\text{-}11)$$

Si la distancia a Q es mayor que la distancia a P', entonces P' es el punto más próximo. Observando la expresión (3-8), eso ocurre si $Q_y-P'_y>0$. Por otro lado si $Q_y-P'_y\leq 0$:

$$\begin{aligned} |r| &= \left|\frac{Q_y-P'_y}{Q_x-P'_x}\right|\leq 1 \\ |Q_y-P'_y| &\leq |Q_x-P'_x| \end{aligned} \qquad (3\text{-}12)$$

Consecuentemente $\left|Q_x-P'_x\right|+Q_y-P'_y \geq 0$, entonces $N_1(P,Q) \geq N_1(P,P')$ y P' es el punto más próximo a P en este caso también.

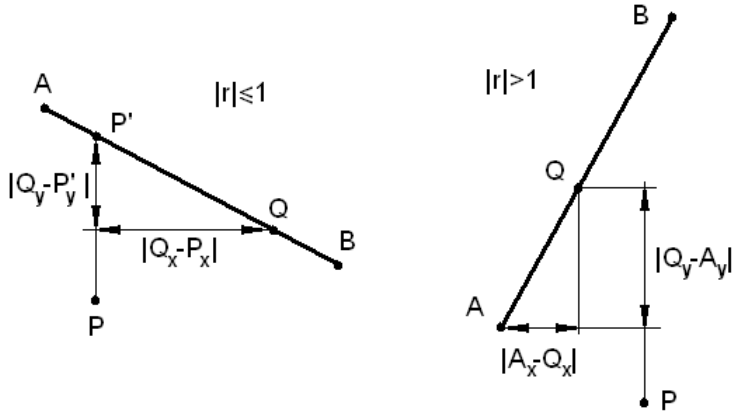


Figura 3.11: Distancia de un punto a un segmento.

Lema 3 Distancia a un segmento de elevada pendiente

Sea $r=(B_y-A_y)/(B_x-A_x)$ la pendiente del segmento $\overline{AB}$ ($A_x<B_x$). Sea $P=(P_x,P_y)$ un punto cualquiera tal que $P_x \in [A_x,B_x]$ y $P_y \leq \min(A_y,B_y)$. Sea $P'=(P'_x,P'_y)$ el punto de $\overline{AB}$ cuya componente $P'_x=P_x$. Si $\left|r\right|>1$. Entonces:

$$N_1(P,\overline{AB}) = \min \{ N_1(P,A), N_1(P,B) \}$$

Demostración: *A partir de la condición $\left|r\right|>1$ (ver figura 3.11 derecha):*

$$|r| = \left|\frac{Q_y - A_y}{Q_x - A_x}\right| > 1 \qquad (3\text{-}13)$$
$$\left|Q_y - A_y\right| > \left|Q_x - A_x\right|$$

Supóngase el caso r>1. Entonces $A_y<Q_y<B_y$, dado que $A_x<Q_x<B_x$.

$$\begin{aligned} N_1(P,Q) &= \left|Q_x - P_x\right| + \left|Q_y - P_y\right| = \\ &= \left|Q_x - P_x\right| + Q_y - A_y + A_y - P_y = \\ &= \left|Q_x - P_x\right| + Q_y - A_y - \left|A_x - P_x\right| + \left|A_x - P_x\right| + A_y - P_y = \\ &= \left|Q_x - P_x\right| + Q_y - A_y - \left(P_x - A_x\right) + N_1(P,A) \end{aligned} \qquad (3\text{-}14)$$

Aplicando (3-13) al término (Q_y-A_y) de 3-14 se tiene

$$N_1(P,Q) \le N_1(P,A) + |Q_x - P_x| + Q_x - A_x - (P_x - A_x)$$
$$N_1(P,Q) \le N_1(P,A) + |Q_x - P_x| + Q_x - P_x \qquad (3\text{-}15)$$

En (3-15) si $Q_x \ge P_x$ entonces $Q_x - P_x > 0$ y la inecuación $N_1(P,Q) \le N_1(P,A)$ se cumple. Si por el contrario $Q_x < P_x$, entonces $Q_x - P_x = -|Q_x - P_x|$, con lo que ambos términos se anulan y la inecuación se sigue cumpliendo.

La otra alternativa sería $r < -1$. Entonces $A_y > Q_y > B_y$, y los mismos pasos pueden realizarse para el punto B, que sería el más próximo a P.

Corolario 2 ***Distancia a un segmento alejado***

Sea el segmento $\overline{AB}$ con $A_x < B_x$ y r su pendiente. Sea $P=(P_x,P_y)$ un punto tal que $P_x \notin (A_x,B_x)$ y $P_y \le \min\{A_y,B_y\}$. Entonces:

$$N_1(P,\overline{AB}) = \min\{ N_1(P,A),\ N_1(P,B)\}$$

Demostración: *Sea P' la proyección de P sobre la línea $x=A_x$ si $P_x < A_x$, o sobre $x=B_x$ si $P_x > B_x$. Por el lema 1 se tiene que $N_1(P,\overline{AB}) = N_1(P,P') + N_1(P',\overline{AB})$. Como $P'_x \in (A_x,B_x)$, si $|r| < 1$, entonces se satisface el lema 2 y $N_1(P',\overline{AB}) = N_1(P',P'')$, donde P'' es la proyección de P' sobre $\overline{AB}$, es decir, A o B dependiendo del valor de P_x. Si $|r| > 1$, por el lema 3 se tiene: $N_1(P',\overline{AB}) = \min\{ N_1(P',A), N_1(P',B) \}$. De cualquier modo $N_1(P,\overline{AB}) = N_1(P,P') + \min\{ N_1(P',A) , N_1(P',B) \}$, y el corolario queda demostrado.*

Teorema 2 ***Pendiente máxima de la función de flanco***

Sea f(y) una función de flanco. Entonces $|f'(y)| \le 1$

Demostración: *Sea P un punto del flanco. Por definición $f(P) = N_1(P,E)$ donde E es el obstáculo más cercano a P, ya sea un vértice o un segmento. Si E es un vértice (caso 1), $N_1(P,E) = |P_x - E_x| + |P_y - E_y|$. Todo punto P del flanco tiene la misma P_x, así $f'(P) = N_1'(P,E) = |P_y - E_y|' = \pm 1$.*

Si E es un segmento entonces los puntos del flanco pueden pertenecer al intervalo $[A_y,B_y]$ (caso 2) o no (caso 3). En el caso 2, si $|r|<1$, aplicando el lema 2, $N_1(P,\overline{AB}) = N_1(P,P')$, donde P' es la proyección de P sobre $\overline{AB}$. Así:

$$N_1(P,P')=|P'_x-P_x| + |P'_y-P_y| = |P'_x-P_x| = |r(P_y-A_y)+A_x-P_x| \qquad (3\text{-}16)$$

En (3-13) todo es constante excepto P_y. Por lo tanto, $N_1'(P,P')=|r|<1$

Si $|r|>1$ (caso 3), por el lema 3 el punto más cercano será A o B. Éste caso comparte la condición del caso 1. En ambas situaciones la función de flanco tiene un vértice como obstáculo más próximo, y por lo tanto f'(P)=±1.

Teorema 3 Morfología de la función de flanco

La función de flanco es una línea poligonal.

Demostración: *Tal y como se demostró en el teorema 2, la función de flanco está compuesta por distancias a vértices, o distancias a segmentos poco inclinados ($|r|<1$), es decir, segmentos cuya pendiente en valor absoluto nunca supera la unidad. Para llegar a la conclusión de que se trata de una línea poligonal se precisa demostrar continuidad. Sea $P\in CB$ el punto más próximo a Q, un punto sito en el flanco. $\forall R\in\Re^2$ se tiene que:*

$$\begin{aligned} &N_1(P,Q) = |Q_x - P_x| + |Q_y - P_y| = \\ &N_1(P,Q) = |Q_x - R_x + R_x - P_x| + |Q_y - R_y + R_y - P_y| \\ &N_1(P,Q) \le |Q_x - R_x| + |R_x - P_x| + |Q_y - R_y| + |R_y - P_y| \\ &N_1(P,Q) \le N_1(P,R) + N_1(R,Q) \end{aligned} \qquad (3\text{-}17)$$

Si R es un punto del flanco, cuando $R\to Q$, $N_1(R,Q)\to 0$, y por (3-17) significa que $N_1(P,R)\to N_1(P,Q)$, lo cual implica continuidad en la función de flanco.

La figura 3.12 muestra la función de flanco obtenida para los obstáculos representados. En ordenadas se representa por un lado el valor de la función, y de forma solapada, el semiplano del flanco que constituye el eje de abscisas. Así se tiene la línea poligonal de vértices *M, E1, b, E2, c, E3, d, e, f, E4, N* como función de flanco, y por otro lado los obstáculos *A B* y *C* que la definen. Obsérvese que de dichos obstáculos sólo

afectan los segmentos o vértices de la función de flanco más cercanos. Así, del segmento $\overline{ab}$, dado que su pendiente en valor absoluto es inferior a la unidad, se tiene que todo el fragmento $\overline{E1b}$ pasa a definir la función de flanco (Lema 2). Entre b y E2, el segmento más próximo es el $\overline{ab}$, y por el corolario 2 se sabe que el vértice b es el punto más cercano, con lo que la función pasa a ser una recta de pendiente unidad. Igual ocurre con el tramo $\overline{E2c}$, donde por ser el segmento $\overline{c'c}$ el más próximo y su pendiente inferior a -1, cumple las condiciones del lema 3, siendo *c* el vértice más próximo. $\overline{cE3}$ repite el caso anterior pero con pendiente positiva. Para $\overline{E3d}$, *d* es el vértice más próximo. Los segmentos *de*, $\overline{ef}$ y $\overline{fE4}$ tienen pendiente en valor absoluto inferior a la unidad con lo que se integran directamente en la función de flanco. En los límites *M* y *N* existe un obstáculo, el que define por la parte inferior y superior la celda. Desde estos extremos se levantan los segmentos $\overline{ME1}$ y $\overline{NE4}$ indicando que en sus respectivos dominios de la función de flanco, el vértice más próximo resulta ser precisamente *M* o *N*.

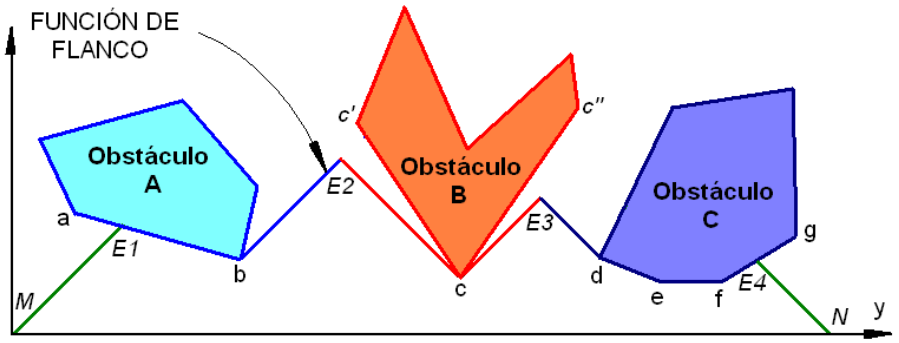


Figura 3.12: Aspecto de la función de flanco.

Teorema 4 ***Horizontalidad de las Aristas de Voronoi laterales***

Las aristas de Voronoi definidas por dos obstáculos situados al mismo lado de una celda, son horizontales dentro de la celda.

Demostración: *Sin pérdida de generalidad, considérese el flanco izquierdo (figura 3.13). Si la arista de Voronoi no fuese horizontal, entonces podría haber dos puntos P y Q interiores a la celda, tales que* $P_y=Q_y$, *y*

P_x<Q_x en dos regiones de Voronoi diferentes. Sea P' la proyección de P sobre el flanco. Para P, N(P,P')+N(P',E1) es la distancia hasta el obstáculo más próximo E1. Para Q, N(Q,P')+N(P',E2) también debe ser la distancia hasta su obstáculo más próximo E2. Si N(P',E1) = N(P',E2) entonces E1 y E2 son equidistantes y P y Q pertenecerán a una arista de Voronoi. Por otro lado, si N(P',E1)<N(P',E2) entonces E1 será el obstáculo más cercano a Q, y por consiguiente Q no puede pertenecer a la región de Voronoi de E2. Lo mismo ocurre con el punto P para el caso N(P',E1)>N(P',E2).

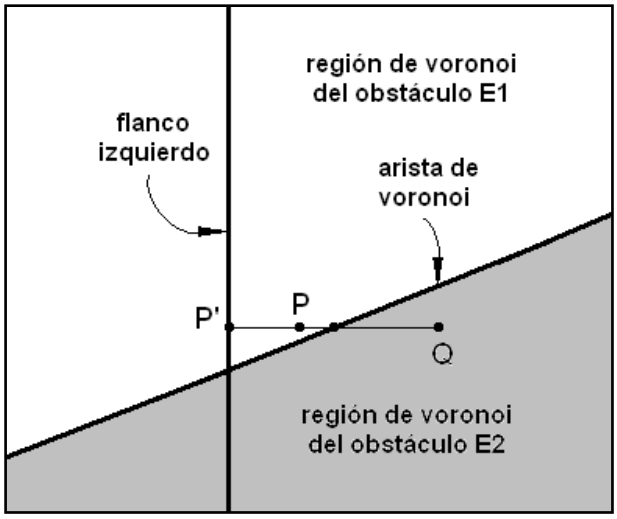


Figura 3.13: Demostración del teorema 4.

El teorema 3 permite definir la función de flanco como una secuencia de vértices. Las aristas de Voronoi del teorema 4 parten del flanco con pendiente nula hasta cortar con otras, o bien, si no es así, hasta atravesar completamente la celda (figura 3.14). El punto de partida coincidirá con uno de los vértices de la función de flanco, pues ha de ser la intersección entre las funciones de distancia a dos elementos (el caso en que dicha intersección es diferente a un punto se tratará al final del capítulo). De este modo, para cada vértice de la función de flanco, se reserva la información adicional sobre si es o no origen de una arista de Voronoi. De acuerdo a la definición 12, a estos puntos se les denomina entradas a la celda y están representados en la figura 3.12 como los vértices *Ex*, donde *x*=1,2,3,4.

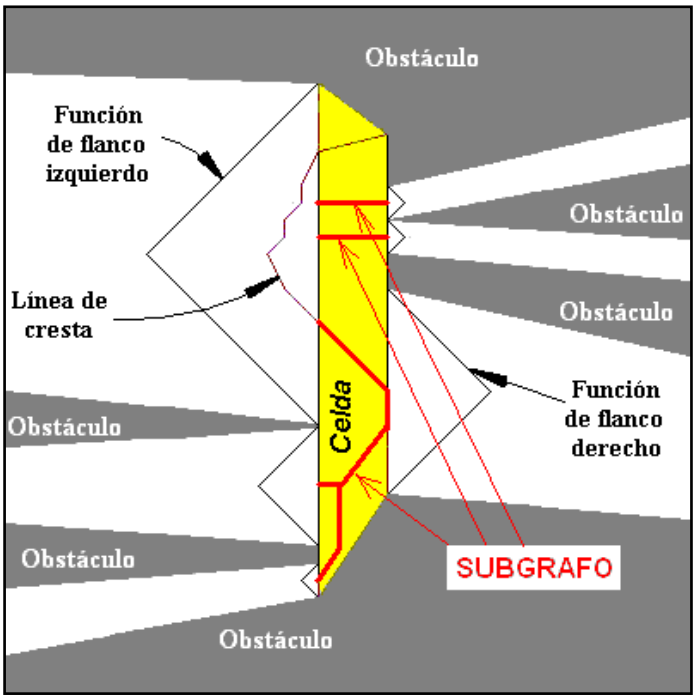
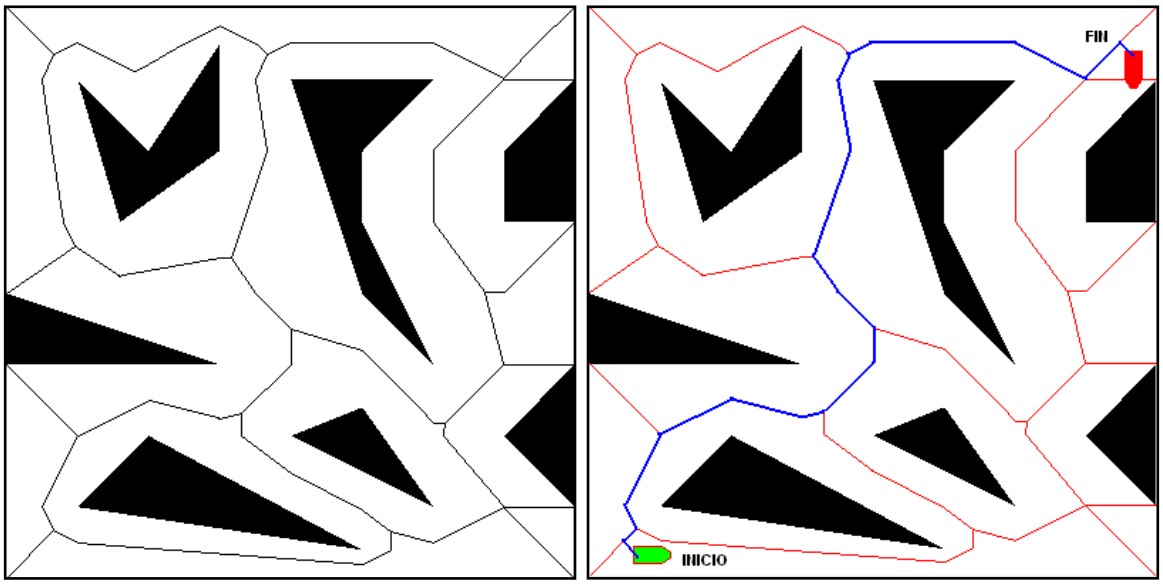


Figura 3.14: Subgrafo de una celda.

Al principio, cuando las funciones de flanco están definidas de acuerdo a los obstáculos que tocan, las entradas corresponden a los máximos de la función (vértices de los triángulos isósceles). Después, conforme la información métrica se va propagando, estos vértices son actualizados.

Cabe destacar que si un obstáculo presenta concavidades, las celdas se construyen en ellas de igual modo. Los segmentos adyacentes a los límites de las celdas se pueden tratar como obstáculos independientes, marcando entradas en los límites entre éstos, aunque fueran lados de un mismo obstáculo. Esto tiene como consecuencia que el diagrama de VODEC se prolonga por estas oquedades, incluso laberínticas si lo fueran, sin necesidad de modificar un ápice el algoritmo.

Un diagrama de Voronoi tradicional podría lograr este propósito, pero tendría que asignar regiones de Voronoi a cada segmento de cada polígono por separado (como si cada lado fuera un obstáculo independiente). Así, entre cada par de segmentos consecutivos existe necesariamente una arista de Voronoi que comienza en su vértice común. Esto agrega muchas aristas innecesarias (zonas convexas del polígono, aristas dentro de *CB*) al resultado final. Por tanto, esta opción de VODEC resulta específicamente adecuada para la planificación. El resultado de VODEC puede observarse en la fig. 3.15a.



a) *b)*

Figura 3.15: Resultado ofrecido por VODEC.

Hasta aquí el resultado obtenido corresponde al diagrama de Voronoi, y como tal puede utilizarse para otras aplicaciones. Sin embargo el objetivo presente consiste en la obtención de una trayectoria. Se agregan pues los puntos inicial y final como nodos del grafo de Voronoi, y se conectan al punto más próximo del grafo dentro de la celda que respectivamente los contiene. A continuación se busca dentro del grafo el conjunto de nodos y arcos que une dichos puntos entre sí, usando para ello alguno de los algoritmos clásicos como el denominado A* (Nilson, 1980). El resultado se muestra en la fig. 3.15b.

3.4. Mejoras aportadas por VODEC para la planificación

Tal y como se explica al final de la sección 3.3.2, para obtener una trayectoria entre el punto de inicio y el final, primero se halla el diagrama de Voronoi completo y después se selecciona dentro de éste el camino a seguir. Sin embargo, no todas las celdas participan de la trayectoria, tampoco todos los obstáculos del escenario influyen en la solución. Es decir, bastaría con calcular el diagrama que discurre por las celdas afectadas. Gracias a la capacidad de VODEC de restringir el cómputo del diagrama de Voronoi dentro de las celdas, es posible realizar un diagrama de Voronoi parcial. Para

ello se proponen a continuación cuatro métodos. Los dos primeros tienen carácter instructivo para servir de introducción a los otros. El tercero obtiene la solución exacta. El último ofrece una solución rápida con un algoritmo más sencillo de implementar.

El primer método que se propone, el más intuitivo, sería el siguiente. En primer lugar se realiza la descomposición vertical, obteniéndose el grafo de adyacencia. Desde este grafo se selecciona una secuencia de celdas comprendida entre la que contiene el punto de partida y la que contiene el punto de destino. A continuación se ejecuta VODEC, con la salvedad de que se parte únicamente de dichas celdas (la lista *A* únicamente contiene la secuencia) hasta equilibrarlas sin considerar ninguna otra. Por último se determina el camino final a partir del diagrama parcial de Voronoi calculado. La solución a este procedimiento se muestra en la fig. 3.16. Las celdas *A* y *C* contienen los puntos de partida (*m*) y meta (*n*) que se desean unir con una trayectoria. La secuencia de celdas en el grafo de adyacencia sería {*A, B, C*}. La línea de trazo grueso constituye la solución utilizando el procedimiento planteado.

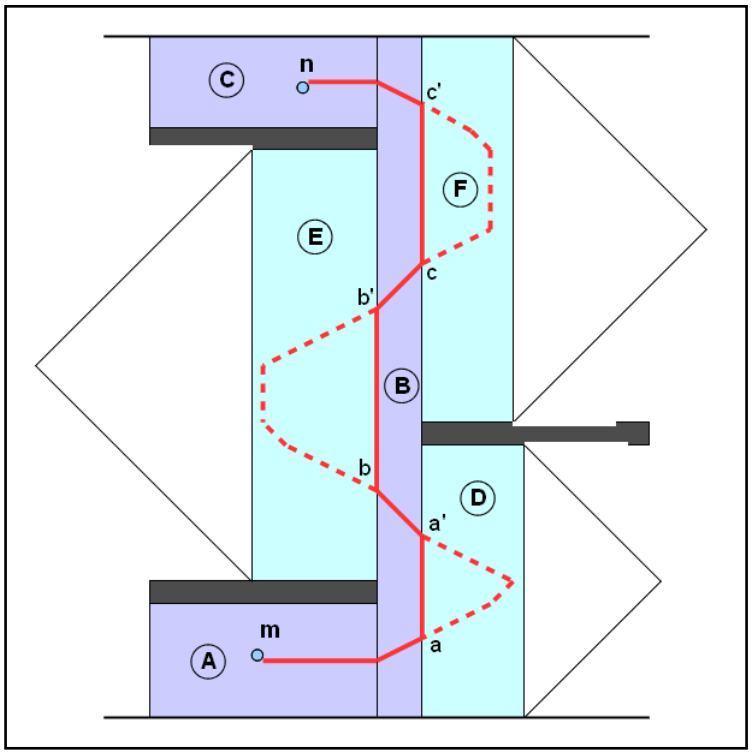


Figura 3.16: Fragmento del escenario atravesado por la trayectoria.

Esta solución es muy rápida y sencilla, aunque no constituye la solución dada por un diagrama de Voronoi. No obstante puede apreciarse que, dentro de las celdas de la

secuencia, el método trata de alejarse de los obstáculos en lo posible, mejorando la solución natural de la descomposición vertical. Esta última uniría el punto medio del segmento común a las celdas *A* y *B*, con el punto medio del segmento común a las celdas *C* y *B*, devolviendo una trayectoria que discurriría pegada a dos de los obstáculos del escenario (los situados a la izquierda de la celda *B*).

Otra propuesta sencilla (segundo método) consiste en, partiendo de la lista con la secuencia de celdas anterior, ejecutar el algoritmo principal pero agregando cualquier celda invadida a la lista. Es decir, se calculan sólo las celdas iniciales, y a partir de éstas todas aquellas que en las sucesivas iteraciones resulten invadidas. En la misma figura 3.16 puede verse el resultado de este método en la línea discontinua. En la solución del método anterior hay tramos (segmentos $\overline{aa'}$, $\overline{bb'}$ y $\overline{cc'}$) que implican invasiones no resueltas y que por tanto ofrecen puntos más próximos a los obstáculos de lo que podría obtenerse. Las celdas invadidas se denotan como *D*, *E* y *F*. Al calcular el subgrafo de las mismas se obtiene una solución mejor (más alejada de los obstáculos).

No obstante, con éste método podrían calcularse innecesariamente muchas celdas. Es decir, celdas invadidas pero cuyo cómputo no afecta en absoluto al camino final. Sería más útil determinar el conjunto de celdas realmente involucrado en el camino final, y sólo calcular ésas.

Con el propósito de no propagar un cómputo de celdas innecesario (tercer método), puede seguirse el efecto que los obstáculos de las celdas iniciales generan en las demás, y detenerse cuando desaparezca este efecto. Es decir, se considera el intervalo de invasión y la dirección del mismo (por ejemplo y de nuevo en la figura 3.16 serían los intervalos $\overline{aa'}$ y $\overline{cc'}$ hacia la derecha, y $\overline{bb'}$ hacia la izquierda). Al calcular la celda afectada, el cambio en su función de flanco debido a la invasión podrá afectar al flanco opuesto o no. Si no lo hace, significará que la región de Voronoi del obstáculo acaba en esa celda, y por tanto su límite discurrirá por ésta (caso de las celdas *D, E* y *F* de la figura). Si lo hace, ocurrirá en un intervalo inferior o igual al de partida (gracias al teorema 4 las aristas de Voronoi tendrán que mantenerse horizontales a menos que otros obstáculos aparezcan, y si aparecen su efecto será siempre en detrimento de la región de Voronoi que se está extendiendo), entonces será necesario calcular la siguiente, y así sucesivamente.

Por otro lado, una vez equilibradas las celdas afectadas, pudiera haber obstáculos aledaños que afecten a la trayectoria buscada, pero que por estar contiguos a

celdas no calculadas no fuesen detectados. En la figura 3.17 pueden observarse dos obstáculos, uno que sí influye (obstáculo α) y otro que no (obstáculo β).

Es fácil observar que tal circunstancia puede detectarse escrutando la función de flanco abatida sobre el plano. Si el obstáculo interseca la región comprendida entre el flanco y la función de flanco, entonces influirá en el resultado (caso del obstáculo α). Si está fuera significa que la distancia desde el obstáculo al flanco es superior a la detectada en la función de flanco, y por tanto su región de Voronoi no alcanza a la celda considerada (caso del obstáculo β).

Dado que las funciones de flanco están acotadas por las distancias a sus extremos, su valor máximo (altura del triángulo abatido) será de *h*/2, siendo *h* la altura del flanco. Por tanto sólo han de explorarse las celdas aledañas hasta que su anchura agregada supere dicho término (*h*/2).

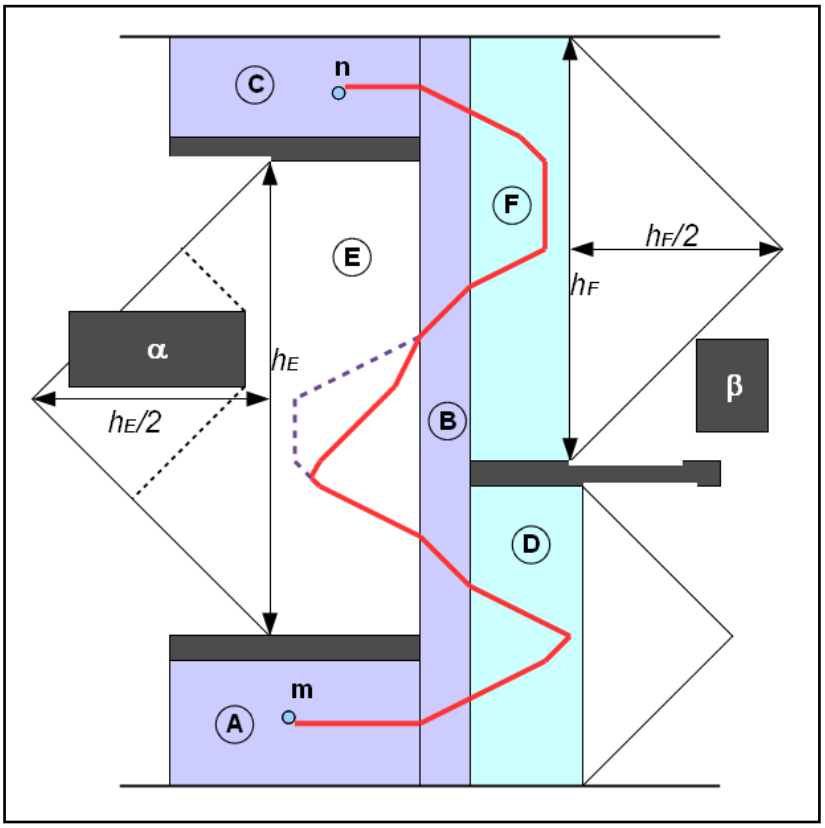


Figura 3.17: Efecto de obstáculos externos a las celdas calculadas.

En la figura 3.17 se han indicado las alturas de dos de estos flancos. En el flanco derecho de la celda *F* sería h_F, y como puede observarse, las celdas adyacentes a *F*

hasta una anchura de $h_F/2$ serían exploradas. El flanco izquierdo de la celda *E* tendría una altura h_E, y las celdas colindantes con el obstáculo α, puesto que entrarían dentro de la distancia $h_E/2$, serían exploradas.

Resumiendo, este último método parte de la secuencia de celdas que une la celda de partida con la de llegada, es decir, que contienen las posiciones inicial y final del robot. Después se procede a equilibrar estas celdas y, si existen invasiones a otras celdas no consideradas, se calculan también en un proceso recursivo siempre y cuando las invasiones estén relacionadas con los obstáculos detectados en la secuencia inicial. Una vez resueltas estas invasiones, y partiendo de las celdas que contienen un flanco común con celdas inexploradas, llámese flanco frontera, se procede a computar aquellas que se encuentren dentro de una distancia horizontal dada. Esta distancia es igual a la mitad de la altura de cada flanco frontera.

Con este método se toman en consideración todos los obstáculos que pudieran afectar a la trayectoria, es decir, su solución coincide con la de VODEC aplicado al escenario completo. Sin embargo, a efectos prácticos, puede no resultar necesaria tanta precisión. Dependiendo del robot, sus dimensiones y sus características, habrá una distancia de alejamiento suficiente para poder incorporar cualquier maniobra. Más allá de esta distancia, el objetivo de alejarse de los obstáculos pierde su sentido, con lo que el esfuerzo computacional invertido es innecesario.

Así pues, considérese una distancia fija *w* de exploración sobre las celdas de la secuencia inicial. La idea no es buscar el camino exacto, pero sí un camino de puntos alejados de los obstáculos garantizando la posibilidad evasión hasta una distancia dada. La diferencia con la solución real consiste en un acercamiento acotado hasta un máximo de *w*/2 hacia los obstáculos en el peor de los casos. El procedimiento (cuarto método) consiste en expandir la lista de celdas a computar añadiendo las aledañas por cada flanco frontera de las iniciales hasta una distancia *w* (en el sentido del eje horizontal). En los nuevos flancos frontera, se impone una función de flanco nula en todos los puntos (como si un obstáculo adyacente las cerrara por completo). De este modo cualquier obstáculo existente queda solapado con la nueva función de flanco, y la solución responderá al espacio libre estrictamente explorado.

En la figura 3.18 puede observarse dicho método. Los obstáculos *α, β* y γ, corresponden a obstáculos virtuales considerados al asumir una función de flanco nula. La distancia de exploración tomada obliga al cálculo de las celdas *G* y *H*, aunque en este

caso no afectan al camino hallado. Por último la celda *E* también es calculada, aunque no por efecto de la invasión desde *B*, sino por estar dentro del espacio de exploración fijado con w. En este caso, el cierre del escenario por este lado con el obstáculo *α* sí que afecta a la solución, acercándola a los obstáculos reales.

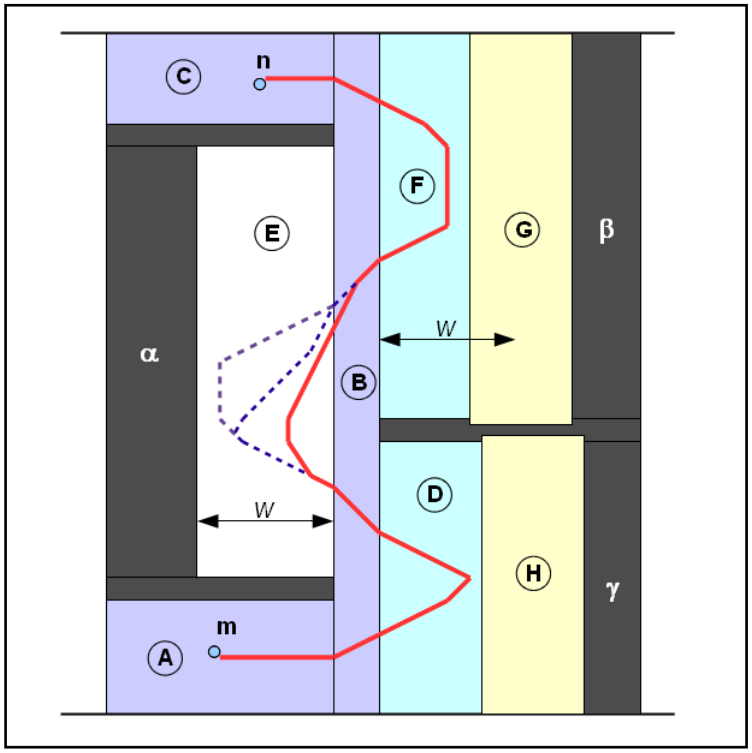


Figura 3.18: Efecto de obstáculos externos a las celdas calculadas.

En el caso extremo de que *B* fuera infinitamente delgada y *E* coincidiera con *w*, este acercamiento quedaría limitado al semiancho de *E*, es decir *w*/2. Si a efectos prácticos dicha distancia es suficiente para permitir maniobrar con holgura, la solución resultará útil y rápida.

3.5 Aplicación de VODEC a escenarios dinámicos

En las tareas de planificación robótica es usual enfrentarse a escenarios dinámicos, es decir, entornos donde algunos de los obstáculos pueden moverse, aparecer o desaparecer del mismo. Uno de los problemas que presentan los métodos de tiempo computacional óptimo es que cualquier cambio en el escenario implica

necesariamente el cálculo completo del mismo. De los métodos clásicos expuestos, sólo el de construcción incremental enunciado en Green y Sibson, (1978) e implementado por Ohya y otros, (1984) y por Sugihara e Iri, (1992), resulta eficaz en estos casos. Dicho método utiliza la triangulación de Delunay sobre un escenario de puntos, de modo que un nuevo punto se inserta en el grafo buscando a cuál de los triángulos pertenece. A partir de ahí se realiza un procedimiento mediante el cual el diagrama es actualizado. El procedimiento completo sin embargo no es óptimo ($O(n^2)$).

Con VODEC puede hacerse algo similar, pero a nivel de polígonos completos. En primer lugar se determina el conjunto de celdas afectado por el cambio (aquellas intersecadas o adyacentes al obstáculo en sus posiciones inicial y final). A continuación se eliminan y se reemplazan con celdas nuevas a partir de los nuevos datos (el conjunto de vértices del nuevo obstáculo o del que se está moviendo). Por último se calculan y se resuelven las invasiones de flanco. Estas invasiones, claro está, podrán extenderse más allá del conjunto de celdas recién definidas, pero se detendrán allí donde el cambio no influya. Es decir, VODEC ha de calcular únicamente un número limitado de celdas, las estrictamente afectadas por el cambio. Esto último permite una gran celeridad de respuesta respecto a métodos que deban calcular el grafo completo.

Como ejemplo véase la figura 3.19, donde se ha supuesto el caso de un obstáculo móvil desplazándose por un escenario con obstáculos fijos. Dicho obstáculo móvil se representa en la figura 3.19a en color azul, junto a los obstáculos fijos en negro y a la solución dada por VODEC en rojo. En la figura 3.19b se vuelve a mostrar el robot en el mismo color superpuesto a la posición que ocupará en verde claro. Ambos polígonos (huella presente y huella futura del robot) afectarán a las celdas adyacentes o secantes a los mismos. Es decir, el área representada en las figuras 3.19b, c y d en color amarillo. Una vez determinada dicha área, se eliminan todas las celdas que la definen (fig. 3.19c). A continuación, se determinan las nuevas celdas (fig. 3.19d). Sobre las mismas se procede al cálculo de la línea de cresta.

Tal y como define el algoritmo podrán suceder invasiones entre las propias celdas recién creadas o incluso a las demás. En tal caso se ejecuta el procedimiento recursivo hasta lograr el equilibrado completo de las celdas, y por tanto la solución (fig. 3.19e).

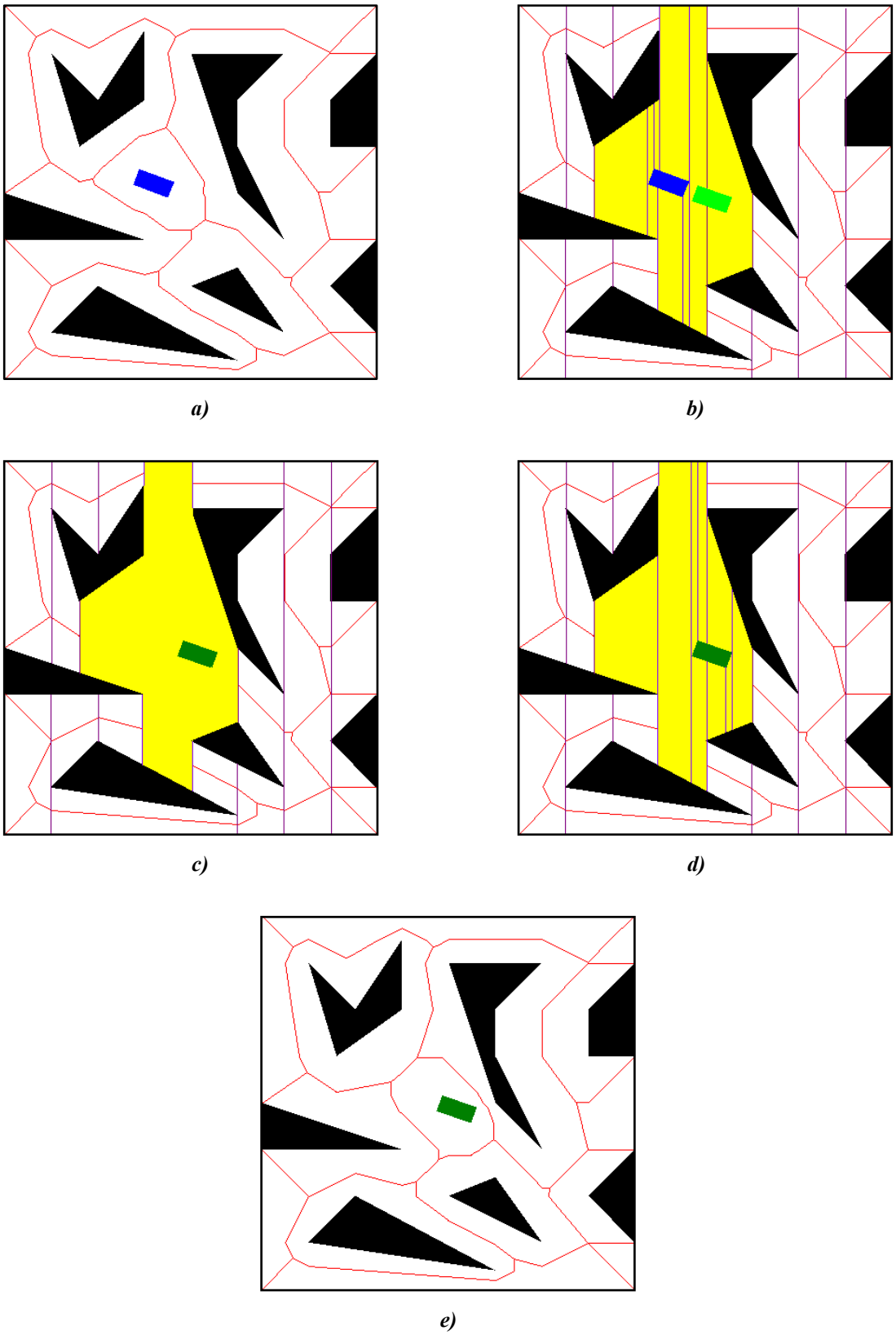
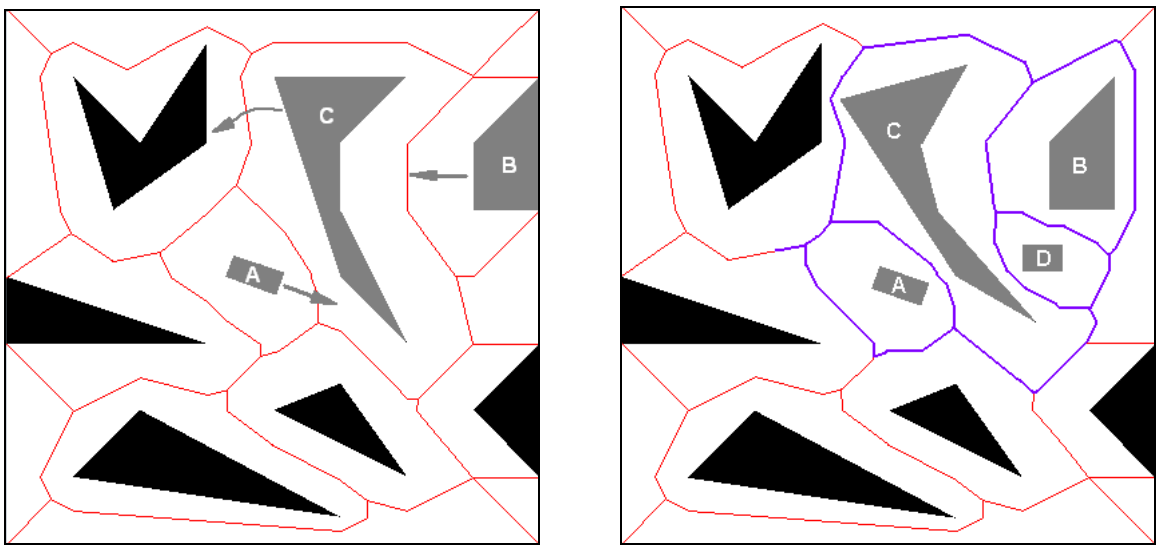


Figura 3.19: Proceso de actualización con obstáculos dinámicos.

En la figura 3.20 se muestra un esquema del procedimiento seguido. La figura 3.21 ilustra otro experimento con cambios más severos en el escenario. En concreto son tres los obstáculos desplazados, denotados con las letras *A, B* y *C*. Además existe un cuarto que es detectado por primera vez (obstáculo *D*). En la figura 3.21a se representa el escenario antes del cambio, con los obstáculos móviles en color gris y el diagrama de Voronoi resuelto con líneas rojas. En la figura 3.21b se muestra el efecto tras los cambios realizados. En color morado se han representado las nuevas aristas de Voronoi.

1. *Sea L una lista vacía.*
2. *Determinar las celdas intersecadas o colindantes con los obstáculos móviles*
3. *Redefinir nuevas celdas considerando la posición final de los obstáculos móviles.*
4. *Agregar estas celdas a la lista L.*
5. *Ejecutar VODEC a partir de L hasta que todas las celdas estén equilibradas.*
6. *Siguiente iteración*

Figura 3.20: Esquema del proceso de actualización con obstáculos dinámicos.



a) *b)*

Figura 3.21: Alteración del diagrama de Voronoi por varios obstáculos.

3.6. Propiedades

3.6.1. Tiempo de cómputo

En las estimaciones de tiempo de cómputo de los algoritmos clásicos, se parte de una cuantificación del problema planteado en base a los vértices de los obstáculos existentes, en adelante el parámetro *n*. Tras lo visto en la sección 3.4.1, puede deducirse que la velocidad de cómputo de VODEC en su aplicación a la planificación de trayectorias escapa a una comparativa tradicional. Ello se debe a que VODEC no necesita procesar todo el escenario, sino tan sólo las celdas afectadas, y por tanto resultará rápido. No obstante, se ofrece a continuación una valoración de su tiempo de cómputo en su aplicación como método de obtención del diagrama de Voronoi completo.

En este apartado se seguirá la notación de G. Brassard (1996) en cuanto a límites de tiempo de cómputo en algoritmos. El primer paso de este método es la clasificación de los vértices de obstáculos por abscisa creciente para poder realizar la descomposición vertical. Esto se logra en un tiempo del orden de O*(nlogn*) siendo *n* el número total de vértices. A continuación de definen las celdas y sus funciones de flanco. Cada celda tiene dos flancos y cada uno ha de tocar al menos un vértice. Por otro lado, un vértice puede ser común a un máximo de tres celdas. Inicialmente, las tareas asociadas a la construcción de la celda, como la definición de la función de flanco, dependen únicamente de los mismos vértices implicados en la definición de la celda. Por lo tanto la duración de esta fase se mantiene dentro de O(*n*).

La segunda fase trata las invasiones e implica el cálculo de la celda tantas veces como éstas se produzcan. Ha de tenerse en cuenta que la ordenación de vértices genera de forma natural una ordenación de celdas. Así, en la lista de celdas resultantes no habrá nunca una celda seguida de otra que la flanquee por el lado izquierdo; todas quedarán dispuestas de izquierda a derecha. Esto significa que, si se calcula su línea de cresta siguiendo este orden, todas las invasiones que se produzcan hacia la derecha quedarán resueltas. Una vez concluido el cómputo de todas las celdas, quedarán en la lista *A* sólo aquellas celdas afectadas por invasiones pendientes hacia la izquierda. Del mismo modo, puede procederse a calcular de nuevo las líneas de cresta siguiendo un orden inverso. Así, cada celda habrá de procesarse un máximo de dos veces.

Cada invasión implica la incorporación de nuevos vértices a las funciones de flanco, con lo que el tamaño de estas funciones sólo está limitado por *n*. En concreto, si *k* es el número de celdas invadidas y a_i es el número de vértices implicado en la *i*-ésima celda calculada, el tiempo de esta fase será proporcional a $\sum_{i=1}^{k} a_i$ Ha de tenerse en cuenta que a_i sólo comprende aquellos vértices responsables de la invasión, dado que merced al teorema 4 el límite de la Región de Voronoi es horizontal, y por tanto no afecta al resto de la celda. Si se define *l* como un valor medio del número de vértices de las celdas a calcular, entonces el tiempo de cómputo será del orden de O(*k*·*l*).

En el peor caso los *n* vértices tendrían influencia en todas las celdas. Esto significaría que el proceso de cálculos de celdas tendría un coste global dentro de O(n^2). En el mejor de los casos no se producirían invasiones y el coste de esta fase sería nulo.

El último proceso consiste en la extracción del Diagrama de Voronoi final. Para ello, en cada celda se procesa su línea de cresta, se calculan las regiones de los obstáculos superior e inferior y se añaden las aristas horizontales. Los tres procesos recorren los vértices de las funciones de flanco. Por tanto, el coste computacional de esta fase pertenece a O(n^2), donde se utiliza *n* como cota superior del número de celdas, y del número de vértices de los flancos de la celda. Al igual que sucede con el método de inserción, si los obstáculos se hallan regularmente distribuidos, puede considerarse un valor constante *l* como media de los vértices de la función de flanco, con lo que el tiempo de cómputo sería proporcional a *n*·*l*.

Por tanto, de la agregación de las tres fases del proceso podemos estimar un rango de operación situado entre $\Omega(nlogn)$ y O(n^2). El primer término se debe a que la ordenación no puede realizarse en un tiempo inferior al expresado, y las otras fases podrían comportarse como lineales si existe distribución regular. El segundo corresponde a cualquiera de las otras dos fases, que son las más lentas en el peor de los casos.

3.6.2. Computación paralela

Hoy en día es usual contar en cualquier PC con tarjetas gráficas multiprocesadoras, con capacidad para ejecutar varios hilos de programación de forma simultánea. Por tanto, algoritmos no óptimos pueden resultar más rápidos en la práctica debido a su capacidad de procesado paralelo. La singularidad que presenta VODEC al

restringir las operaciones dentro de las celdas, puede aprovecharse para realizar fácilmente una implementación informática multihilo.

En concreto se proponen dos formas de llevar a cabo esta implementación paralela. La primera y más simple consiste en elaborar una lista en cada iteración con las celdas que precisan ser calculadas. De dicha lista se extrae la primera celda y se abre un hilo para su cálculo. Inmediatamente se apartan de la lista todas sus celdas adyacentes. De la lista restante se vuelve a tomar una celda y se realiza la misma operación. De este modo pueden procesarse en paralelo todas aquellas celdas que no compartan sus flancos, y por tanto las invasiones que surjan quedarán almacenadas en posiciones de memoria disjuntas de las de otros hilos. En la lista de la siguiente iteración se van almacenando las celdas apartadas y las celdas invadidas. El procedimiento continuará mientras queden celdas en la lista, computando en paralelo tantas como le es posible. Este sistema requiere cambios mínimos sobre el algoritmo original.

La segunda forma, es más eficiente aunque más compleja. Consiste en definir de forma redundante los flancos, de modo que el resultado de las invasiones se almacenen en las copias de éstos. Así, todas las celdas, ya sean contiguas o no, podrán procesarse en hilos independientes que guardarán sus resultados en unas funciones de flanco provisionales. Tras cada iteración, los flancos que han sufrido invasiones son examinados cotejando los resultados obtenidos a cada lado del mismo. El algoritmo descartará los valores más altos quedándose con la función de flanco mínima. Esto implica la detección de nuevas invasiones y por lo tanto el ingreso de la celda o celdas correspondientes de nuevo en la lista.

3.6.3. Áreas equidistantes

La métrica Manhattan tiene algunos inconvenientes como ya se señala en Aurenhammer, (1991). Consiste en la posible existencia de áreas de equidistancia en lugar de líneas. Esto puede observarse en la fig. 3.15a donde una determinada configuración de obstáculos (los obstáculos de tamaño puntual *R* y *S*) provoca dicha situación. VODEC lo resuelve escogiendo la línea inscrita en la celda. Así no se detectan invasiones innecesarias y se agiliza el algoritmo. En cuanto al error cometido entra dentro del error general por asumir dicha métrica en lugar de la euclídea.

En la figura 3.15a puede observarse dicho error entre el punto escogido por VODEC (punto *P'*) y el obtenido de un Voronoi de métrica euclídea (punto *P*). El error máximo dependerá del tamaño del escenario. Supóngase un escenario cuadrado de lado *L*. Para maximizar el área de equidistancia propia de la métrica Manhattan, los dos obstáculos *R* y *S* estarían infinitamente próximos a una esquina del escenario, siendo toda el área del mismo equidistante en métrica Manhattan. Sin embargo, las regiones de Voronoi de los límites del escenario lo cubrirían, eliminándola.

Para buscar el máximo error se hace coincidir la mediatriz de los vértices *R* y *S* con la diagonal del escenario, sobre la cual se les ubica recorriéndola progresivamente. Por un lado se irá reduciendo el área de equidistancia, pero por el otro se irá disminuyendo las regiones de Voronoi de los límites del escenario. El resultado es que el error inicialmente aumenta tal y como se muestra en las figuras 3.15b, 3.15c y 3.15d. Se define el error máximo como el existente entre el peor punto escogido por VODEC y el propio de un diagrama de Voronoi de métrica euclídea, denotado como *e* en las figuras 3.15. A partir de la posición de la figura 3.15d, donde el error alcanza su valor máximo ($e = \sqrt{2} \cdot L/6$) comienza a disminuir para posiciones más avanzadas de *R* y *S* sobre la diagonal.

Respecto a otros obstáculos y configuraciones ha de resaltarse el hecho de que la forma del obstáculo es irrelevante siempre que sus lados no influyan en el área de equidistancia, es decir, que sólo los vértices de *R* y *S* sean los puntos más próximos a dicha área. Por otro lado la separación de los obstáculos de la diagonal (fig. 3.15e) o la confrontación sobre otras líneas (fig. 3.15f) lleva a valores del error inferiores. En el primer caso el flanco derecho de la celda común a *R* y a *S* se desplaza hacia la derecha arrastrando el punto *P* y disminuyendo el error. En el segundo, según sea el desplazamiento sobre el máximo de *R*, o bien se reduce *RP'*, o bien se reduce *RP*, siempre en detrimento del error *PP'*.

3.7 Conclusiones

En este capítulo se ha presentado un nuevo método para la realización de diagramas de Voronoi a partir de la descomposición vertical. Se ha justificado el procedimiento y se han mostrado ejemplos de su aplicación. Se ha realizado un especial

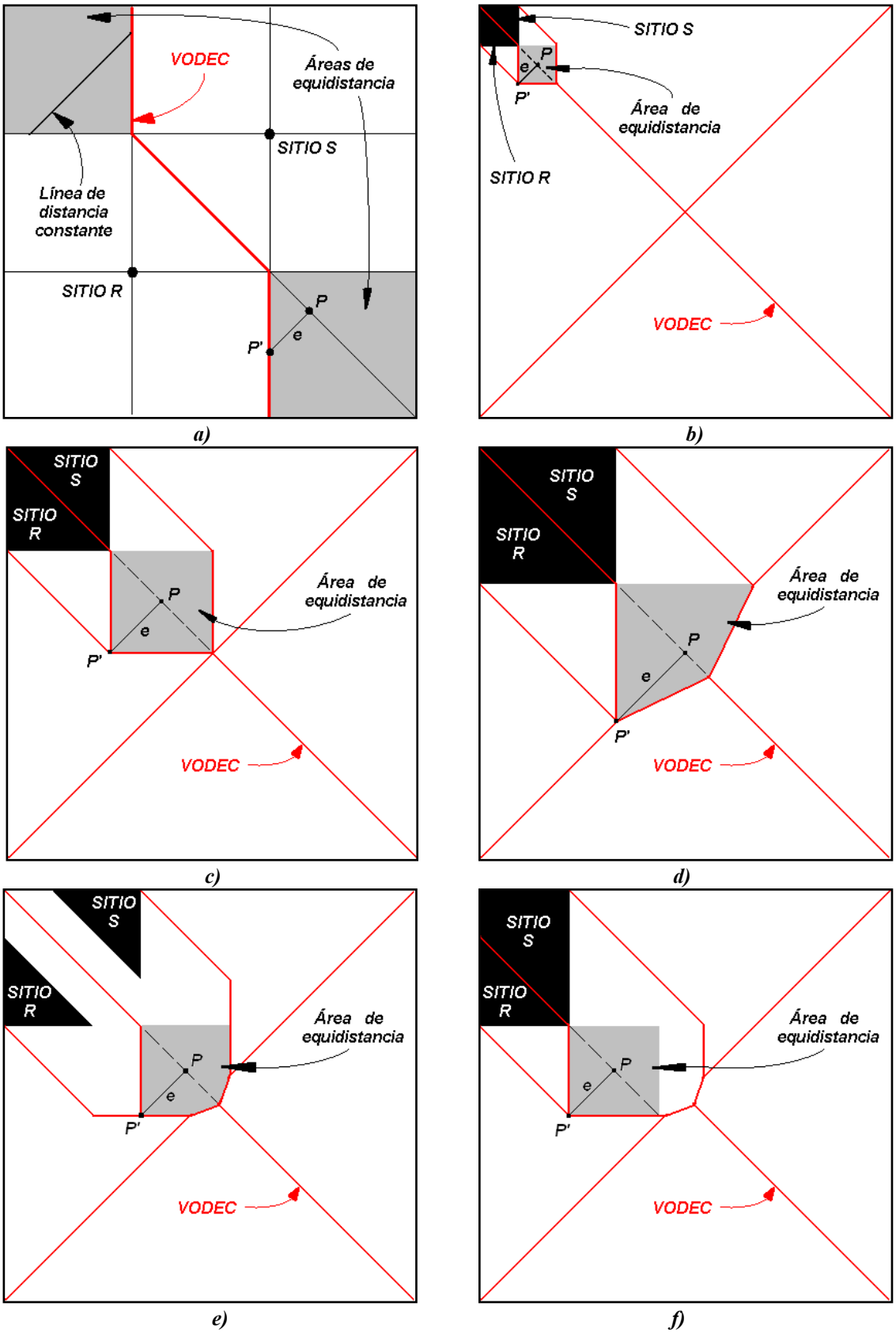


Figura 3.15: Área de equidistancia en métrica Manhattan

hincapié en el caso de escenarios dinámicos, y se ha estudiado la posibilidad de restringir el cálculo a una secuencia de celdas con vistas a la obtención de trayectorias.

Las ventajas más notables de este método son su sencillez, al escoger una métrica no euclídea, y su capacidad para realizar cálculos parciales del diagrama de Voronoi. Ambas cualidades son de especial aplicación a la planificación de trayectorias, donde interesa un cierto alejamiento de los obstáculos sin excesiva precisión.

En particular, la capacidad de realizar cálculos parciales del diagrama de Voronoi resulta ventajosa en escenarios dinámicos, frecuentes en robótica. Por otra parte, también la generación de trayectorias se beneficia de esta propiedad, ya que sólo las celdas involucradas han de ser calculadas.

Por todo ello VODEC resulta un algoritmo rápido y sencillo, especialmente apropiado para tareas de planificación.

CAPÍTULO 4 Planificación de trayectorias y maniobras en sistemas no holónomos

4.1 Introducción

Hasta ahora se han detallado los diversos sistemas de planificación existentes, tratando el problema desde un enfoque global, teniendo como únicos elementos el espacio de configuraciones (*C*), los obstáculos y el espacio que generan dentro de *C* (*CB*), el espacio de configuraciones libres de colisión (C_{free}) y nada más; dando por supuesto que toda la casuística del problema podía parametrizarse utilizando sólo estos elementos. Sin embargo la mayoría de los sistemas presentan algún tipo de imposición que complica el hallazgo de trayectorias válidas.

En este capítulo se estudiará con especial interés el caso de vehículos con restricciones no holónomas, que puede surgir en los procesos de planificación. Se describe así mismo un método, denominado postprocesado, que utiliza los planificadores estudiados para contender con este tipo de restricciones.

En el caso de vehículos rodados, suele asumirse la simplificación de un escenario plano donde se modela la configuración del vehículo como su posición cartesiana (la de su centro de referencia) y su orientación. El medio más intuitivo de representar las trayectorias consiste en trazar sobre el plano la línea que recorre dicho centro de referencia. Debido a la morfología característica de esta línea resulta útil definir algunos conceptos (Latombe, 1991). Se denomina *inversor* al punto de la trayectoria de un vehículo en el que la velocidad de su centro de referencia cambia de sentido. Se define *maniobra* como una trayectoria en la que aparece al menos un inversor. Los inversores provocan la detención completa del vehículo para iniciar un nuevo movimiento en sentido contrario. Es decir, implican un proceso de deceleración o frenado y aceleración posterior, con el consecuente consumo energético. Estos procesos, además, suelen conllevar deslizamientos, lo cual deteriora la calidad de la odometría, afectando al proceso de control del robot. Por todo ello no son deseables. No obstante, según los requerimientos del problema planteado pueden ser imprescindibles. Así pues, y de forma genérica, se buscarán soluciones que contengan el mínimo número de maniobras posible, a fin de reducir los inversores.

4.2 Restricciones cinemáticas

Tal y como se describe en los capítulos precedentes, la posición de cada punto de un robot puede hallarse en función de la serie de parámetros q_i, que agrupados componen el vector de configuración $\boldsymbol{q}$. El espacio C es el conjunto de todas las configuraciones $\boldsymbol{q}$ posibles.

Considérese ahora la existencia de restricciones. Éstas, en su forma más general, consistirán en una relación entre las variables de configuración. Dado que éstas suelen identificarse con posiciones y orientaciones de los sólidos rígidos que componen el robot, las derivadas de estas variables con respecto al tiempo tienen el sentido físico de velocidades. La naturaleza de las restricciones cinemáticas y dinámicas puede formalizarse mediante expresiones (Latombe, 1991):

$$F_j(\dot{q}_1, \dot{q}_2, ..., \dot{q}_n, q_1, q_2, ..., q_n, t) = 0 \qquad ; j = 1, .., m \tag{4-1}$$

Donde *n* es el número de variables del espacio de configuraciones y *m* es el número de restricciones. Si estas restricciones son integrables entonces se dice que son "holónomas". En cambio si no lo son, se denominan restricciones "no holónomas". Concretamente, dado un conjunto de restricciones, se puede determinar si son holónomas mediante el teorema de integrabilidad de Frobenius (Spivak, 1979). Los sistemas que están sometidos a tales restricciones también pueden adquirir el apelativo de sistemas holónomos o no holónomos según la naturaleza de sus restricciones.

Si las restricciones son holónomas, entonces (4-1) podría integrarse, con lo que podría obtenerse una expresión del tipo (Gómez-Bravo, 2001a):

$$G_j(q_1, q_2, ..., q_n, t) = 0 \qquad ; j = 1,..,m \tag{4-2}$$

A partir de (4-2) sería posible encontrar m ecuaciones que expresen m variables en función de las restantes (n-m) (Gómez-Bravo, 2001a):

$$q_j = g_{,}(q_{m+1}, q_{m+2}, ..., q_n, t) = 0 \qquad ; j = 1,..,m \tag{4-3}$$

Así, sólo es necesario definir el conjunto de parámetros $(q_{m+1}, q_{m+2}, ...q_n, t)$ para determinar las restantes variables y por tanto la configuración del robot. Esto implica que se puede reducir la dimensión del espacio de configuraciones en *m* variables. De esta forma el problema de planificación con restricciones holónomas equivale a un problema de planificación sin restricciones y con un menor número de variables. Al ser todas las variables independientes entre sí, el planificador puede generar cualquier tipo de trayectoria, cosa que como se verá no es posible en las no holónomas.

Cuando las restricciones son no holónomas, la dimensión no puede reducirse y sin embargo su presencia impone condiciones al movimiento del robot. En concreto, si se desea unir dos configuraciones dentro de *C* (sin obstáculos), las restricciones no holónomas limitarán el conjunto de trayectorias posibles, dado que la variación de cada una de las variables de configuración no es libre, sino que está condicionada a la de sus compañeras. En efecto, las restricciones no holónomas no reducen la dimensión de *C*, pero sí lo hacen en el espacio de las velocidades.

4.3 Caracterización cinemática de robots móviles rodados

Un ejemplo de restricción no holónoma puede observarse en cualquier sistema que utilice ruedas como medio de locomoción. Si se presta atención a una rueda individual ideal (indeformable y perfectamente circular), ésta se apoya en un punto de la superficie sobre la que se desplaza. Sobre este punto la rueda puede avanzar en un determinado sentido (el de rodadura) pero le está restringido el movimiento de deslizamiento en el sentido axial. En la figura 4.1 se explica esto más detalladamente (Gómez-Bravo, 2001a).

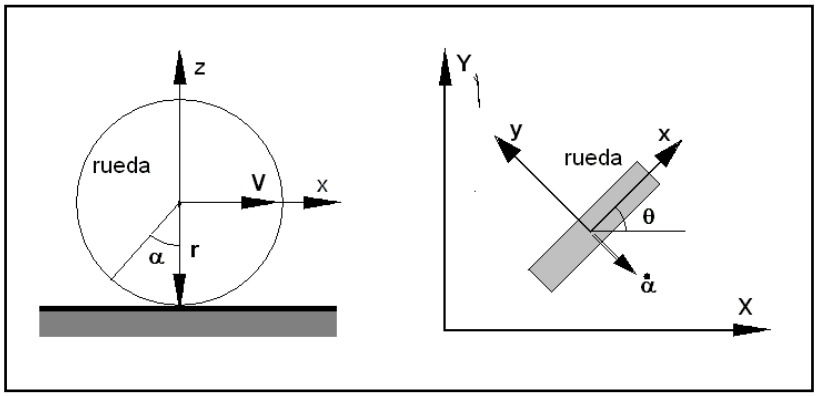


Figura 4.1: Esquema de la rueda

Sea (*X,Y,Z*) una terna de ejes ortogonales que constituyen el sistema de referencia fijo. Se define (*x,y,z*) como el conjunto de ejes cuyo centro está ubicado en el centro de la rueda, desplazándose con él. El eje *z* es ortogonal al plano en donde se apoya la rueda y el eje *x* apunta siempre en la dirección de rodadura. El radio de la rueda es *R* y cuando ésta se mueve, la velocidad de su centro se denota con *V*. Se sabe que la velocidad de cualquier punto *P* del perímetro de la rueda (V_p) es igual a la velocidad de su centro (V_o), mas la velocidad con la que dicho punto gira alrededor del centro (Gómez-Bravo, 2001a):

$$V_P = V_O + \dot{\alpha} \times r \qquad \textit{(4-4)}$$

Donde $\dot{\alpha}$ expresa el vector velocidad de rotación alrededor del centro de la rueda (punto *O*) y ***r*** es un vector de módulo igual al radio de la rueda, cuya base es *O* y su

extremo *P*. Si de entre los puntos del perímetro se escoge el punto de contacto con la superficie de apoyo, cuya velocidad es nula, obtenemos la siguiente condición:

$$0 = V_O + \dot{\alpha} \times r \qquad (4\text{-}5)$$

Expresados estos vectores con respecto al eje de coordenadas fijo se tiene:

$$V_O = \begin{bmatrix} \dot{x} & \dot{y} & 0 \end{bmatrix}^t \; ; \; r = \begin{bmatrix} 0 & 0 & -R \end{bmatrix}^t \; ; \; \dot{\alpha} = \begin{bmatrix} \dot{\alpha}_x & \dot{\alpha}_y & 0 \end{bmatrix}^t \qquad (4\text{-}6)$$

La resolución de la ecuación vectorial (4-5) queda por tanto descrita como dos ecuaciones escalares:

$$\dot{x} = \dot{\alpha}_y r \; ; \; \dot{y} = \dot{\alpha}_x r \qquad (4\text{-}7)$$

Como se observa en la figura 4.1 derecha, el vector $\dot{\alpha}$ es ortogonal al eje *x*; si se define un vector unitario ***u*** alineado con dicho eje, el producto escalar de ***u*** y $\dot{\alpha}$ será nulo:

$$u = \begin{bmatrix} \cos\theta & \sin\theta & 0 \end{bmatrix}^t \; ; \; \dot{\alpha} \cdot u = 0 \qquad (4\text{-}8)$$

El resultado de esta propiedad se resume en la siguiente ecuación escalar:

$$\dot{\alpha}_x \cos\theta + \dot{\alpha}_y \sin\theta = 0 \qquad (4\text{-}9)$$

Despejando $\dot{\alpha}$ del sistema de ecuaciones formado por (4-7) y (4-9) se obtiene:

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0 \qquad (4\text{-}10)$$

Como se puede comprobar aplicando el teorema de integrabilidad de Frobenius, esta restricción es no integrable y por lo tanto no holónoma. Todos los vehículos rodados presentan entonces restricciones no holónomas. En este documento se expondrá el tratamiento de estas restricciones centrándose en dos tipos de sistemas de locomoción muy extendidos: el tipo diferencial y el tipo “Ackerman”.

4.3.1 Cinemática del vehículo diferencial

Un vehículo diferencial consta de dos ruedas motrices paralelas de tracción independiente entre sí (ver figura 4.2). Esto le permite avanzar o retroceder en línea recta, desplazarse describiendo un radio de curvatura dado y sobre todo le permite cambiar su orientación sin desplazarse (esto se consigue haciendo girar las ruedas en sentidos opuestos). En cambio, las restricciones cinemáticas le impiden desplazarse en el sentido del eje de las ruedas. Muchos robots con sistema locomotor eléctrico, incorporan dos motores en sendas ruedas, adscribiéndose a éste tipo de vehículos. Esta abundancia es debida a la maniobrabilidad que les confiere la capacidad de girar ambas ruedas de forma independiente, facilitando la tarea de planificación a pesar de existir restricciones no holónomas como se describe a continuación.

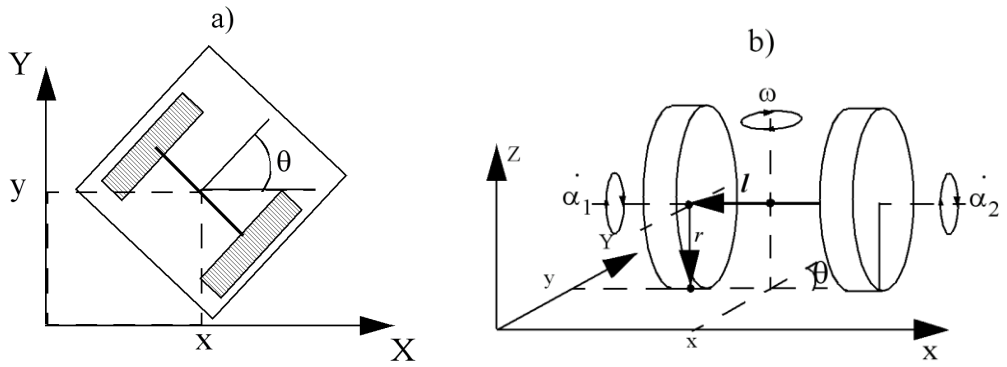


Figura 4.2: Esquema de un robot diferencial

Como ejemplo puede observarse el caso de un vehículo de planta rectangular y conducción diferencial cuya configuración puede determinarse mediante las coordenadas (*x,y*) de un punto de referencia (se suele escoger el centro del eje de las ruedas, ver figura 4.2a) y el valor de θ, que define la orientación del eje principal (o línea de avance) del vehículo. El movimiento de éste puede modelarse mediante la ecuación (Ollero, 2001):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{r\cos\theta}{2} \\ \frac{rsen\,\theta}{2} & \frac{rsen\,\theta}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \cdot \begin{bmatrix} \dot{\alpha}_1 \\ \dot{\alpha}_2 \end{bmatrix} \qquad (4\text{-}11)$$

En estos vehículos (ver figura 4.2b) la dirección del movimiento está controlada por las velocidades de rodado de cada una de las ruedas *(α_1, α_2)*. Por otro lado cada rueda está sometida a una restricción cinemática del tipo (4-10) que viene a significar que la componente axial de la velocidad del centro de cada rueda es nula. Ambas restricciones referidas al eje de coordenadas sito en el vehículo resultan ser la misma ecuación:

$$\dot{x}_0 \sin\theta - \dot{y}_0 \cos\theta = 0 \qquad \textit{(4-12)}$$

Esta ecuación implica la imposibilidad de mover el vehículo en una dirección distinta a la determinada por la orientación del vehículo; es decir, sólo se puede mover en el sentido de avance de las ruedas, nunca en el sentido de su eje de giro. Dado que no existe modo de integrar esta ecuación (Latombe, 1991), no es posible utilizarla para describir alguna de las variables en función de las demás. Por tanto la dimensión del espacio de configuraciones permanece igual, a pesar de que (4-12) vincula sus variables.

Esta restricción afecta al cálculo de trayectorias, dado que si se desea pasar de una configuración inicial (punto origen) hasta otra final (punto destino), la evolución desde la una a la otra no puede ser arbitraria (por ejemplo la línea recta), sino que ha de cumplir la restricción. En concreto, la ecuación (4-12) está limitando el espacio de las velocidades a un plano, es decir, hay que hallar una trayectoria cuyas velocidades en cada punto de la misma pertenezcan a dicho plano.

Este problema es común a todos los sistemas no holónomos, y para resolverlo no hay ninguna herramienta analítica que, atendiendo a una función de coste, proporcione una solución exacta. Tan sólo se dispone de un conjunto de métodos heurísticos más o menos eficientes según el caso, que intentan generar una trayectoria factible en un tiempo lo menor posible. Entre estos métodos como se verá, están las adaptaciones del RRT.

El espacio de las velocidades, también llamado espacio tangente (Latombe, 1991), representa, para cada configuración ***q*** de *C*, las velocidades que puede adoptar el sistema en dicho punto. Las restricciones no holónomas como la (4-12) implican una reducción de la dimensión de este espacio. De hecho, (4-11) presenta una matriz de dos columnas, con lo que el espacio de velocidades queda reducido al plano definido por la

combinación lineal de los vectores definidos por dichas columnas, es decir v_1 y v_2 (4-13 y 4-14):

$$v_1 = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{rsen\theta}{2} & \frac{r}{2l} \end{bmatrix}^t \qquad (4\text{-}13)$$

$$v_2 = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{rsen\theta}{2} & -\frac{r}{2l} \end{bmatrix}^t \qquad (4\text{-}14)$$

Por otro lado, cualquier trayectoria construida siguiendo estos vectores satisfará automáticamente la restricción. Ésta es la idea que subyace en la formulación de las maniobras restringidas que se describirán más adelante y que constituyen la base del método de planificación capaz de sortear dichas restricciones.

4.3.2 Cinemática del vehículo tipo Ackerman

Se dice que un vehículo tiene la configuración Ackerman cuando consta de dos ruedas fijas paralelas no orientables en un eje situado en la parte posterior, y otras dos ruedas orientables alineadas según un eje paralelo al anterior situado en la parte delantera del vehículo (ver figura 4.3). A este tipo corresponden los automóviles comunes y la mayoría de los vehículos con motor de explosión.

La abundancia de sistemas de locomoción que responden a este modelo aumenta considerablemente el interés y la importancia de los hallazgos científicos en materia de planificación al respecto. Cabe destacar la adaptación de automóviles comerciales para su uso como robots, como el robot Romeo4R, utilizado para los experimentos de esta tesis (Gómez-Bravo y otros, 2008a). También es reseñable el objetivo de lograr la conducción autónoma de automóviles en entornos urbanos para su uso en el transporte civil, como se ha estado estimulando en las sucesivas ediciones del concurso de robótica "*DARPA Grand Challenge*" (años 2004-2005) y *"DARPA Urban Challenge"* (2007).

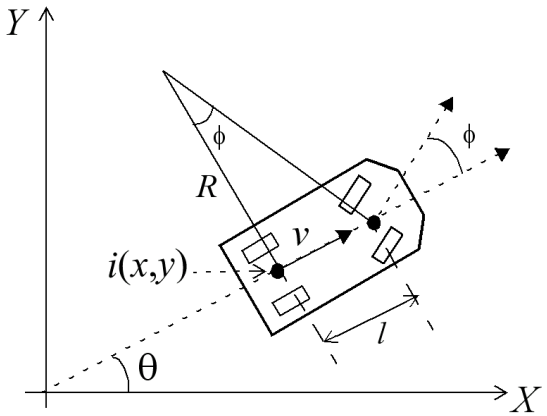


Figura 4.3: Esquema de un robot Ackerman

El comportamiento cinemático del vehículo tipo Ackerman comparte abundantes similitudes con las configuraciones de un triciclo (dos ruedas paralelas traseras no orientables y una delantera orientable) y de una bicicleta (una rueda trasera no orientable y otra delantera orientable), siendo las ecuaciones de su posición global y orientación iguales en los tres modelos. Ello se debe a que fundamentalmente estos vehículos se desplazan siguiendo un radio de curvatura determinado por el ángulo de giro de sus ruedas orientables. En el vehículo tipo Ackerman, la imposición de este radio de curvatura común a todas las ruedas, implica una restricción a los ángulos de giro de sus ruedas orientables. Restricción inexistente por otro lado en los otros dos tipos de vehículos (triciclo y bicicleta), debido a que sólo disponen de una rueda para determinar dicho radio de curvatura.

Los vehículos tipo Ackerman se caracterizan precisamente por la existencia de un sistema mecánico destinado a cumplir tal restricción. Sin embargo, desde el punto de vista del planificador, sólo interesa su comportamiento global, como se describe a continuación. En efecto, si se sustituyen las dos ruedas orientables por una situada en el punto medio (se pasa de la configuración Ackerman a la de un triciclo equivalente), el ángulo de ésta sería ϕ (ver figura 4.3). Si a continuación se determina la intersección de su eje con el de las ruedas traseras se obtiene el centro instantáneo de rotación del vehículo; también se obtiene el radio de curvatura *R*. Pero además, dada la ortogonalidad entre el eje de giro y la dirección de rodadura de la rueda orientable, se observa que el ángulo de dirección ϕ es exactamente el ángulo que forman sendos ejes (ver figura 4.3). Si se define *v(t)* como la velocidad en módulo del vehículo, y $\rho(t)$ como la curvatura en cada instante ($\rho(t)$=1/*R(t)*) se obtiene (Ollero, 2001):

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}v(t)\\ v(t)\rho(t)\end{bmatrix} \qquad (4\text{-}15)$$

$$\left|\rho(t)\right| < \rho_{\max}=\frac{\tan\phi_{\max}}{l}=\frac{1}{R_{\min}} \qquad (4\text{-}16)$$

$$\dot{x}\sin\theta-\dot{y}\cos\theta=0 \qquad (4\text{-}17)$$

Como puede observarse, (4-15) se refleja la descripción del movimiento del robot usando sólo dos variables de control. Así, en vez de tener como parámetros de control el par (α_1,α_2) como en el tipo diferencial, ahora se dispone de (v,ρ) que son respectivamente la velocidad del centro de referencia del vehículo y el radio de curvatura. Hay que hacer notar que éstas no son variables de configuración, pues con sólo la información ofrecida en ellas resulta imposible determinar la posición y orientación del vehículo. Sería necesario integrarlas en el tiempo desde una posición conocida hasta la que se desea saber. Ello se debe a la forma diferencial de (4-15).

En este tipo de vehículos existen dos ejes en cuyas direcciones el movimiento no es posible (la componente de la velocidad del centro de la rueda en su dirección axial es nula (4-10)). El primero lo conforma el eje de las ruedas traseras del vehículo. El segundo cualquiera de los ejes de las delanteras (ambos están condicionados para posibilitar el movimiento). Así, sólo es factible mover el vehículo siguiendo el radio de curvatura definido por la intersección de ambos ejes y el centro de referencia del vehículo. Aunque por supuesto dicho radio pueda variarse de forma continua.

El espacio de las velocidades, al igual que en el caso anterior, queda reducido a un plano. Este plano está definido por los vectores columna $\boldsymbol{w_1}$ y $\boldsymbol{w_2}$ de la matriz definida en (4-15):

$$w_1=\begin{bmatrix}\cos\theta & sen\theta & 0\end{bmatrix}^t \qquad (4\text{-}18)$$

$$w_2=\begin{bmatrix}0 & 0 & 1\end{bmatrix}^t \qquad (4\text{-}19)$$

Además de la restricción cinemática (4-17) es habitual encontrar otra restricción en este tipo de vehículos debido a las limitaciones físicas en cuanto al ángulo de

dirección (4-16), lo cual implica un radio de giro mínimo, es decir, la curvatura está acotada superior e inferiormente. No existe en cambio ninguna cota inferior, el radio máximo de giro no está limitado pues puede ser infinito, lo que equivale a una curvatura nula, o lo que es lo mimo, avanzar en línea recta.

Comparativamente con el anterior, este tipo de vehículos resulta más difícil de tratar a la hora de la planificación, pues presenta una restricción más a tener en cuenta. El efecto de la misma se hace patente en múltiples aspectos de la planificación. Por ejemplo, en tanto que un robot diferencial puede cambiar su orientación sin variar su posición cartesiana, un robot tipo Ackerman precisa abandonar dicha posición para poder reorientarse. Por tanto las maniobras necesarias para alcanzar una configuración dada, son más complejas, y el recorrido que necesitan suele resultar más amplio aumentando las posibilidades de colisión con obstáculos.

La reducción del espacio de velocidades a un plano de la que se habló en el apartado anterior vuelve a ser patente en este caso, observando una matriz diferente en (4-15) pero del mismo rango que la de (4-11). La restricción (4-16) en cambio acota la región de velocidades posibles dentro de un cono, pues también puede ser expresada de la siguiente forma (Latombe, 1991):

$$\dot{x}^2 + \dot{y}^2 + \left| \frac{\dot{\theta}}{R_{\min}} \right| = 0 \qquad \textit{(4-20)}$$

La simultaneidad de ambas restricciones implica la limitación del espacio de velocidades a un área triangular contenida en el plano definido por los vectores columna de la matriz de (4-15) y limitada por el cono definido en (4-20).

4.4 Maniobras Restringidas

4.4.1. Introducción

Las maniobras restringidas surgen como un método de planificación en escenarios sin obstáculos basado en la sucesión de trayectorias con ciertas

propiedades. Esta idea se inicia con las aportaciones publicadas en Dubins (1957), Reeds y Shepp (1990), Latombe (1991), Laumond y otros (1994), y Fortune y Wilfong (1988) entre otras. Sin embargo los planificadores propuestos se alejaban de las trayectorias más utilizadas entre los conductores humanos. Éstas últimas suelen ser comparativamente con las anteriores más simples, con menos cambios de sentido, y con menor superficie necesaria para ejecutar la trayectoria.

De este modo surgen las maniobras restringidas (Gómez Bravo y otros, 2001b; Gómez-Bravo, 2001a; Cuesta, 2004), que consisten en trayectorias predefinidas capaces de variar uno y sólo uno de los parámetros de configuración. Estas maniobras no son únicas, como se verá a continuación, existiendo varias para un mismo cambio de parámetro. Además, cada tipo de robot, presentará un conjunto de maniobras diferente según sea su tipo (Ackerman, diferencial, etc.).

Para poder ubicar un robot en una configuración cualquiera, en caso de que no existan obstáculos ni límites de escenario, partiendo de otra dada, son necesarias al menos, tantas maniobras como variables de configuración existan. En efecto, para cada variable de configuración se precisa de una maniobra restringida que pueda alterar su valor. La secuencia dichas maniobras calculadas de modo que eliminen la diferencia entre la configuración de partida y la final parámetro a parámetro proporcionará una trayectoria hasta alcanzar la meta.

A continuación se expondrá un tipo de notación matemática (Laumond, 1994), necesaria para expresar los distintos tipos de maniobras restringidas, así como ciertas bases teóricas para la definición de las maniobras restringidas.

Dado un espacio de configuración de dimensión *n* (*n* variables de configuración), con *r* restricciones no holónomas, si éste es controlable, es posible encontrar en el espacio de configuración una serie de *m=n-r* vectores campo, V_i, linealmente independientes, a partir de los cuales se podrá expresar el vector de velocidad del sistema V_{sys} (Laumond y otros, 1994). Es decir:

$$\exists A = \{V_1, V_2, ..., V_m\} \left| V_{sys} = \sum_{i=1}^{m} V_i \right. \qquad (4\text{-}21)$$

Además, por el Teorema de Controlabilidad de los sistemas no holónomos (Latombe,1991), se sabe que si el sistema es controlable, cualquier par de configuraciones pueden conectarse mediante un camino admisible, generado mediante

una secuencia de vectores de *A*. Estos vectores están directamente relacionados con las maniobras restringidas como se verá en los apartados siguientes.

Sea *L(t)* una trayectoria, parametrizada respecto a *t* ($t \in \mathcal{R}$), que describe el recorrido del sistema en el espacio de configuración; se dice que *L(t)* sigue el vector V_i desde *a* hasta *b* si:

$$\left.\frac{d}{dt}L(t)\right|_{t\in[a,b]} = V_i|_{L(t)} \qquad (4\text{-}22)$$

Donde el vector de campo V_i está definido en los puntos de *L(t)* para $t \in [a,b]$. Otra forma de expresar (4-22) es la siguiente (Laumond y otros, 1994):

$$L(t) = L(a) \cdot e^{tV_i} \qquad (4\text{-}23)$$

Esta expresión exponencial establece una operación definida en el espacio de configuración con el siguiente significado: "desde el punto *L(a)* se alcanza el punto *L(t)* si se sigue el vector V_i durante un valor del parámetro igual a *t* unidades".

Por tanto, si se inicia una trayectoria en un punto p_0, siguiendo el vector V_1 durante un valor t_1, se llegará a un punto p_1; si a continuación se sigue el vector V_2 durante un t_2 dado, se alcanzará un nuevo punto p_2. Este último se podrá expresar:

$$p_2(t_1,t_2,p_0) = p_0 \cdot e^{tV_1} \cdot e^{tV_2} \qquad (4\text{-}24)$$

Este formalismo ha sido utilizado para la generación de trayectorias en robot móviles. Así, en Laumond y otros (1994), se utiliza este procedimiento en robots con el modelo cinemático de un automóvil, obteniendo una expresión genérica que permite enlazar cualquier configuración con el punto origen de coordenadas y viceversa.

Se define el concepto de *camino genérico* como todo aquel movimiento generado a partir de una configuración inicial específica, siguiendo una determinada secuencia de vectores campo. Así, por ejemplo, un camino Genérico podría expresarse de la siguiente manera:

$$\Phi(t_1,t_2,...,t_n,p_0) = p_0 \cdot e^{tV_1} \cdot e^{tV_2} \cdots e^{tV_n} \qquad (4\text{-}25)$$

Adicionalmente, si dos caminos genéricos Φ_1 y Φ_2 están definidos en la forma:

$$\Phi_1(t,p) = p \cdot e^{tV_1} \qquad y \qquad \Phi_2(t,p) = p \cdot e^{tV_2} \tag{4-26}$$

La composición de Caminos Genéricos se define como:

$$\Gamma(t_1,t_2,p) = \Phi_1(t_1,p) \circ \Phi_2(t_2,p_2) \equiv p \cdot e^{tV_1} \cdot e^{tV_2} \text{ donde } p_2 = \Phi_1(t_1,p) \tag{4-27}$$

Se denomina *camino elemental* a un camino genérico definido por un único vector. Las maniobras restringidas son caminos genéricos capaces de alterar una, y sólo una, de las variables de configuración. En general se hallan como composición de los caminos genéricos definidos a partir de vectores campo del conjunto *A* (4-21). Es decir, en primer lugar se determina un conjunto de vectores campo linealmente independientes. Por cada uno de ellos existirá un camino elemental, tales como los mostrados en (4-26). La composición de estos caminos elementales permitirá la elaboración de un conjunto de maniobras restringidas. Éstas podrán utilizarse para conectar dos puntos cualesquiera del espacio de configuraciones.

4.4.2 Maniobras restringidas del robot diferencial

La configuración de un robot diferencial se descompone en tres variables: posición en el plano *(x,y)* y orientación (*θ*). La restricción no holónoma impuesta (4-12), se traduce en una reducción de la dimensión del campo de velocidades del sistema, lo cual se recoge en (4-11) (obsérvese que el rango de la matriz es dos). Por tanto, en lo que atañe a la expresión (4-21) los valores para este tipo de robots son *n=3*, pues se dispone de tres variables de configuración, *r=1* por la restricción (4-12) y *m=n-r=2*. Es decir, el conjunto de vectores campo linealmente independientes tiene tamaño dos. Podrían elegirse directamente los vectores columna de la matriz presente en (4-11), es decir $\boldsymbol{v_1}$ y $\boldsymbol{v_2}$ (4-13 y 4-14), sin embargo, debido a la mayor facilidad en el cálculo de trayectorias, se tomarán los generados por los siguientes vectores de control:

$$U_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad y \quad U_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{4-28}$$

Su producto por la matriz de (4-11) resulta:

$$V_1 = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{r\cos\theta}{2} \\ \frac{rsen\,\theta}{2} & \frac{rsen\,\theta}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ rsen\,\theta \\ 0 \end{bmatrix} \qquad (4\text{-}29)$$

$$V_2 = \begin{bmatrix} \dot{x}_2 \\ \dot{y}_2 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{r\cos\theta}{2} \\ \frac{rsen\,\theta}{2} & \frac{rsen\,\theta}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ r/l \end{bmatrix} \qquad (4\text{-}30)$$

El vector V_1 es nulo en su última componente, que corresponde a la variación de la orientación θ con respecto al tiempo, por tanto el vector campo V_1 no altera dicha orientación. Las otras dos componentes corresponden a la velocidad del centro del robot, claramente alineada con el eje principal del mismo. Consecuentemente, el camino elemental Φ_x basado en el vector campo V_1 corresponde a una traslación rectilínea realizada en la dirección en la que está orientado el robot.

$$\Phi_x(t,p) = p \cdot e^{tV_1} \qquad (4\text{-}31)$$

El vector V_2 por el contrario presenta una velocidad nula en el centro, con lo que el robot no se desplaza. En cambio se varía la orientación del mismo. Es decir, el camino elemental Φ_θ basado en el vector campo V_2 consiste en una rotación del robot sobre su centro de referencia sin desplazarse en ningún sentido.

$$\Phi_\theta(t,p) = p \cdot e^{tV_2} \qquad (4\text{-}32)$$

Estos dos caminos elementales, representados en la figura 4.4, definen las maniobras restringidas asociadas al robot diferencial, ya que su combinación permite conectar cualquier par de puntos del espacio de configuración (fig. 4.5).

En efecto, para conectar dos puntos de C basta trazar la recta que une dichos puntos, orientar el vehículo paralelo a la misma usando Φ_θ, recorrerla con Φ_x, y realizar una última reorientación con Φ_θ. Las dos primeras maniobras permiten ubicar el robot en

cualquier punto del plano (cambio de las variables *x* e *y* sin alterar *θ*), la última consigue cualquier orientación (cambio de la variable *θ* sin alterar *x* e *y*).

Figura 4.4: Maniobras restringidas en el robot diferencial

Cualquier desplazamiento del vehículo estará compuesto por el camino genérico:

$$\xi(t_1,t_2,t_3) = \Phi_\theta(t_1) \circ \Phi_x(t_2) \circ \Phi_\theta(t_3) \qquad (4\text{-}33)$$

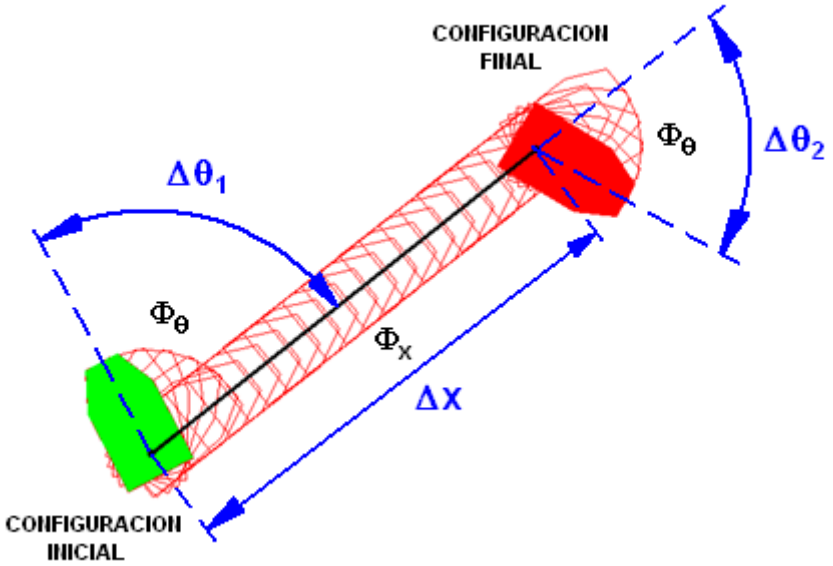


Figura 4.5: Camino obtenido usando maniobras restringidas en el robot diferencial

$$t_1 = \Delta\theta_1 \;\; ; \;\; t_2 = \Delta x \;\; ; \;\; t_3 = \Delta\theta_2 \qquad (4\text{-}34)$$

Es decir, de un giro para orientarse hacia la posición de destino, un desplazamiento hasta el punto deseado y una nueva reorientación para alcanzar el ángulo final solicitado. El parámetro *t* corresponderá con las longitudes recorridas en el

espacio de configuración, es decir, incrementos angulares para los giros y distancia euclídea en los desplazamientos sobre el plano *xy*.

También habría sido posible generar un conjunto de maniobras alternativo que, como se comenta anteriormente, alterase los parámetros *x* e *y* de forma independiente. Sin embargo, el desarrollo de estas maniobras se describe en la siguiente sección para el caso de vehículos con cinemática Ackerman. Todo lo desarrollado en ellos es aplicable a los vehículos de conducción diferencial.

4.4.3 Maniobras restringidas del vehículo tipo Ackerman

Como se describió en el apartado 4.3.2, un vehículo tipo Ackerman presenta una restricción adicional a la impuesta en los vehículos diferenciales que consiste en que su radio de curvatura está acotado. No obstante se puede emplear la misma metodología descrita con algunas consideraciones añadidas. Para estudiar las distintas maniobras restringidas se puede asumir sin pérdida de generalidad la ubicación inicial del vehículo en el origen del eje de coordenadas con orientación paralela al eje de abscisas. Esto implica que el sistema de referencia local del vehículo coincide inicialmente con el fijo y que la configuración inicial es nula en todas sus componentes. A la hora de extrapolar los resultados obtenidos por una maniobra restringida ejecutada desde esta posición, bastará con realizar un cambio de coordenadas a cada una de las posiciones de la trayectoria obtenida.

Procediendo del mismo modo que en el caso del robot diferencial, se parte de dos vectores de control. Para obtener las columnas de la matriz de (4-13) serían:

$$U_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad y \quad U_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4-35}$$

Aplicando el modelo cinemático (4-13) al primer vector se obtiene:

$$V_1 = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ sen\,\theta & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ sen\,\theta \\ 0 \end{bmatrix} \tag{4-36}$$

De nuevo corresponde a un movimiento rectilíneo del robot manteniendo la orientación constante. De hecho, puede observarse que la primera componente del vector $\boldsymbol{U_1}$ corresponde con la velocidad del vehículo en la ecuación (4-13). La segunda componente corresponde con $v\rho$, así pues la curvatura ha de ser nula.

En el vector $\boldsymbol{U_2}$ se da una contradicción. La primera componente es nula, con lo que la velocidad ha de serlo, y el producto $v\rho$ debería serlo también; a menos que ρ fuera infinita. Sin embargo este caso está excluido por la restricción (4-14). Recuérdese que esta restricción reducía el espacio de velocidades a un área triangular. Por este motivo en vez del vector $\boldsymbol{U_2}$ se escogen los vectores $\boldsymbol{U_3}$ y $\boldsymbol{U_4}$:

$$U_3 = \begin{bmatrix} 1 \\ c_1 \end{bmatrix} \quad y \quad U_4 = \begin{bmatrix} 1 \\ c_2 \end{bmatrix} \tag{4-37}$$

Donde los coeficientes c_1 y c_2, siendo $c_1>0$ y $c_2<0$, se han elegido de modo que cumplan la condición $c_1 = -c_2 = \rho_{max}$, valor especificado en la restricción (4-16). El resultado de aplicar (4-15) a $\boldsymbol{U_3}$ y $\boldsymbol{U_4}$ son los vectores:

$$W_3 = \begin{bmatrix} \cos\theta & sen\theta & c_1 \end{bmatrix}^t \quad y \quad W_4 = \begin{bmatrix} \cos\theta & sen\theta & c_2 \end{bmatrix}^t \tag{4-38}$$

Los cuales corresponden a una trayectoria curva utilizando el radio positivo mínimo posible ($\boldsymbol{W_3}$) y a otra curva con radio negativo de mínimo módulo posible ($\boldsymbol{W_4}$). En el área triangular del espacio de las velocidades definido por (4-16) y (4-17), estos vectores corresponden justo con los límites de dicha área. A partir de estos tres vectores se definen los caminos elementales:

$$\Phi_x(t,p) = p \cdot e^{tV_1} \quad ; \quad \Phi_{r+}(t,p) = p \cdot e^{tV_3} \quad ; \quad \Phi_{r-}(t,p) = p \cdot e^{tV_4} \tag{4-39}$$

A partir de estos caminos elementales se pueden obtener por composición otros capaces de alterar una, y sólo una, de las variables de configuración. A estos últimos se les denominará maniobras restringidas y se les denotará con la letra Φ^p, donde p será x, y o θ dependiendo cuál sea la variable afectada. La primera de estas maniobras se utiliza para conseguir un Δx, sin variar y o θ. Así, la primera queda definida en la forma:

$$\Phi^x(t_1) = \Phi_x(t_1, p) \quad \text{con} \quad t_1 = \Delta x \tag{4-40}$$

En la figura 4.4 derecha se puede observar esta maniobra, ya que es idéntica a la definida en robots diferenciales.

La segunda maniobra restringida que se expone es la de cambio de orientación, identificada como Φ^{θ}, que se define:

$$\Phi^{\theta}(p,t_1,t_2,t_3) = \Phi_x(t_1,p) \circ \Phi_{r-}(t_2,p) \circ \Phi_x(t_3,p) \qquad (4\text{-}41)$$

$$\Phi^{\theta}(p,t_1,t_2,t_3) = p \cdot e^{t_1 V_1} \cdot e^{t_2 V_4} \cdot e^{t_3 V_1} \qquad (4\text{-}42)$$

En la figura 4.6 puede observarse esta trayectoria y además el polígono que la inscribe.

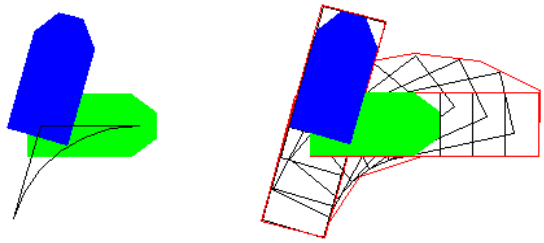


Figura 4.6: Maniobra restringida de reorientación Φ^{θ}.

Para realizarla el vehículo se desplaza en el sentido de su coordenada local *x* hacia adelante, continúa con un arco hacia atrás hasta situarse en línea con la posición cartesiana inicial, y por último realiza un desplazamiento recto que lo sitúa en el punto del que partió, pero con una orientación diferente.

A raíz de los resultados obtenidos en Gómez-Bravo (2001a) los valores de t_1, t_2 y t_3 para un incremento de orientación $\Delta\theta$ dado serían:

$$t_1 = -(r_1 \cdot sen\Delta\theta + t_3 \cdot \cos\Delta\theta) \qquad (4\text{-}43)$$

$$t_2 = r_1 \cdot \Delta\theta \qquad (4\text{-}44)$$

$$t_3 = r_1 \frac{(\cos\Delta\theta - 1)}{\sin\Delta\theta} \qquad (4\text{-}45)$$

Donde r_1 es el radio de de la máxima curvatura posible $c_1 = \rho_{max}$. Se puede observar que no es la única maniobra capaz de provocar el mismo resultado. Basta con considerar la maniobra simétrica en cuanto al sentido del movimiento para obtener idéntico objetivo. En efecto, si el vehículo inicia el movimiento desplazándose hacia atrás en un segmento recto de igual longitud, continúa con un arco de curvatura inversa y hacia delante y termina en la posición inicial con una maniobra recta marcha atrás, se obtiene otra maniobra factible (Gómez-Bravo, 2001a).

Para el cambio de ordenada se define la maniobra restringida Φ^y:

$$\Phi^y(p,t_1,t_2,t_3) = \Phi_{r+}(t_1,p) \circ \Phi_{r-}(t_2,p) \circ \Phi_x(t_3,p) \qquad (4\text{-}46)$$

$$\Phi^y(p,t_1,t_2,t_3) = p \cdot e^{t_1 V_3} \cdot e^{t_2 V_4} \cdot e^{t_3 V_1} \qquad (4\text{-}47)$$

Los valores t_1, t_2 y t_3 correspondientes a un desplazamiento Δy (Gómez-Bravo, 2001) son:

$$\theta_2 = \arccos\left(1 - \frac{\Delta y}{r_1 - r_2}\right) \qquad (4\text{-}48)$$

$$r_1 = \frac{1}{c_1} \;\; ; \;\; r_2 = \frac{1}{c_2} \qquad (4\text{-}49)$$

$$t_1 = r_1 \cdot \theta_2 \qquad (4\text{-}50)$$

$$t_2 = -\frac{t_1 \cdot r_2}{r_1} \qquad (4\text{-}51)$$

$$t_3 = (r_2 - r_1) sen\theta_2 \qquad (4\text{-}52)$$

Esta maniobra comienza con un arco de radio positivo hacia delante, seguido de uno de radio negativo, de tal forma que la orientación final coincida con la de partida. Posteriormente se realiza un desplazamiento recto para rectificar el cambio en la variable *x*. En la figura 4.7 puede observarse esta trayectoria y además el polígono que la inscribe.

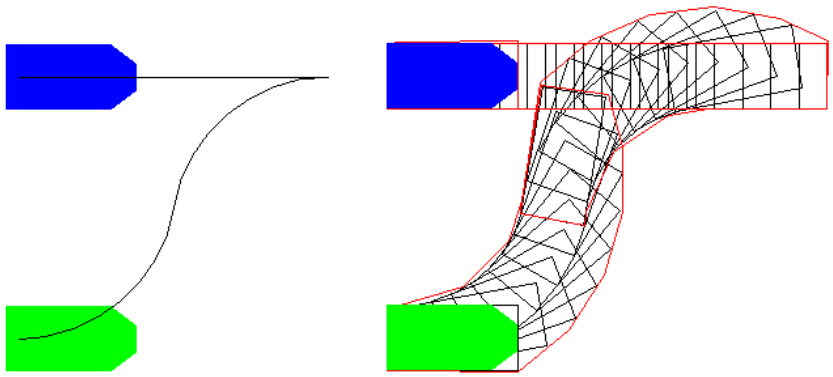


Figura 4.7: Maniobra restringida de cambio de ordenada Φ^y.

Si la diferencia de ordenadas es inferior a cuatro veces el radio mínimo ($\Delta y < 4\rho_{min}$), como es el caso de la figura 4.7, se requerirá un desplazamiento recto para anular la variación de abscisas experimentada (último camino de la composición en (4-46)). Si por el contrario se excede o se iguala dicho valor ($\Delta y \geq 4\rho_{min}$), la maniobra podría realizarse con dos curvas semicirculares en forma de S cuyo radio sería la cuarta parte del incremento de ordenada a salvar (Δy), siendo innecesaria una maniobra recta al final.

Con ésta se completa la terna de maniobras restringidas necesarias para hallar cualquier solución al problema de conectar dos configuraciones cualesquiera en *C.* En el siguiente apartado se detalla la forma de utilizar estas maniobras restringidas.

4.4.4 Maniobras de conexión del vehículo tipo Ackerman

En un escenario sin obstáculos ni límites se plantea el problema de conectar dos configuraciones cualesquiera con una trayectoria que cumpla las restricciones no holónomas del sistema (4-16) y (4-17). La diferencia entre estas dos configuraciones será ($\Delta x, \Delta y, \Delta\theta$). Para salvar esta diferencia se han definido las maniobras restringidas del apartado anterior, que permiten variar una a una las variables de configuración del sistema. A las composiciones de maniobras restringidas capaces de unir dos puntos cualesquiera de *C* se las denomina *maniobras de conexión* (Gómez-Bravo, 2009). Por ejemplo, el camino genérico representado en la figura 4.5 muestra una maniobra de conexión del vehículo diferencial.

En vehículos tipo Ackerman, la maniobra de conexión más evidente consiste en elegir una secuencia cualquiera de las maniobras restringidas e ir componiéndolas. Por ejemplo la secuencia $\{\Phi^y, \Phi^x, \Phi^\theta\}$ da lugar a la que denominaremos maniobra de conexión Γ, y consiste en variar en primer lugar la variable de configuración *y* con una maniobra restringida Φ^y, después la variable *x* con Φ^x, y por último efectuar el cambio de orientación que sea necesario con Φ^θ (ver figura 4.8). En la siguiente fórmula se han omitido los parámetros t_i por simplicidad en la notación:

$$\Gamma = \Phi^y \circ \Phi^x \circ \Phi^\theta \qquad (4\text{-}53)$$

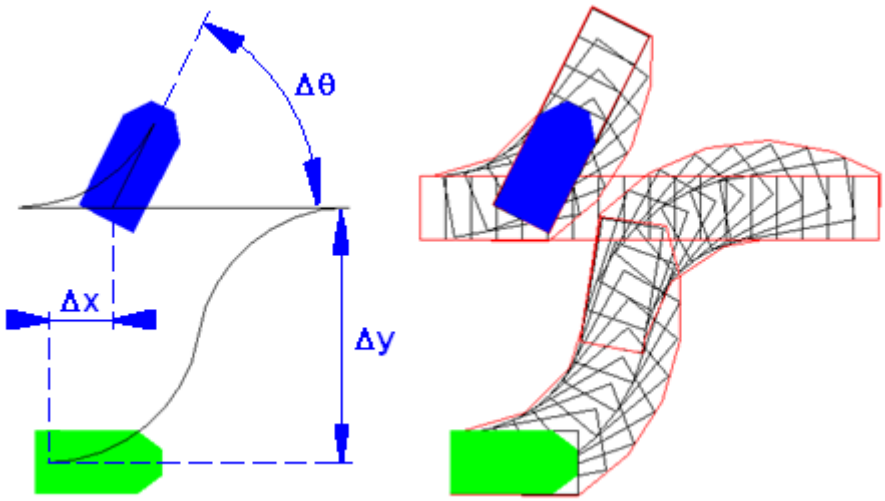


Figura 4.8: Maniobra de conexión Γ.

Existen varias puntualizaciones con respecto a estas maniobras de conexión que es conveniente resaltar. En primer lugar, al realizar la composición suelen aparecer maniobras superfluas. En efecto, si en la ecuación (4-53) se descomponen las maniobras restringidas en sus caminos elementales, entre una y la siguiente puede aparecer el mismo camino elemental, es decir, avances y retrocesos sobre la misma línea:

$$\Gamma = p \cdot e^{t_1 V_3} \cdot e^{t_2 V_4} \cdot e^{t_3 V_1} \cdot e^{t_4 V_1} \cdot e^{t_5 V_1} \cdot e^{t_6 V_4} \cdot e^{t_7 V_1} \qquad (4\text{-}54)$$

Es necesario detectar estos desplazamientos innecesarios antes de procesar estas maniobras, ya que normalmente implican un recorrido mayor, con el consecuente

aumento de riesgo de rechazo por colisionar con obstáculos. De este modo, (4-54) puede reducirse a:

$$\Gamma = p \cdot e^{t_1 V_3} \cdot e^{t_2 V_4} \cdot e^{(t_3+t_4+t_5)V_1} \cdot e^{t_6 V_4} \cdot e^{t_7 V_1} \qquad (4\text{-}55)$$

En la figura 4.9 se muestra la maniobra de conexión Γ antes de la simplificación (trayectoria superior) y después de la misma (trayectoria inferior). Los números representados corresponden a la secuencia de caminos elementales efectuada, y a la vez, a los subíndices de los parámetros t_i en las fórmulas (4-54) y (4-55). En color azul se indican los caminos elementales asociados a la maniobra restringida Φ^y. En color verde el asociado a la maniobra restringida Φ^x. Y por último en color rojo los asociados a la maniobra restringida Φ^θ. Como puede observarse la simplificación elimina en este caso largos recorridos innecesarios.

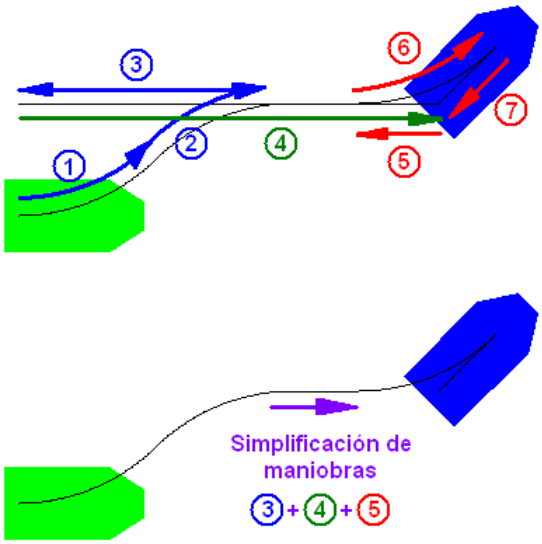


Figura 4.9: Simplificación de la maniobra de conexión Γ.

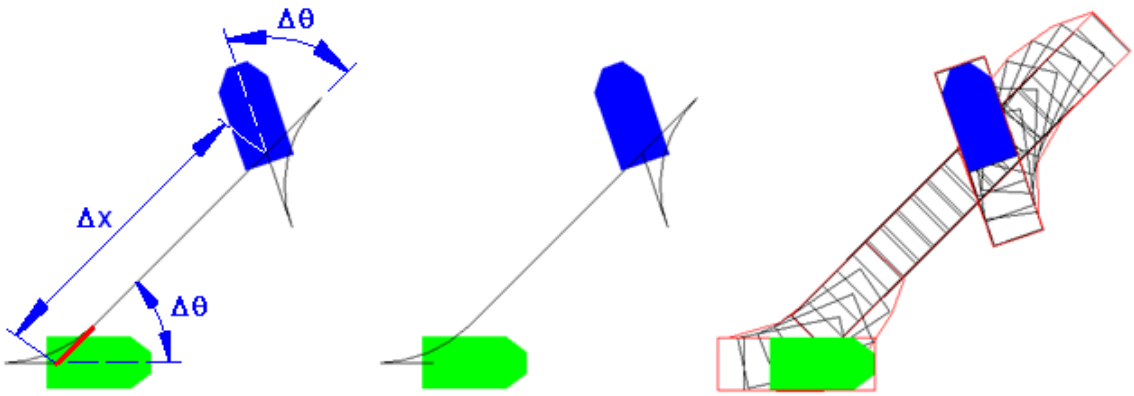
En segundo lugar ha de destacarse el mayor número de maniobras disponibles. Esta diversidad es importante dado que al existir más restricciones, las trayectorias de las maniobras restringidas resultan en ocasiones más largas que sin ellas, aumentando la posibilidad de colisión con obstáculos. Por tanto resulta muy útil disponer de maniobras alternativas. El amplio conjunto de maniobras de conexión obedece por un

lado a la variedad de maniobras restringidas que existen para un mismo cambio de coordenada, y por otro, a los diferentes órdenes en que podemos secuenciarlas.

Además, hay otras formas de conectar dos configuraciones cualesquiera del vehículo a través de maniobras restringidas que la simple variación consecutiva de cada una de sus coordenadas. Un ejemplo de ello se puede observar en la figura 4.10, donde se presenta una de estas posibilidades (maniobra de conexión Λ). En este caso se trata de la secuencia compuesta por una reorientación hacia el punto destino, una traslación recta que salva de una vez tanto el incremento de la variable *x* como de la variable *y*, y una reorientación final que ubique el vehículo en su configuración definitiva.

$$\Lambda = \Phi^{\theta} \circ \Phi^{x} \circ \Phi^{\theta} \qquad (4\text{-}56)$$

Obsérvese que esta maniobra de conexión es la más parecida a la utilizada en el vehículo diferencial (4-33), diferenciándose con éste únicamente en la complejidad de la maniobra de reorientación.



a) Original. *b) Simplificada.* *c) Envolvente poligonal.*

Figura 4.10: Maniobra de conexión Λ.

Esta maniobra de conexión también puede simplificarse para evitar desplazamientos superfluos. De nuevo se concatenan tres desplazamientos rectos en el ecuador de la maniobra que equivaldrán a uno solo, cuyo parámetro *t* será la suma de los anteriores:

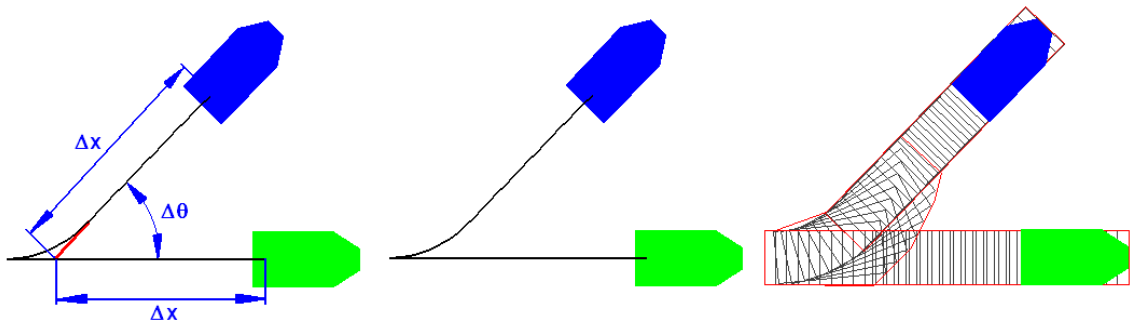
$$\Lambda = p \cdot e^{t_1 V_1} \cdot e^{t_2 V_4} \cdot e^{t_3 V_1} \cdot e^{t_4 V_1} \cdot e^{t_5 V_1} \cdot e^{t_6 V_4} \cdot e^{t_7 V_1} \qquad (4\text{-}57)$$

$$\Lambda = p \cdot e^{t_1 V_1} \cdot e^{t_2 V_4} \cdot e^{(t_3+t_4+t_5)V_1} \cdot e^{t_6 V_4} \cdot e^{t_7 V_1} \qquad (4\text{-}58)$$

En la figura 4.10a se muestra la maniobra de conexión Λ correspondiente a la fórmula (4-57), es decir, antes de realizar ninguna simplificación. Puede observarse en color rojo un fragmento de trayectoria que se recorre dos veces sin necesidad. La figura 4.10b corresponde a la maniobra de conexión simplificada (4-58), donde desaparece el fragmento anterior. Y por último la figura 4.10c muestra los polígonos que acotan el espacio recorrido por el vehículo.

Sin embargo, Λ no es la maniobra de conexión más intuitiva. Ésta quizás se aproxime más a la mostrada en la figura 4.11 donde se hace uso de configuraciones intermedias de fácil acceso desde sendas posiciones inicial y final. Es decir, hallamos la intersección de los ejes longitudinales del vehículo en sus posiciones inicial y final descubriendo ahí configuraciones de acceso sencillo por parte de ambas. La maniobra de conexión Ξ utiliza este recurso:

$$\Xi = \Phi^x \circ \Phi^\theta \circ \Phi^x \qquad (4\text{-}59)$$



a) Original. *b) Simplificada.* *c) Envolvente poligonal.*

Figura 4.11: Maniobra de conexión Ξ (I).

Como puede observarse en la figura 4.11, también se han eliminado trayectorias redundantes. En la figura 4.11a se ha marcado en color rojo el desplazamiento supérfluo. La figura 4.11b muestra la trayectoria simplificada. Y por último la figura 4.11c detalla la huella de la maniobra y el polígono que la circunscribe. Al contrario que en las maniobras de conexión precedentes, la simplificación de maniobras se realiza al principio y al final, ya que sólo en esas localizaciones coinciden los tipos de maniobra:

$$\Xi = p \cdot e^{t_1 V_1} \cdot e^{t_2 V_1} \cdot e^{t_3 V_4} \cdot e^{t_4 V_1} \cdot e^{t_5 V_1} \qquad (4\text{-}60)$$

$$\Xi = p \cdot e^{(t_1+t_2) V_1} \cdot e^{t_3 V_4} \cdot e^{(t_4+t_5) V_1} \qquad (4\text{-}61)$$

Puede observarse que la simplificación permite generar trayectorias sencillas cuando es posible, como el caso de la figura 4.12 donde la maniobra de reorientación intermedia carece de desplazamientos rectos debido precisamente a esta simplificación. Esto demuestra la capacidad de las maniobras de conexión de ofrecer trayectorias continuas cuando es posible y salpicadas de inversores cuando no.

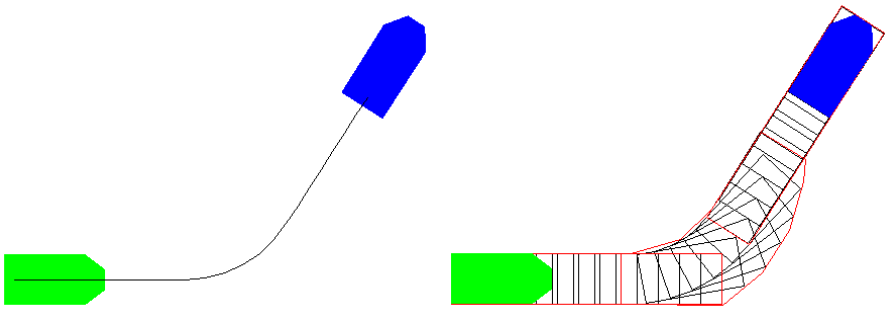


Figura 4.12: Maniobra de conexión Ξ (II).

4.5 Envolvente poligonal

En la definición de las maniobras de conexión no se ha tenido en cuenta la existencia de obstáculos. Como se verá en el apartado 4.6, para utilizar las maniobras de conexión en un escenario con obstáculos será necesario comprobar si éstas provocan la colisión del robot. La función de detección de colisión debe ser rápida, dado que ha de ejecutarse muchas veces. Por lo tanto hay que identificar el espacio por el que discurre el vehículo al ejecutar la maniobra, y hacerlo de modo que sea fácil contrastarlo con el espacio libre del escenario.

Dentro de este contexto (robot tipo Ackerman y escenario modelado en dos dimensiones), existen diferentes formas de identificar este espacio. La primera consiste en descomponer el escenario en celdas rectangulares de igual tamaño (por ejemplo los

pixels de un mapa de bits), y marcarlas como ocupadas o libres, dependiendo de si dentro de ellas hay presencia de obstáculos. Esto dará lugar a una matriz booleana cuyos elementos describirán el estado de cada celda. Por otro lado puede crearse una matriz similar para cada maniobra, de modo que sus elementos reflejen si una determinada celda será o no ocupada por el robot al desplazarse. Realizando un examen celda por celda entre ambas matrices se averiguará si el robot colisiona o no con algún obstáculo.

La segunda forma, parte de una descripción poligonal de los obstáculos. Con respecto al robot, ha de acotarse con un polígono el área ocupada por el mismo al ejecutar la maniobra. A continuación se comprueba la posibilidad de solapamiento entre el área delimitada por dicho polígono, con cada una de las delimitadas por los polígonos que modelan los obstáculos del escenario. En cada una de estas comprobaciones se considera la intersección de los segmentos (cada uno de los de un polígono con cada uno de los del otro), y además la posibilidad de inclusión (se contrasta si un vértice de uno de los polígonos está en el interior del otro). Este método presenta un alto número de iteraciones dependiente de la cantidad de vértices a considerar. Si el número de vértices de los polígonos de los obstáculos es pequeño, este método será más rápido que el anterior, y viceversa. Por el contrario tiene la ventaja de ser mucho más preciso, dado que no descompone el espacio en celdas.

En cualquier caso la velocidad comparativa entre los métodos anteriores dependerá del número de obstáculos presentes en el escenario, su morfología (número de vértices de cada uno), y el grado de discretización mínimo requerido.

En la práctica, las inexactitudes en el control, la localización o el modelado del robot, revocan la utilidad de una descomposición en celdas excesivamente pequeñas. Por este motivo es frecuente utilizar los propios pixels de una imagen representativa del escenario como celdas, siendo habitualmente suficiente para los propósitos habituales en robótica. Esto incita a utilizar el primer método, a pesar de que el área barrida por el robot al desplazarse puede contener miles de puntos, es decir, miles de iteraciones en la función de comprobación de colisiones.

Por otra parte, está claro que en caso de pocos obstáculos, o mejor dicho, pocos vértices que los definan, el procedimiento de los polígonos resulta ventajoso. Intuitivamente, parece más acertado recurrir a un perímetro poligonal (menos necesidad de espacio en memoria, idea de que la frontera define en área contenida). Además, en la

mayoría de los casos los obstáculos son siempre mayores que el robot, resultando innecesaria la verificación de inclusiones.

Todas estas razones han llevado a tomar la decisión de utilizar un procedimiento mixto entre ambas soluciones para los experimentos y simulaciones de esta tesis. Este procedimiento consiste en utilizar el polígono que inscribe las maniobras pero contrastado con el mapa de bits en lugar de con los polígonos de los obstáculos. Es decir, se comprueban únicamente los píxels asociados al polígono que inscribe la maniobra. Si ninguno de ellos coincide con un pixel marcado en el mapa de bits como ocupado por algún obstáculo, entonces no hay colisión. Si uno sólo coincide, entonces la hay. Con este procedimiento se logran velocidades de cómputo satisfactoriamente bajas en la mayoría de los escenarios.

Tanto en este procedimiento mixto como en el de intersecciones de polígonos (la segunda forma) es necesario definir uno que inscriba el área utilizada para efectuar las maniobras. Este polígono ha de tener el mínimo número de vértices posible como se ha justificado anteriormente, de ahí la relativa imprecisión de los mostrados en las figuras del apartado anterior. Obviamente es posible definir polígonos más ceñidos a la maniobra utilizando un conjunto de vértices mayor, pero a costa de elevar el tiempo de cómputo tanto en su cálculo como en la función de detección de colisión. En concreto, para inscribir los límites curvos de las maniobras, en los experimentos se ha trabajado con líneas poligonales cuyos segmentos coinciden en tamaño con el lado menor de la planta del vehículo.

Llegados a este punto, se procede a describir el método utilizado para generar los polígonos que inscriben las maniobras de los robots tipo Ackerman. Para comenzar se parte de la planta del vehículo. Ésta es inscrita en un rectángulo mínimo. Dicho rectángulo será el utilizado como área de referencia para generar la huella del robot al desplazarse. Cualquier tipo de maniobra que realice el robot, puede descomponerse en una secuencia de fragmentos rectilíneos o curvos. Estos dos tipos de maniobras básicas, el desplazamiento rectilíneo y el movimiento con radio de curvatura constante, tienen asociado un polígono característico.

El movimiento rectilíneo es sencillo de inscribir en un rectángulo, dado que el mismo robot suele describirse con una planta rectangular que sólo puede desplazarse así siguiendo uno de sus ejes de simetría. El polígono solución (ver fig. 4.13) resulta ser

un rectángulo con la misma anchura del robot y con una longitud igual a la suma de la del robot más la del desplazamiento efectuado.

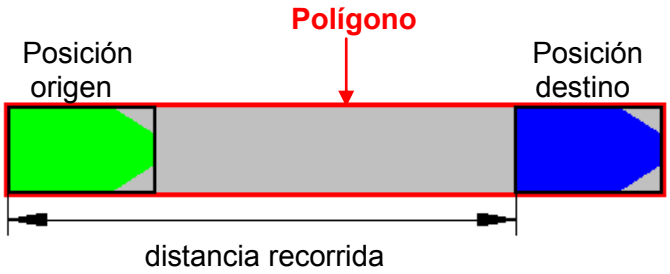


Figura 4.13: Polígono envolvente de una maniobra recta

El desplazamiento con radio de curvatura constante deja una huella menos obvia de delimitar con un polígono. Sin embargo basta atender a las posiciones inicial y final de la planta del vehículo y a los arcos descritos por los vértices extremos en el sentido del radio de curvatura, es decir, el más alejado y el más cercano al centro de giro. En la figura 4.14 pueden observarse dichos vértices, donde se ha tomado un centro de giro alineado con el límite izquierdo de la planta rectangular que inscribe a la del robot.

El arco descrito por el vértice más cercano al centro de giro ha de ser aproximado mediante una línea poligonal. Se parte del dato de la longitud máxima *l* de segmento, a partir de ésta, el arco asociado θ equivalente a dicha cuerda será (ver fig. 4.15a):

$$\theta = 2 \cdot \arcsin\left(\frac{l}{2 \cdot r}\right) \qquad (4\text{-}62)$$

Donde *r* representa el radio desde el centro de giro hasta el vértice más próximo. Para construir la línea poligonal se divide el ángulo total de giro entre el valor de θ. El cociente determinará el número de puntos del arco interior que servirán de vértices a la línea poligonal. El resultado puede observarse en la figura 4.16.

El caso de la línea poligonal exterior resulta algo más complejo. Dicha línea debe quedar circunscrita a la circunferencia definida por el vértice más alejado, de radio *R,* con objeto de asegurar la inexistencia de colisión. Así se calcula un radio ampliado *R'* sobre cuya circunferencia estarán los vértices de la línea poligonal, de tal forma que sus segmentos sean tangentes a la circunferencia máxima alcanzada por el robot.

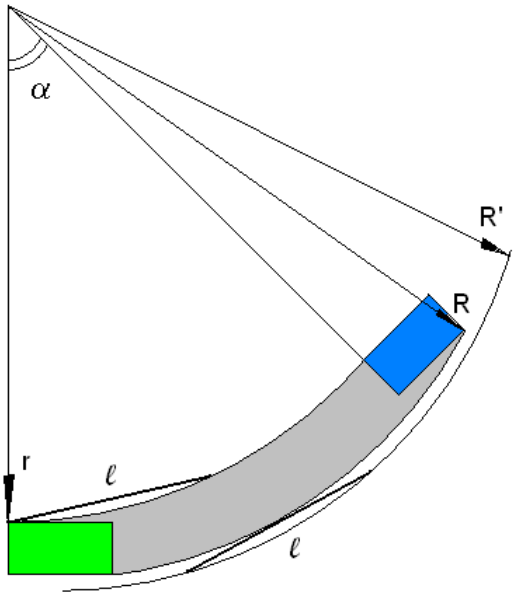
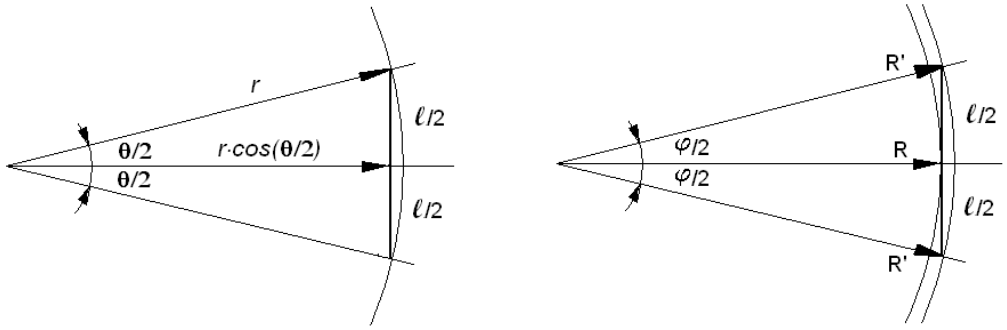


Figura 4.14: Envolvente de un camino elemental curvo



a) Segmento interior al arco de radio r *b) Segmento exterior al arco de radio R*

Figura4.15: Detalle de envolvente para arcos

Sea *c* el ancho del robot y *d* su envergadura. Observando las figuras 4.14 y 4.15b pueden deducirse las siguientes relaciones:

$$R = \sqrt{(r+c)^2 + d^2} \qquad (4\text{-}63)$$

$$R' = \sqrt{R^2 + \left(\frac{l}{2}\right)^2} \qquad (4\text{-}64)$$

$$\varphi = 2 \cdot \arcsin\left(\frac{l}{2 \cdot R'}\right) \qquad (4\text{-}65)$$

Con las dos últimas ecuaciones (4-62 y 4-63) pueden obtenerse los puntos de la línea poligonal exterior, dado que pertenecen a la circunferencia definida por R' y están separados con la magnitud φ. Al igual que en el caso anterior, se divide el giro completo α en porciones de φ radianes para marcar los puntos de la línea poligonal. En la figura 4.16 puede observarse el resultado

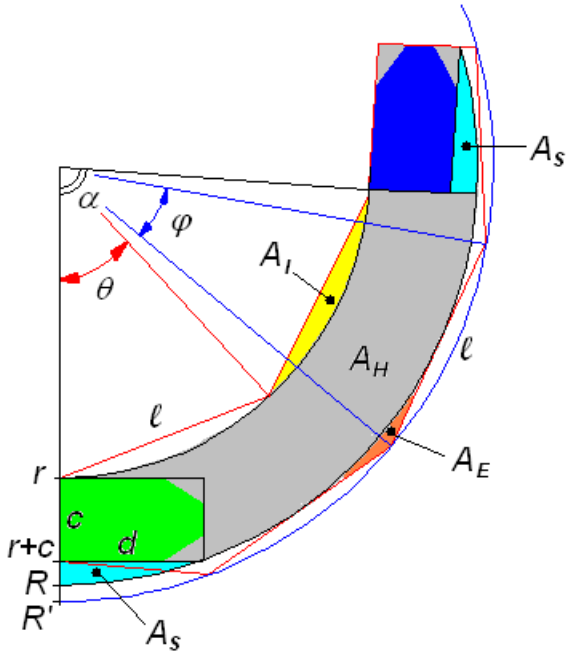


Figura 4.16: Poligono envolvente de una trayectoria curva

La elección del tamaño de *l* no resulta obvia. Por un lado si se escoge demasiado grande, el espacio añadido por el polígono a la huella del robot será excesivo, y maniobras factibles podrían ser rechazadas. Por otro, si se escoge demasiado pequeño, el número de vértices del polígono aumentará, y con éste el tiempo de cómputo en las funciones que lo utilicen. Para clarificar esta cuestión se define la función *E* como el error relativo en términos de área cometido al aproximar con un polígono la huella del robot en un desplazamiento curvo. Dicho error se origina en dos lugares: la línea poligonal interior y la exterior. El área añadida interior, A_I, se halla restando del sector circular de ángulo θ

correspondiente a la cuerda *l*, el triángulo definido por dos radios *r* y la propia *l*. Para una maniobra curva de ángulo *α* el área añadida total será:

$$A_I = \frac{\theta}{2\pi}\pi \cdot r^2 - \frac{lr\cos(\theta/2)}{2} = r^2 arcsen\left(\frac{l}{2r}\right) - \frac{lr}{2}\sqrt{1-\left(\frac{l}{2r}\right)^2} \qquad (4\text{-}66)$$

Para la línea poligonal exterior sucede al revés, es decir, del triángulo formado por dos radios *R'* y *l* hay que sustraer el sector circular de radio *R* y ángulo *φ*. El área de error en el lado exterior A_E para una trayectoria curva de ángulo total *α* es:

$$A_E = \frac{Rl}{2} - \frac{\varphi}{2\pi}\pi R^2 = \frac{Rl}{2} - R^2 arcsen\left(\frac{l}{2R'}\right) \qquad (4\text{-}67)$$

El área total correspondiente a la huella del camino elemental curvo de ángulo *α* corresponde al sector circular del mismo tamaño angular *α* más el área del robot. Esto puede observarse en la figura 4.14. La huella del robot en su posición inicial queda dentro del sector circular definido a partir de la línea vertical de la figura 4.13 (es decir, en -π/2), de radio interior r, de radio exterior R, y con un arco de alfa grados. Si a continuación se añade el área del robot en su posición final, se observa que sólo falta un fragmento de área aledaña al vértice extremo del vehículo (A_s). Ésta es justo el fragmento del sector circular por debajo de la cota *y=-(r+c)*, que por otra parte no pertenece a la huella. Por lo tanto el área total cubierta por la maniobra, A_H sería:

$$A_H = \frac{\alpha}{2\pi}(\pi \cdot R^2 - \pi \cdot r^2) + cd = \frac{\alpha}{2}(R^2 - r^2) + cd \qquad (4\text{-}68)$$

Dividiendo A_I y A_E por el área total A_H se obtiene el error relativo *E*:

$$E = \frac{r^2 - \frac{lr\sqrt{1-(l/2r)^2}}{2arcsen(l/2r)} + \frac{lR}{2arcsen(l/2R')} - R^2}{R^2 - r^2 + \frac{2cd}{\alpha}} \qquad (4\text{-}69)$$

El número de vértices del polígono, *N*, sería igual a los 4 visibles por el robot más tantos como lados tengan cada una de las líneas poligonales. Es decir:

$$N = 4 + \frac{\alpha}{\theta} + \frac{\alpha}{\varphi} = 4 + \frac{\alpha}{2arcsen(l/2r)} + \frac{\alpha}{2arcsen(l/2R')} \qquad (4\text{-}70)$$

En la fig. 4.17 se muestra una representación de la función *E* y en la figura 4.18 la función *N*, para los valores concretos de planta y radio de giro correspondientes al robot Romeo4R. En el eje *x* se muestra el valor de *l*. En el eje *y* valores de la maniobra curva completa (*α*). Los valores de *E* están en tanto por ciento. Estas gráficas muestran hasta qué punto ambas funciones son sensibles a los valores de *l* elegidos, y su dependencia con la longitud de la maniobra.

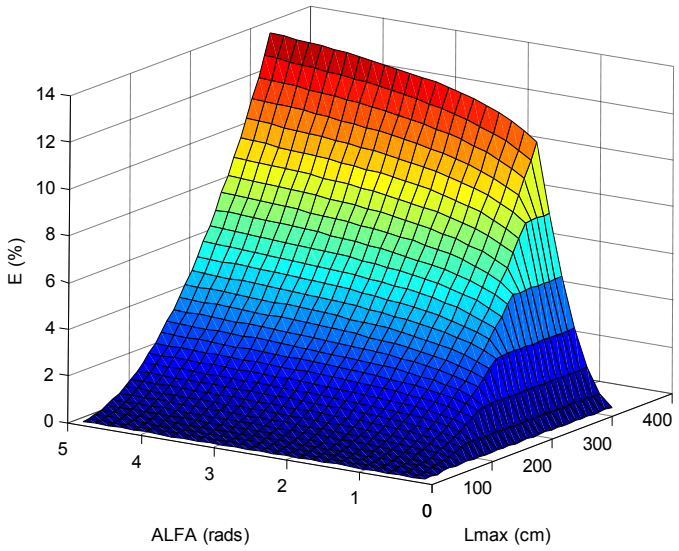


Figura 4.17: Función E (Error relativo en el área inscrita).

Resulta difícil plantear la elección del valor óptimo, pero el estudio de estas gráficas facilita la tarea. En el caso de los experimentos aquí planteados se ha optado por un valor de *l* igual a la mitad de la longitud del vehículo *d*, dado que en los experimentos realizados no se ha observado rechace de maniobras allí donde eran necesarias, y la velocidad de cómputo ha permitido la navegación en línea.

Cabría considerar la posibilidad de utilizar directamente los conjuntos de celdas (pixels) ocupados por la maniobra y compararlos con las celdas del escenario, tal y como se expresó al principio de esta sección. Para mostrar la bondad del método escogido se realizará un breve razonamiento. Si el ancho del robot es *c* y la longitud de la trayectoria es *s,* entonces el área de la huella será aproximadamente *c·s*.

En esta aproximación se está asumiendo que la longitud del vehículo *d*<<*s* y que el radio medio de curvatura *r*>>*c*, lo cual es bastante usual en problemas prácticos. En cambio, el perímetro se aproximará al valor 2(*c*+*s*).

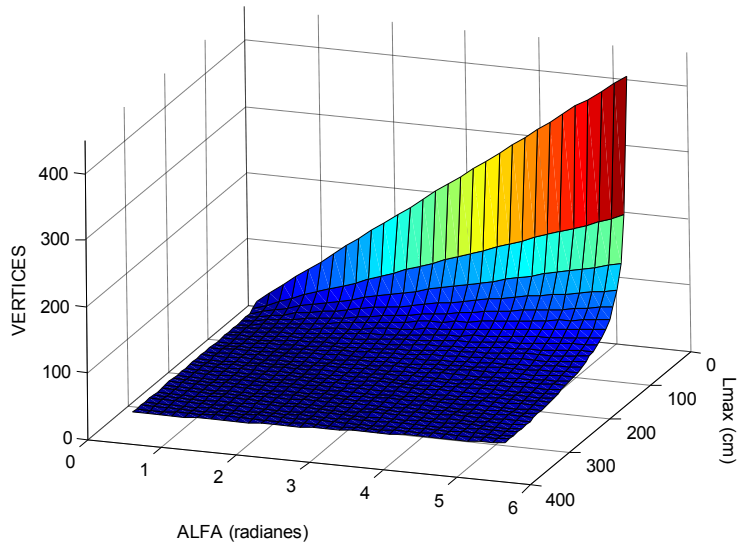


Figura 4.18: Número de vértices del polígono circunscrito.

El coste computacional de comprobación de colisión será proporcional al número de celdas, el cual a su vez será proporcional a los valores descritos. Para comparar ambos costes (el derivado de utilizar polígonos y el debido al área completa) se define el parámetro μ como la proporción entre uno (el área) y otro (el perímetro):

$$\mu = \frac{sc}{2(s+c)} = \frac{c}{2(1+c/s)} \approx \frac{c}{2} \qquad (4\text{-}71)$$

Es decir, el coste computacional debido a la consideración del área barrida por el robot es c/2 veces mayor al necesario cuando utiliza sólo su frontera. En términos prácticos ha de resaltarse que en los experimentos efectuados *c=24*, es decir, el procedimiento de los polígonos resulta unas doce veces más rápido (un orden de magnitud inferior) que la comprobación de todos los puntos del área.

4.6 Planificación con obstáculos

4.6.1 Introducción

Hasta ahora no se han tenido en cuenta ningún tipo de límites en el espacio de configuración *C*, tan sólo las restricciones no holónomas. Se ha demostrado que las maniobras de conexión permiten unir dos puntos cualesquiera de *C* con una trayectoria que cumple las restricciones no holónomas. Si existen obstáculos en *C*, es posible que estas maniobras de conexión los crucen invalidándolas.

Por otro lado se han visto multitud de métodos en el capítulo 2 que eluden los obstáculos ofreciendo una ruta entre origen y destino. Estas trayectorias suelen tener forma de líneas poligonales que violan las restricciones cinemáticas de los robots analizados. No obstante, cada uno de sus puntos son configuraciones posibles del robot que podrían servir de guía para obtener una trayectoria admisible, es decir, que cumpla las restricciones.

Para lograr una trayectoria admisible entre origen y destino que sortee los obstáculos y cumpla las restricciones no holónomas han de utilizarse ambos recursos. La idea es probar las diferentes maniobras de conexión entre los puntos origen y destino de la ruta original. Si alguna de ellas logra salvar los obstáculos se obtiene directamente una solución. Si por el contrario esto no sucede, se procede a elegir puntos intermedios de la ruta original, y se trata de unir cada par de puntos consecutivos nuevamente con maniobras de conexión. Cuanto más distantes sean estos puntos, más fácil será el rechazo de la maniobra por colisión, dado que la maniobra será más amplia. Cuanto más próximos sean estos puntos, más fácil será la conexión, dado que se aproxima más a la ruta original que pertenece a C_{free}, sin embargo aumenta el número de maniobras para recorrer una misma distancia.

Así pues ha de diseñarse un algoritmo capaz de realizar maniobras suaves, sin inversores, siempre que haya espacio para ello; y por otro lado pueda maniobrar de forma prolija cuando la abundancia de obstáculos lo requiera. En el siguiente apartado se analizará el algoritmo propuesto en esta tesis, denominado algoritmo de

postprocesado, ya que precisa un planificador previo que le proporcione una trayectoria de la que partir.

4.6.2 El algoritmo

Para describir el algoritmo cuyo esquema (ver figura 4.19) se presenta, se partirá de uno muy similar explicado en el apartado 2.7.7. En aquel caso se planteaba el problema de mejorar la trayectoria obtenida con el algoritmo RRT. Ésta consistía en una línea poligonal con abundante número de vértices. La estrategia empleada para reducirlos consistía en probar segmentos sustitutorios de forma sucesiva. En primer lugar se trataba de conectar origen y destino. Si el segmento era rechazado se dividía la línea poligonal por la mitad y se trataba de sustituir cada tramo por separado. Siguiendo la misma idea, cada segmento rechazado implicaba la subdivisión de su tramo, mientras que los aceptados pasaban a formar parte de la solución final buscada.

El algoritmo presente (fig. 4.19), es similar en cuanto a su planteamiento pero incorpora diferencias. Uno de los cambios que pueden apreciarse es el hecho de que en vez de ir dividiendo por la mitad el fragmento de trayectoria a comprobar, se opta por un recorrido más exhaustivo. Es decir, se reduce dicho fragmento progresivamente, recorriendo cada una de las posiciones en las que está discretizada la trayectoria, buscando desde un principio la maniobra más extensa posible. Ello es debido a que en el caso anterior era relativamente fácil encontrar la solución, con lo que el sistema de "bisección" resultaba más rápido sin perder eficacia. Sin embargo en el tipo Ackerman las restricciones imponen recorridos más complejos que suelen colisionar con los obstáculos. Debido a lo cual, la mencionada falta de exhaustividad del antiguo método era suficiente en muchos casos para excluir la solución.

Por otro lado, con el fin de lograr un camino lo más suave posible, interesa buscar las maniobras de mayor recorrido, que sólo pueden detectarse con este cambio. Así pues, *p* representa el vértice de partida de cada fragmento de trayectoria a sustituir, e *i* el final. Inicialmente *p* valdrá 1, a medida que se vaya avanzando e incorporando nuevas maniobras *p* se incrementará representando el último punto alcanzado con las mismas, y al mismo tiempo, el de partida para la comprobación de las siguientes. El índice *i* por el

contrario tendrá el valor del último vértice y se irá decrementando cada vez que se compruebe que un determinado fragmento no es sustituible.

```
Función Postprocesado (camino_original)
    p=1;
    Incompleto=falso;
    Mientras i≠Número_de_vértices e incompleto=falso
        i=Numero_de_vértices;
        Mientras i≠p
            Si CuerdaVálida(p, i) Entonces
                EliminaArco(p,i,camino_original,camino_final);
                p=i
            Si no
                Decrementa i en 1
                Si i=p Entonces
                    Incompleto=cierto;
        Fin Mientras
    Fin Mientras
    Si i=Numero_de_vértices Entonces
        Devuelve(camino_final)
    Si no
        PostprocesadoInverso();
    Fin
```

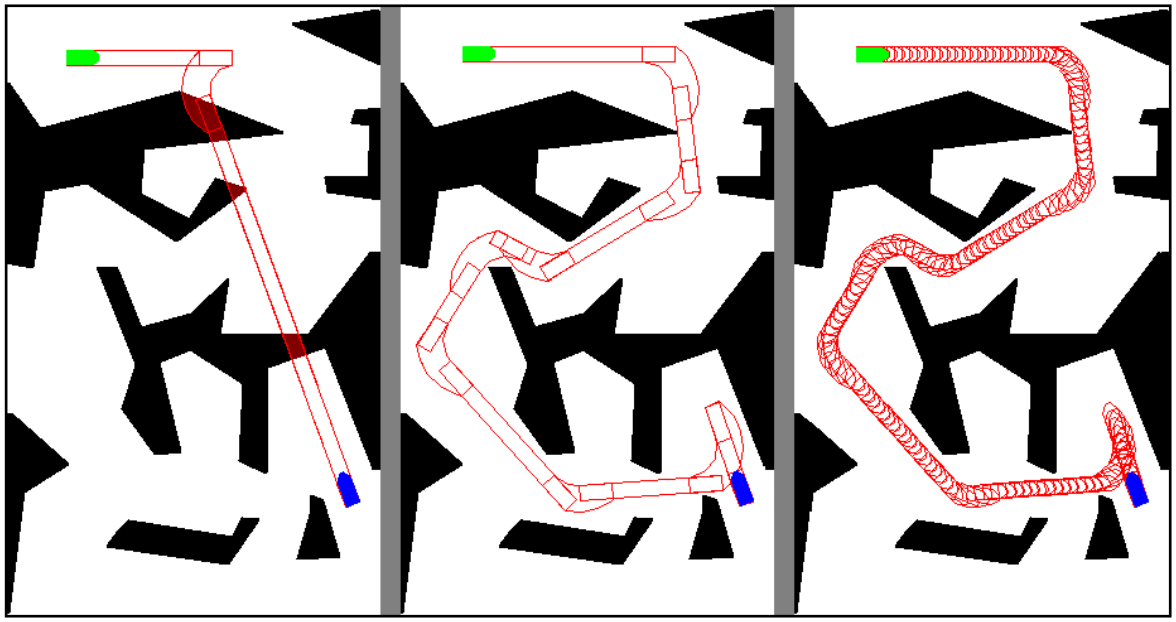
Figura 4.19: Postprocesado con maniobras restringidas

La variable booleana "*Incompleto*" sirve para atestiguar tal eventualidad en el proceso de sustitución de tramos, que se da cuando todas las posibilidades se han comprobado sin éxito. Las funciones "*CuerdaVálida"* y *"EliminaArco"* tienen la misma utilidad que en el postprocesado visto en 2.7.7, es decir, verifican la viabilidad de una maniobra de conexión y la incorporan, pero difieren en la gran cantidad de maniobras que comprueban para un mismo par de puntos a conectar. Por último se tiene especial cuidado en el orden de prueba, comprobando primero las maniobras más directas o

suaves y dejando para el final las demás, que pueden presentar más inversiones (cambios de sentido). Dicha secuencia ha sido mejorada con la experiencia, incorporando instrucciones condicionales que la alteran según sean las configuraciones a unir, pero básicamente consisten en la siguiente serie ordenada: $\{\Phi_x, \Phi_\theta, \Phi_y, \Xi, \Lambda, \Gamma\}$. Las primeras son maniobras restringidas, sin embargo, siempre que permitan alcanzar la posición final son elegidas como primera opción por su simplicidad.

La función *"PostprocesadoInverso"* resume la misma secuencia de instrucciones anterior (es decir, la misma función de Postprocesado) con la salvedad de que el punto de partida será el punto final y viceversa. Existen además algunas incorporaciones adicionales como la posibilidad de aprovechar las maniobras logradas en la primera vuelta, pero en lo demás esta función es igual que la propia función de Postprocesado.

La figura 4.20 muestra un escenario con obstáculos donde se ha llevado a cabo el algoritmo descrito. Inicialmente el algoritmo prueba directamente con las posiciones inicial y final, tratando de conectarlas con una maniobra de conexión. La figura 4.20a muestra los polígonos correspondientes a la maniobra de conexión Ξ. Como puede apreciarse estos polígonos cruzan con varios obstáculos, por lo que esta maniobra es rechazada.



a) Polígono en colisión *b) Polígonos admitidos* *c) Trayectoria final*

Figura 4.20: Evolución del algoritmo de Postprocesado.

Progresivamente el postprocesado logra acomodar maniobra tras maniobra hasta lograr que ninguno de sus polígonos corte con los obstáculos presentes (fig. 4.20b). Una vez hallada la secuencia de maniobras libres de colisión se puede trazar la trayectoria buscada (fig. 4.20c).

Hay varios aspectos relevantes sobre esta novedosa técnica que conviene resaltar. En primer lugar puede inferirse que el proceso de sustitución de tramos puede resultar bastante lento. Evidentemente el número de maniobras a comprobar multiplicadas por el número de posiciones que describen sus trayectorias expresará las veces que el algoritmo consultará la posibilidad de colisión. Así el tiempo ganado en la aplicación de un planificador sin restricciones se vería mermado o incluso superado por el postprocesado. Sin embargo aquí es donde entran en juego los polígonos presentados en el capítulo anterior. Dichos polígonos son suficientes para constatar la posibilidad de colisión de la maniobra completa, reduciendo enormemente el tiempo de cómputo necesario.

Por otro lado, la calidad de la solución obtenida es bastante superior a la devuelta por otros algoritmos de planificación específica para robots con restricciones (como se verá más adelante con el algoritmo RRT adaptado). Dicha calidad puede medirse atendiendo a parámetros como el número de cambios de sentido, el número de cambios de radio de curvatura, longitud recorrida, etc. (parámetros que miden el esfuerzo de control).

Otra consideración clave en la viabilidad del algoritmo de postprocesado concierne a las configuraciones objetivo. El planificador sin restricciones aporta una secuencia de configuraciones cuyos valores exactos no interesan, tan sólo el hecho de que conforman un itinerario que une las posiciones inicial y final. Éstas son las únicas que se han de imponer, las intermedias no han de vincular necesariamente la trayectoria final del postprocesado. Aprovechando esta propiedad se puede alterar el objetivo del algoritmo para que busque configuraciones próximas a la devuelta por el planificador previo, sin por ello perjudicar la eficiencia.

En concreto las maniobras están severamente condicionadas por la orientación, con lo que una cierta flexibilidad en dicho parámetro facilita el acceso de las mismas. Así, en el algoritmo se ha introducido una exploración inicial con las orientaciones más adecuadas a cada maniobra y sólo en el caso de que no sean factibles, se impone la orientación particular del punto dado.

Este último caso corresponde con frecuencia a configuraciones constreñidas por abundantes obstáculos. Por ejemplo pasillos estrechos donde no hay configuraciones de orientaciones diferentes a unas pocas dadas. No obstante, en espacios abiertos, es muy útil poder jugar con dicha flexibilidad.

Esta elección sin embargo sólo puede darse en los puntos intermedios del itinerario, nunca en la configuración final o inicial, que vienen impuestas. La inobservancia de esta cuestión implica un postprocesado poco eficiente, donde en la mayoría de las ocasiones no es capaz de resolver el problema.

La última consideración está relacionada con los pasillos estrechos o “gaps” (Xiaoshan, 2005). En dichas situaciones las configuraciones posibles del vehículo están severamente condicionadas por los obstáculos, dificultando la búsqueda de la solución. Por ejemplo, si el vehículo accede a un sinuoso pasillo orientado en el sentido de avance, permanecerá así en todos los puntos del interior del pasillo, imposibilitando la orientación opuesta. Si la posición final del problema es precisamente ésta y está ubicada en el pasillo, el algoritmo no podrá encontrar la solución. Es por ello que se ha incluido un segundo proceso de búsqueda. Es decir, para evitar el efecto anteriormente mencionado, si el postprocesado acaba sin haber hallado solución, comienza el problema desde el principio intercambiando los puntos de partida y objetivo. De esta forma se utilizará la orientación contraria, dado que el sentido de avance del vehículo es el opuesto, y así se salva este problema.

4.7 Conclusiones

En este capítulo se han explicado las restricciones no holónomas y el problema que suponen para la planificación. Se han detallado las maniobras restringidas y su uso para encontrar soluciones factibles a través del algoritmo de postprocesado. Esta nueva herramienta permite ofrecer trayectorias admisibles para robots con restricciones no holónomas a partir de las ofrecidas por un planificador común (sin restricciones). Las soluciones ofrecidas se adaptan al entorno en el que se desarrollan, de modo que son suaves y sin cambios de sentido cuando es posible, y complejas con alto número de maniobras cuando los obstáculos lo hacen necesario.

En el siguiente capítulo podrán observarse sus resultados, utilizando diferentes planificadores para generar la trayectoria de partida. En especial se estudiará su aplicación con el algoritmo RRT y se comparará la solución del postprocesado con la ofrecida con una variante del algoritmo RRT adaptado a las restricciones no holónomas. Igualmente se aplicará a los resultados obtenidos con Vodec, y también se realizarán estudios comparativos. Como muestra se ofrece la figura 4.21, donde se representan diferentes escenarios sobre los que se ha aplicado el algoritmo RRT seguido del postprocesado.

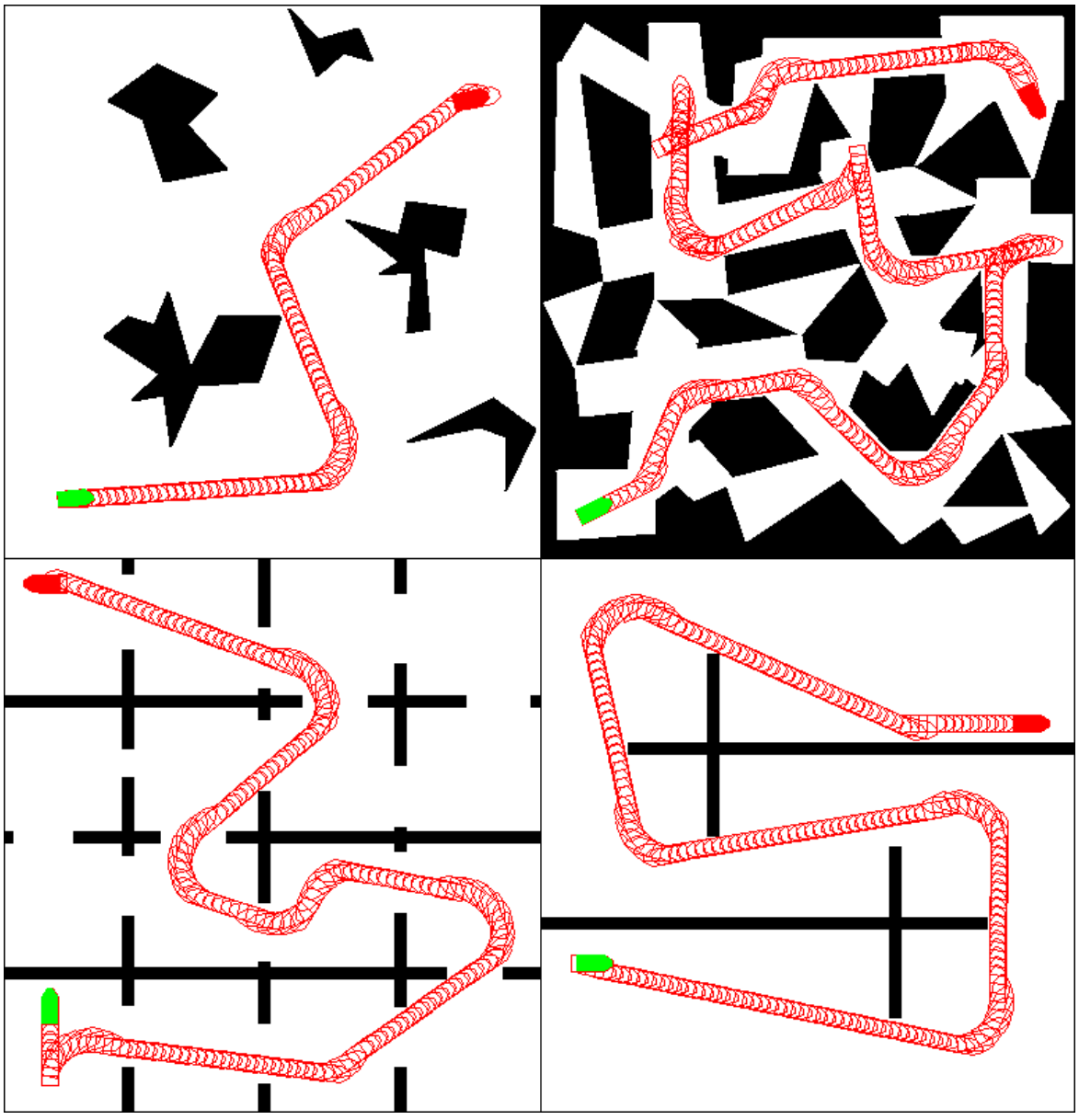


Figura 4.21: Ejecución del algoritmo de postprocesado en diferentes escenarios.

CAPÍTULO 5 Maniobras restringidas sobre diferentes planificadores

5.1 Introducción

En el capítulo anterior se han descrito las restricciones no holónomas y se han visto algunos ejemplos concretos de las mismas. Así mismo se ha presentado un algoritmo, denominado postprocesado, capaz de incorporar maniobras restringidas en base a una ruta dada. El método propuesto en esta tesis consiste en utilizar un planificador sin considerar las restricciones y luego aplicar el algoritmo de postprocesado que sustituye la trayectoria ofrecida en una secuencia de maniobras restringidas que cumplen las restricciones no holónomas.

En los sucesivos apartados de este capítulo se estudiará la aplicación de este método con diferentes planificadores. En primer lugar se mostrará el algoritmo RRT y sus variantes. En segundo lugar se estudiará el postprocesado aplicado a diferentes planificadores habituales en la literatura científica y se comparará la calidad de sus soluciones. De los resultados de esta comparativa nació la idea de Vodec. Así pues, en último lugar y de forma separada se describirán los resultados obtenidos con Vodec, presentando las bondades de este método con respecto a los anteriores.

5.2 Aplicación sobre el algoritmo RRT

En este apartado se estudia la aplicación del algoritmo de postprocesado explicado en el capítulo anterior sobre trayectorias obtenidas con el algoritmo RRT. Dado que el algoritmo RRT en una de sus variantes es capaz de implementar restricciones no holónomas, se comparará dicha variante con el método propuesto en esta tesis.

5.2.1 RRT adaptado a sistemas no holónomos

El algoritmo RRT implementa las restricciones de un modo muy sencillo, aunque con la técnica que se describe a continuación el coste computacional resulta elevado. Como se recordará el algoritmo RRT inicia cada iteración definiendo un punto aleatorio q_{rand} dentro del espacio configuraciones. Después se busca el punto del árbol más cercano q_{near} y a partir de éste se ejecuta la función “Extiende”, pudiendo resultar un movimiento en cualquier dirección. Si existen restricciones este procedimiento deja de ser válido. Por tanto se recurre a la técnica que se describe a continuación.

Sea *U* el conjunto de las órdenes de control asumibles por el robot. La dimensión de *U* será igual al número de parámetros de control independientes que se pueden aplicar (ángulos de giro, velocidad de las ruedas motrices, etc.). Considérese ahora una serie de restricciones no holónomas. Bajo ciertas condiciones es posible identificarlas en la forma de la ecuación vectorial siguiente (Latombe, 1991):

$$\dot{q} = f(q,u) \qquad (5\text{-}1)$$

Donde ***q*** es un vector del espacio de configuraciones *C*, y ***u*** un vector de *U*. Esta ecuación es idéntica a la utilizada en los sistemas de control si se sustituye ***q*** por el vector de estados. Dado que el algoritmo trabaja con valores discretos, se realiza la aproximación discreta de (5-1) y se asume un valor finito de tiempo Δt como paso de integración:

$$q_{k+1} = f(q_k, u_k) \qquad (5\text{-}2)$$

Sea U_{dis} un subconjunto discreto y finito de U ($U_{dis} \subset U$).

$$U_{dis} = \{u_1, u_2, ...u_k ...u_{m-1}, u_m\} \quad (5\text{-}3)$$

De cada $\boldsymbol{u_k} \in U_{dis}$, aplicada durante un tiempo Δt, en virtud de la ecuación (5-2) puede obtenerse una configuración final $\boldsymbol{q_{k+1}}$. A estas m configuraciones así halladas se les llama "*bang-motions*", y constituyen un conjunto finito y discreto de configuraciones próximas a la inicial ($\boldsymbol{q_k}$), debido a que se escoge un Δt pequeño. En realidad se intenta que dicha proximidad sea constante e igual a un valor $\in$ similar a la longitud del segmento de crecimiento estudiado en la función Extiende del RRT básico. Para ello se elige U_{dis} y Δt de modo que generen un conjunto de puntos lo más disperso posible pero equidistantes un valor $\in$ dentro del espacio de configuraciones.

Así pues, para implementar las restricciones se altera la función Extiende. En esta nueva variante del algoritmo RRT, en vez de agregar un segmento en dirección a q_{rand} (como se ha explicado, las restricciones impiden el movimiento en línea recta) se calcula un conjunto de configuraciones próximas a q_{near} que sí cumplan las restricciones (*bang-motions*) y de entre éstas ha de elegirse el q_{new} utilizando la función métrica adecuada. Al igual que en el algoritmo básico, la configuración más próxima es agregada continuando el proceso de modo normal.

Como puede inferirse, el tiempo de proceso asociado a la extensión de nuevas ramas resulta bastante superior, aunque resuelve de forma sencilla el problema de las restricciones. Por otro lado resulta delicado elegir el conjunto de órdenes, pues si es escaso resultará difícil conectar con la configuración final, y si es abundante aumenta ostensiblemente el esfuerzo de cómputo. También la métrica resulta delicada de definir, pues existen parámetros de configuración que pueden tener mayor peso que otros. Por ejemplo en vehículos tipo Ackerman, según la situación no es tan importante la distancia euclídea entre dos configuraciones próximas como la orientación de las mismas.

En la figura 5.1 se muestra la solución aportada utilizando esta técnica en un vehículo diferencial. El conjunto de órdenes de control utilizado genera las configuraciones correspondientes a un avance recto, un retroceso recto, y la reorientación sin desplazarse en varios ángulos.

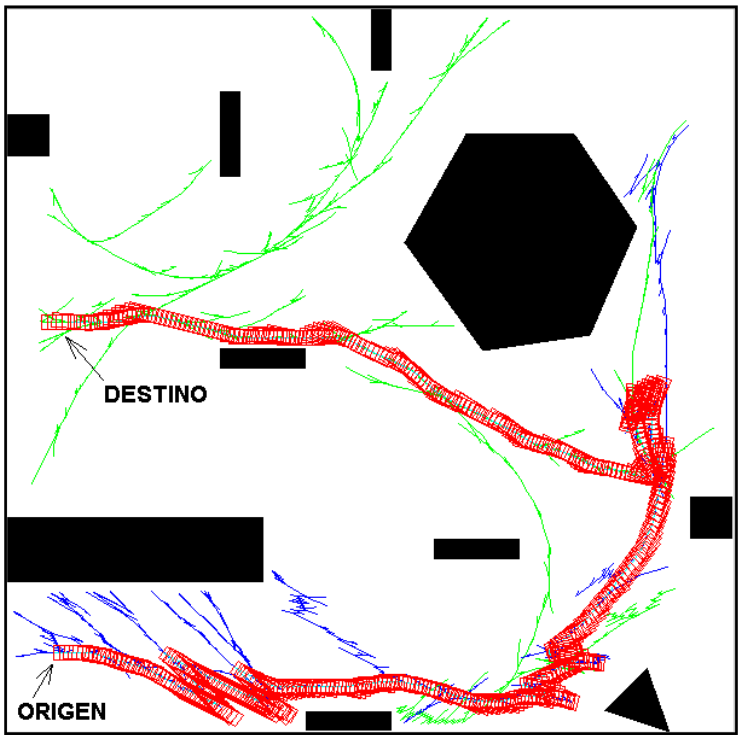


Figura 5.1: Solución del RRT adaptado para un vehículo diferencial.

En la figura 5.2 se ofrece el resultado del RRT adaptado a las restricciones de un vehículo tipo Ackerman. En este caso se han elegido como órdenes de control las correspondientes al avance y retroceso del vehículo, y el giro del volante a un ángulo a escoger dentro de un conjunto discreto. Como resultado se obtienen las configuraciones correspondientes al avance y retroceso en línea recta y con distintos radios de curvatura.

En ambos casos puede observarse la trayectoria típicamente sinuosa y salpicada de cambios de sentido ofrecida como solución por el algoritmo. Tal y como se explica en el capítulo 2, el algoritmo de postprocesado tiene como desempeño limar estas discontinuidades en los parámetros de control del robot. Sin embargo, mientras que en el caso holónomo esto no supone ningún problema, en el caso no holónomo implica serias dificultades nada fáciles de solventar.

Aún así, lo meritorio de este método es su sencillez de implementación. Es fácilmente escalable a sistemas con muchos más parámetros de configuración y, sea cuales fueren las restricciones, no suponen un obstáculo. Basta con la capacidad de calcular la configuración obtenida a partir de otra dada ejecutando un conjunto de ordenes discretas, y nada más.

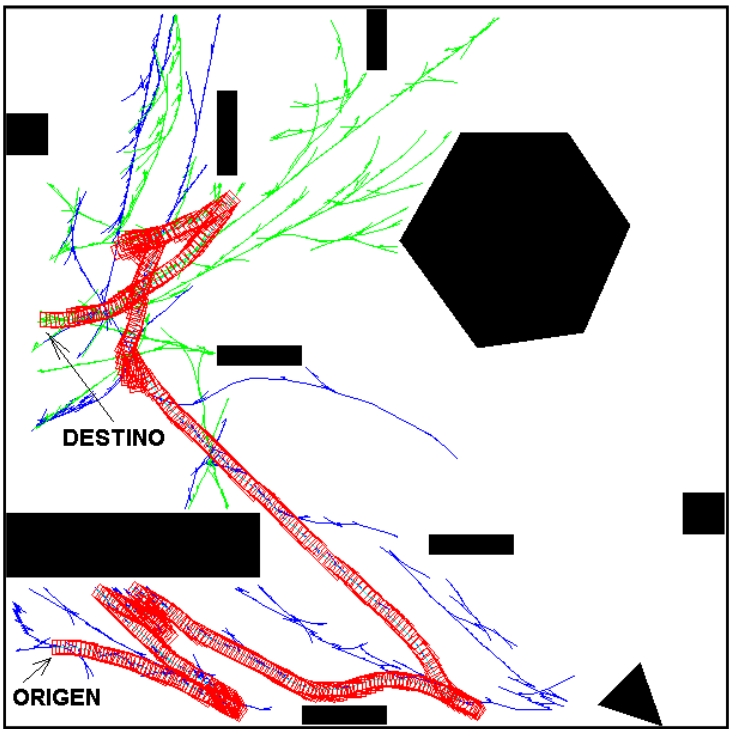


Figura 5.2: Solución del RRT adaptado para un vehículo tipo Ackerman.

5.2.2 RRT con maniobras restringidas

La versión del RRT que se acaba de explicar, presenta elevados tiempos de cómputo y la necesidad de un postprocesado complejo que ha de contender con restricciones no integrables. Sin embargo, la combinación del RRT con las maniobras restringidas que se describen a continuación, constituye un innovador método de planificación (Gomez Bravo y otros, 2008a) de elevadas prestaciones que al menos supera en velocidad al RRT adaptado (usando las "*bang-motions*") en varios escenarios (López y otros, 2004).

Supóngase un escenario con obstáculos en el que hay un robot con restricciones no holónomas ocupando una configuración inicial dada y que se desea conocer una trayectoria válida para alcanzar cierta configuración dada. Éste es el problema de planificación típico en sistemas no holónomos. El método que aquí se presenta se inicia generando primero una trayectoria con el RRT ***sin imponer ninguna restricción***. De este modo se obtiene un camino de forma muy rápida con respecto a la opción de utilizar

el RRT adaptado a las restricciones. A continuación se ejecuta el algoritmo de postprocesado descrito en el capítulo anterior (sección 4.6) que utiliza las maniobras restringidas apoyándose en la trayectoria obtenida. Se ha aplicado este método en dos tipos de sistemas no holónomos: el robot diferencial y el robot tipo Ackerman. En las figuras 5.3 y 5.4 se presenta un experimento realizado con un robot tipo Ackerman. La primera presenta la solución devuelta por el RRT sin restricciones. La segunda el efecto del postprocesado. En éste, la unión de dos configuraciones se realiza mediante diferentes maniobras de conexión. De este modo, se segmenta el camino completo descomponiéndolo en una secuencia de movimientos acordes con las restricciones no holónomas.

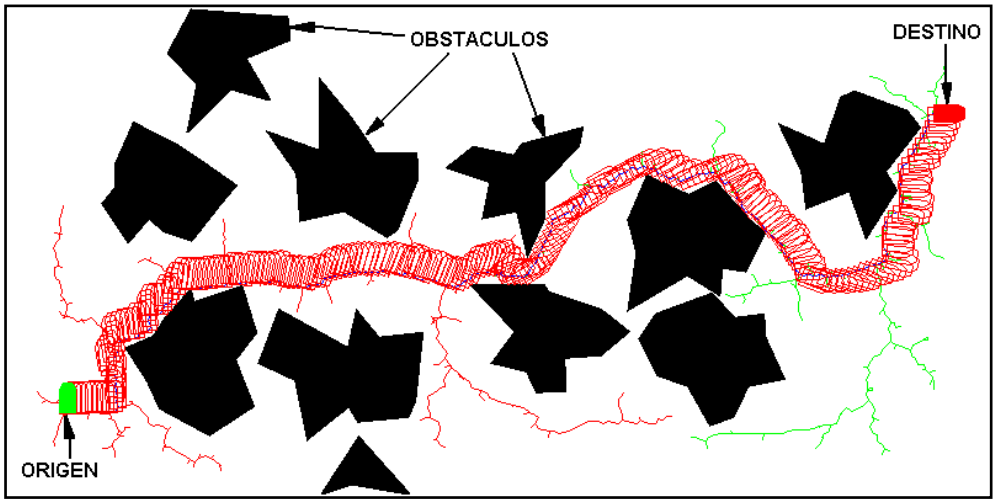


Figura 5.3: Resultado del RRT sin restricciones.

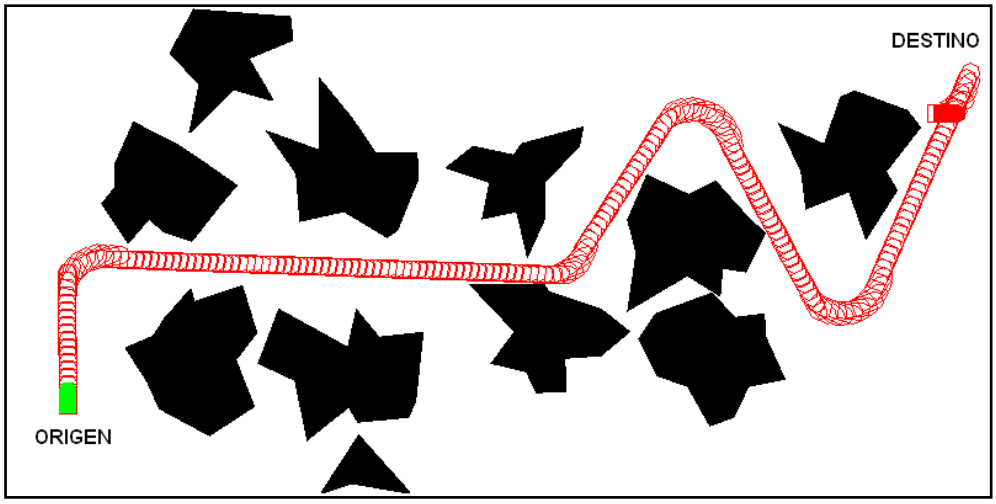


Figura 5.4: Resultado del postprocesado con maniobras restringidas.

Pero lo que sin duda resulta más interesante es el valor comparativo entre éste nuevo algoritmo y el RRT adaptado a las restricciones. Para tal efecto considérese el escenario de la figura 5.1. En ésta se muestra el resultado de la aplicación del RRT adaptado. En concreto, si se aplica esta técnica en el mismo escenario en que se ejecutó el RRT con restricciones, se obtiene una trayectoria de mayor calidad y en un tiempo hasta diez veces menor (véase la figura 5.5 y la tabla 5.1).

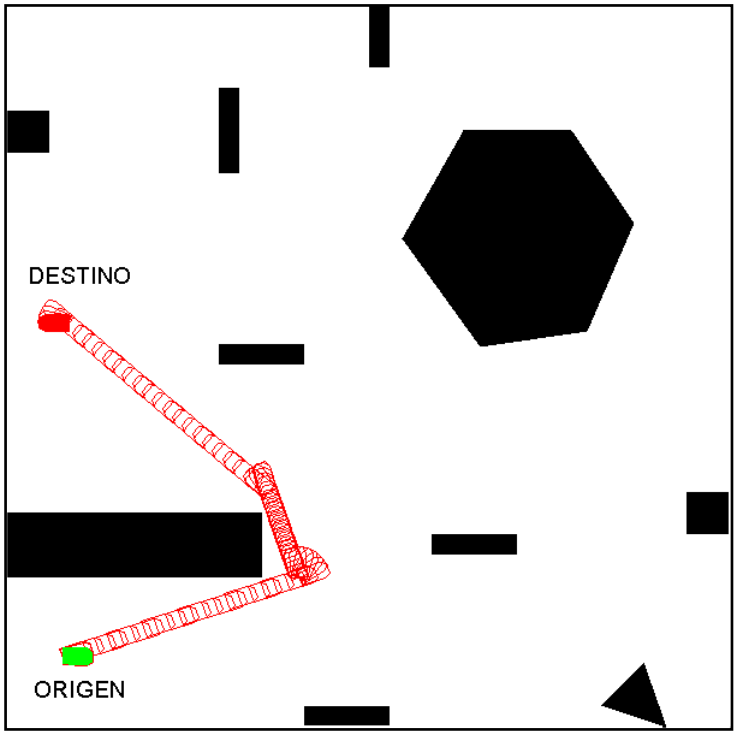


Figura 5.5: Nuevo método aplicado al robot diferencial.

Para las comparativas se ha utilizado una CPU con un microprocesador AMD Athlon 1700 (1,46 GHz) y 1 GB de RAM. La aplicación está desarrollada en JAVA, y se ejecuta sobre un sistema operativo Windows XP. Realizados diez experimentos entre un método y otro, los resultados son notorios (tabla 5.1). El nuevo método de postprocesado no sólo resulta mucho más rápido, sino que la solución proporcionada presenta una longitud bastante inferior.

Además, desde el punto de vista del control, esta solución resulta mucho más simple; por el contrario, la ofrecida por el RRT adaptado, presenta cambios constantes de las consignas de control y sería necesario aplicarle algún tipo de postprocesado que aumentaría aún más el tiempo total requerido.

Tabla 5.1: Comparativa entre el algoritmo RRT adaptado (Dif-RRT) y el algoritmo RRT con maniobras restringidas (Dif-Postprocesado), en un vehículo diferencial.

	Dif-RRT		Dif-Postprocesado	
	Media	***Desv. Est.***	**Media**	***Desv. Est.***
Tiempo (ms)	7620,0	*3563,8*	652,7	*93,8*
Longitud (m)	1796,9	*341,0*	1061,3	*1758,1*

Para el caso del vehículo tipo Ackerman el RRT adaptado debía contender con más restricciones (ver figura 5.2). Esto se traduce en un elenco de configuraciones alternativas alcanzables desde una dada ("*bang-motions*") más reducido. Por tanto los tiempos de cómputo para hallar una solución aumentan con respecto al diferencial. A continuación se muestra para el mismo escenario la trayectoria ofrecida por el nuevo método (ver figura 5.6).

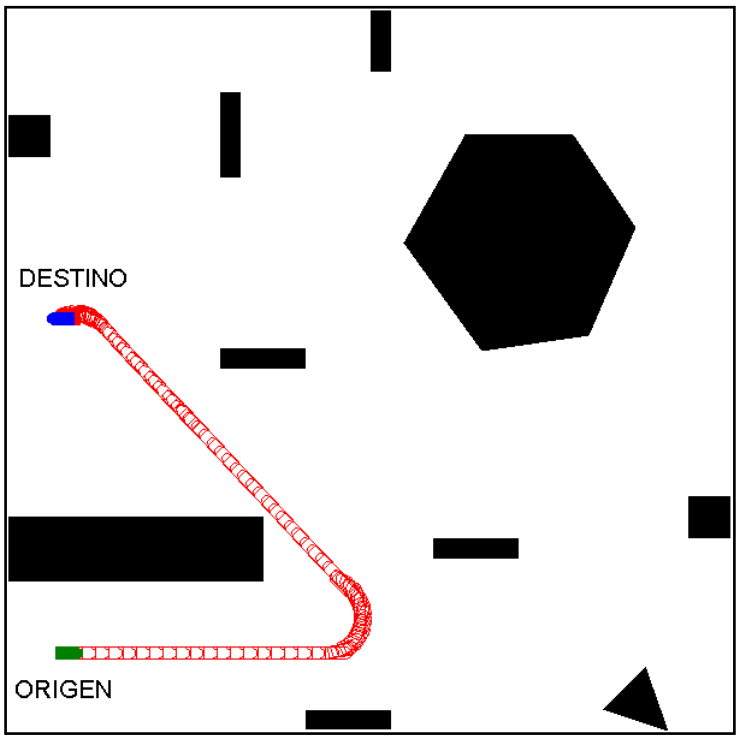


Figura 5.6: Nuevo método aplicado al robot tipo Ackerman.

Los resultados, tal y como cabía esperar tras lo visto en el robot diferencial, son excelentes, superando las distancias entre los anteriores. Por un lado el algoritmo resulta mucho más rápido (tabla 5.2), y por otro presenta una solución de mejor calidad.

Dada la relevancia y mayor complejidad de las soluciones de este tipo de vehículos no holónomos (Ackerman), el trabajo de investigación se ha centrado sobre los mismos, estudiando su comportamiento en diferentes escenarios como por ejemplo el que se puede apreciar en las figuras 5.7 y 5.8.

Tabla 5.2: Comparativa entre RRT adaptado (Ackerman-RRT) y RRT con maniobras restringidas (Ackerman-Postprocesado), en un vehículo Ackerman.

	Ackerman-RRT		Ackerman-Postprocesado	
	Media	***Desv. Est.***	**Media**	***Desv. Est.***
Tiempo (ms)	11408,2	*7146,7*	701,8	*73,5*
Longitud (m)	1731,9	*310,0*	847,6	*3,4*

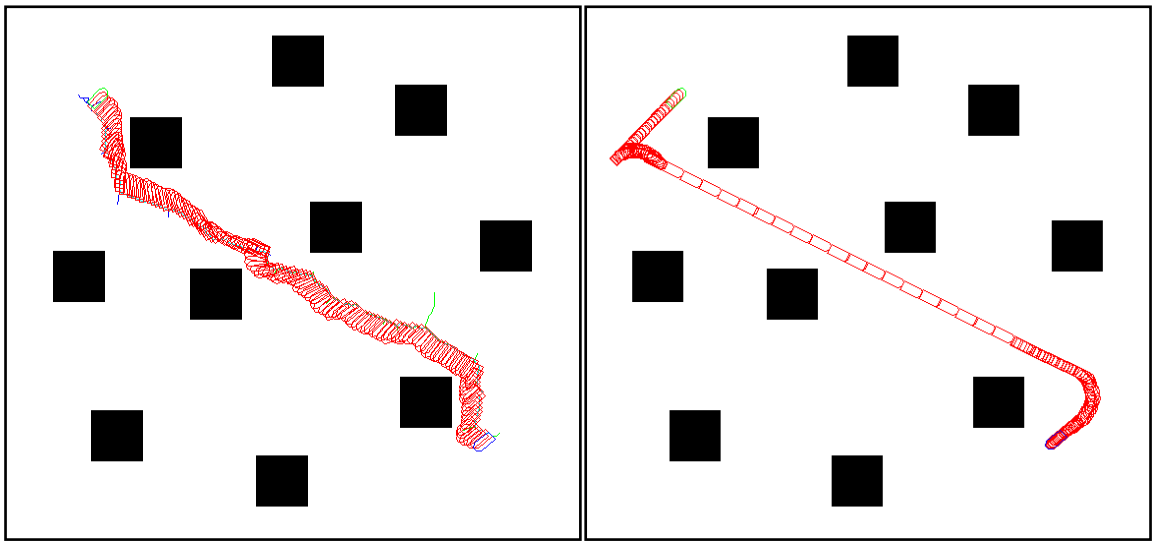


Figura 5.7: Comportamiento del nuevo método en escenarios despejados.

En los escenarios en que existen pocos obstáculos el método resulta rápido y eficaz, logrando construir trayectorias con escaso número de maniobras (ver figura 5.7). Sin embargo es en los escenarios más poblados donde la abundancia de colisiones puede aumentar el tiempo de cómputo. No obstante, el nuevo método logra tiempos de

resolución inferiores al segundo (ver figura 5.8), equiparándose en orden de magnitud a los del caso anterior.

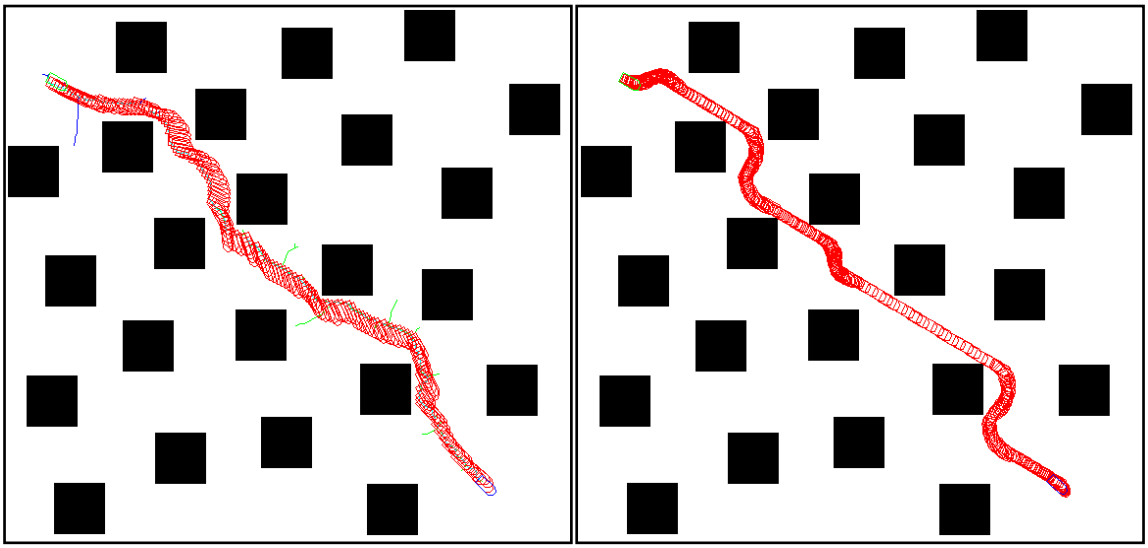


Figura 5.8: Comportamiento del nuevo método en escenarios muy poblados.

Resulta conveniente señalar que un escenario de abundantes obstáculos no implica unas maniobras excesivamente complejas, como se demuestra en la figura 5.8. Dada la eficiencia del método se debe imponer un escenario especial para poder elevar la dificultad ante el algoritmo. Para ello se muestra el escenario de la figura 5.9, donde aparecen pasos estrechos ineludibles.

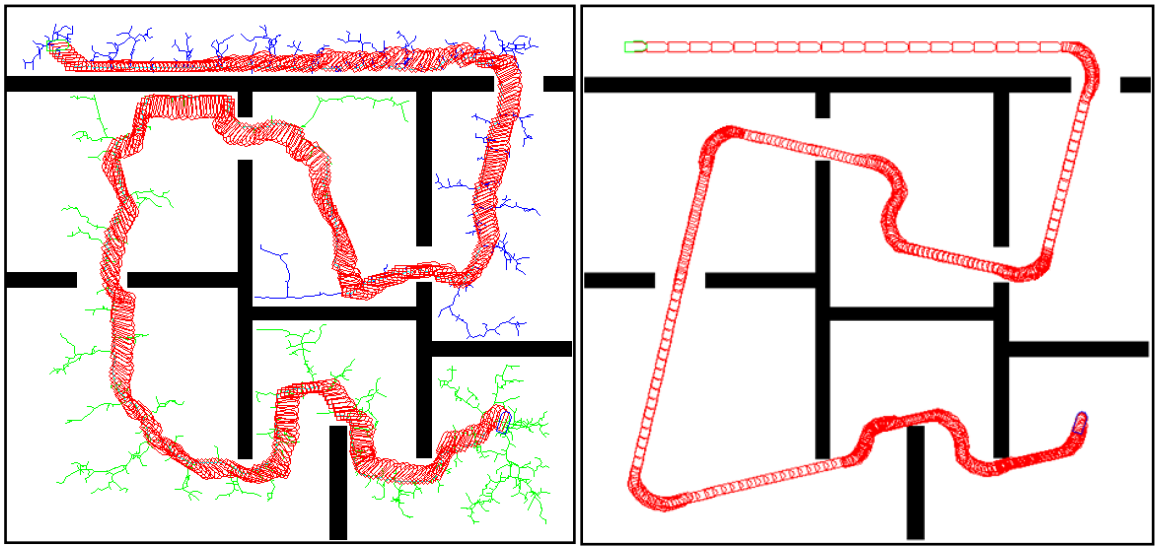


Figura 5.9: Comportamiento del nuevo método en escenarios con puntos de paso.

Un tipo de problema muy frecuente en este tipo de vehículo consiste en el aparcamiento. En la siguiente figura (ver figura 5.10) se muestra el resultado del nuevo método aplicado en dos situaciones típicas de aparcamiento. Como se puede observar el algoritmo opta por maniobras muy similares a las que utiliza un conductor humano.

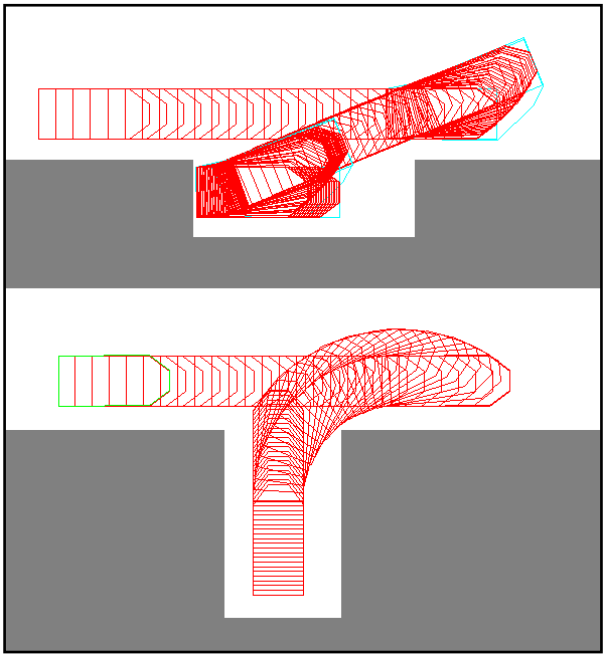


Figura 5.10: Comportamiento del nuevo método en aparcamientos

En la figura 5.9 se muestra uno de los casos más dificultosos de resolver en sistemas no holónomos. Consiste en escenarios que presentan áreas reducidas de paso obligado circundadas por obstáculos muy próximos, permitiendo sólo un pequeño conjunto de configuraciones posibles. En tales casos el RRT sin restricciones consume más tiempo del habitual en resolver el problema (20 segundos); pero siempre muy por debajo del que utilizaría el RRT adaptado (el basado en las "*bang-motions*"). El algoritmo de postprocesado en este ejemplo concreto sólo empleó 1.2 segundos en ofrecer la solución de la fig. 5.9 derecha (una vez hallada la trayectoria por el algoritmo RRT).

Es en este tipo de casos donde interesa conjugar polígonos muy ajustados a la maniobra con idea de salvar los obstáculos en el momento de comprobarlos. En la implementación existe la opción de reducir el segmento de dichos polígonos, mejorando las oportunidades de éxito.

Además de la dificultad de evitar la colisión, los pasillos estrechos conllevan otro problema. Cuando el vehículo entra en él, sólo podrá recorrerlo con una orientación que

no podrá variar hasta que no abandone el mismo. Esto puede no ser significativo en escenarios como el de la figura 5.11 donde antes de entrar en el pasillo y al salir existe cierto espacio para poder maniobrar y alcanzar las posiciones extremas de la trayectoria.

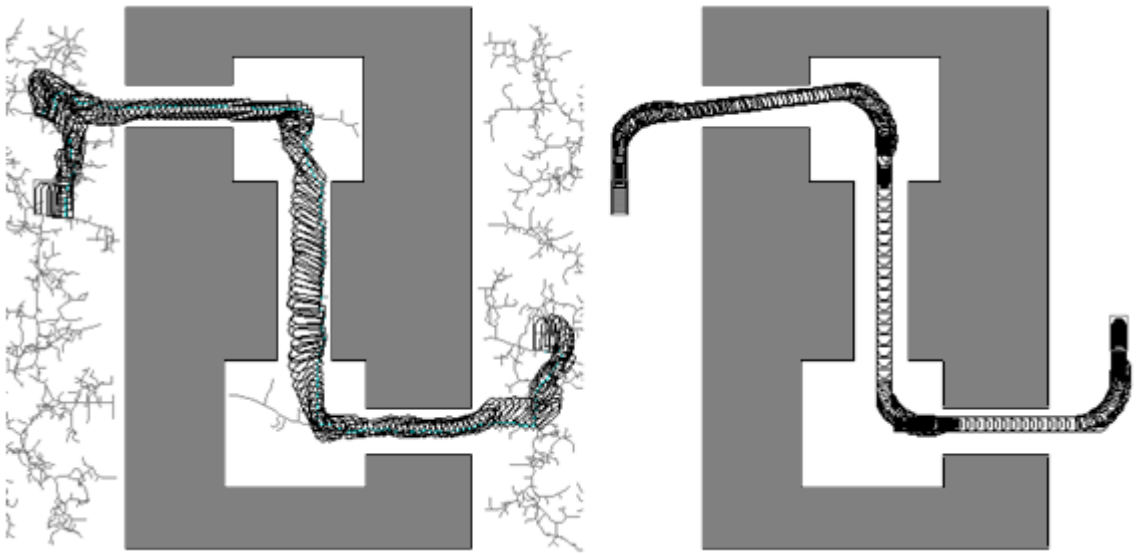


Figura 5.11: Escenarios con pasillos estrechos.

Sin embargo, como puede observarse en la figura 5.12 esto no ocurre. Es decir, existen escenarios donde la meta esté precisamente dentro del pasillo. En el caso mostrado en la figura 5.12 da la casualidad de que la configuración final se encuentra en el sentido de avance, de modo que cuando el algoritmo se acerca a la meta no tiene problemas para hallar una maniobra que la conecte.

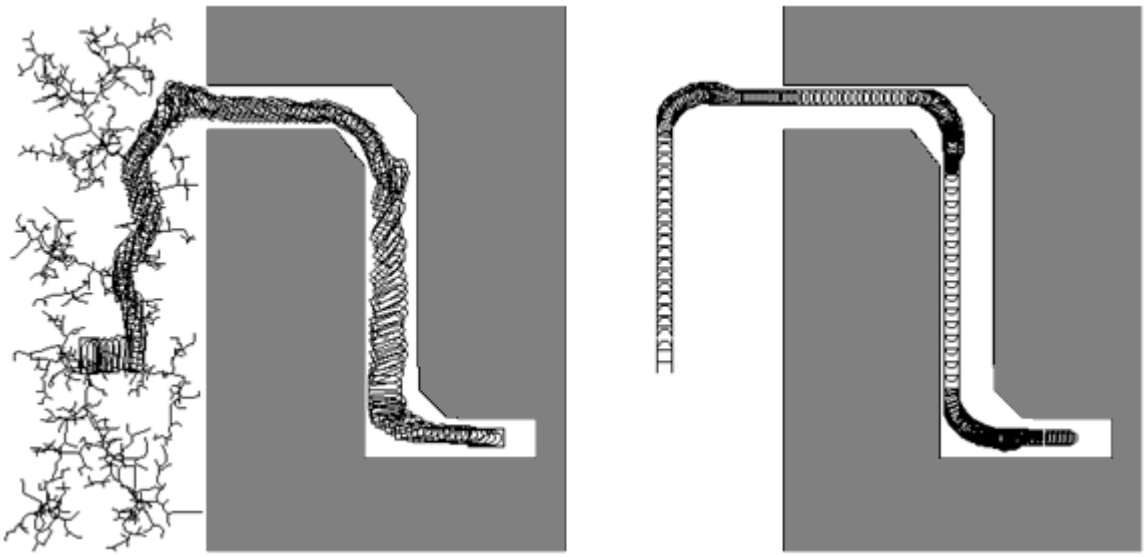


Figura 5.12: Escenarios con configuración objetivo al final de un pasillo.

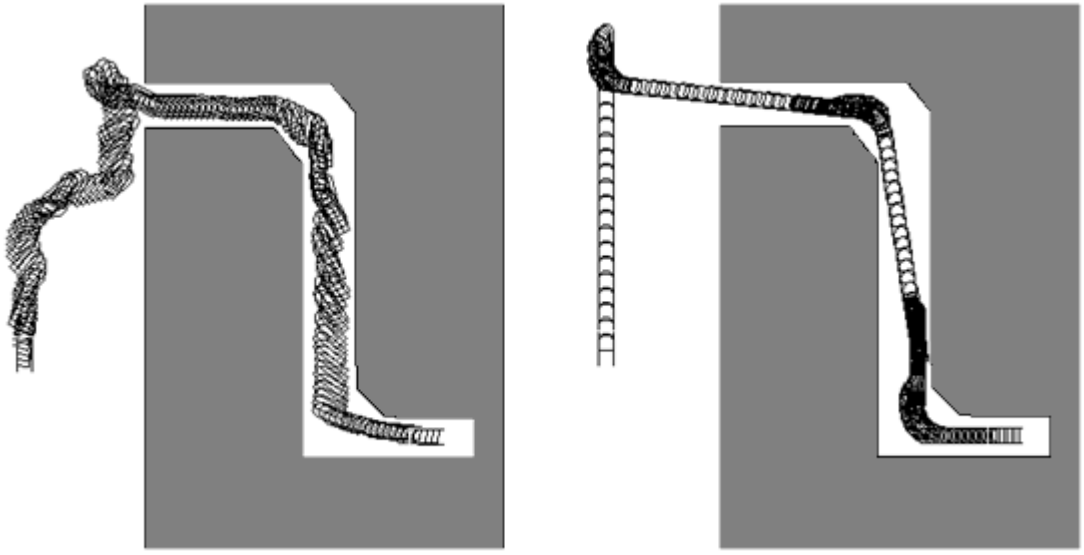


Figura 5.13: Escenarios con configuración objetivo al final de un pasillo en contrasentido.

No obstante, si la orientación es justo la contraria, el algoritmo podría fallar y no encontrar la solución. Su éxito dependería íntegramente del modo en que la incorporación de las maniobras restringidas definiera el sentido de circulación dentro del pasillo. Esta circunstancia es fortuita, pues depende de la trayectoria devuelta por el árbol y la configuración de los obstáculos.

Por ello, como se indicó en el apartado 4.6.2, se introduce al final una segunda vuelta en caso de fracaso, que reinicia el algoritmo intercambiando las posiciones de partida y meta. El resultado se puede apreciar en la figura 5.13, donde se alcanza sin problemas la posición deseada maniobrando marcha atrás desde la embocadura del pasillo.

Otros aspectos importantes que han salido a relucir durante la investigación, sobre todo en casos con alto número de obstáculos, han sido los siguientes:

- El radio de curvatura mínimo tiene un peso decisivo en el cómputo de la solución. Con radios pequeños el algoritmo suele ser rápido, ofreciendo una solución muy próxima a la trayectoria ofrecida por el RRT en pocos segundos. A medida que se va aumentando el radio mínimo, el postprocesado va incrementando su tiempo de cómputo hasta que a partir de un umbral no proporciona soluciones. Aunque dicho comportamiento era de esperar, lo que resulta relevante es la alta sensibilidad que existe con respecto a este parámetro.

- La calidad de la solución está fuertemente influida por el orden en que se prueban las maniobras. Este aspecto no ha sido nada fácil de resolver, e incluso actualmente se sigue investigando en nuevas posibilidades de estructurar esta fase del algoritmo en pro de obtener las soluciones mejores. Actualmente el programa ubica el punto objetivo con respecto al punto de partida, y según su situación da prioridad a una maniobra u otra. Sin embargo, siempre existe la posibilidad que la maniobra más extraña o dificultosa sea precisamente la única que resuelva el problema, por lo que no debe ser excluida. Ello implica una estructura de decisión bastante larga y compleja que debe ejecutarse en cada iteración.

- Se ha observado que cuando la trayectoria ofrecida por el RRT está muy próxima a los obstáculos, después resulta pernicioso para la comprobación de polígonos del postprocesado.

- También resulta crítica la elección de los polígonos que inscriben las maniobras. Si éstos tienen abundantes puntos se eleva en demasía el tiempo de cómputo; pero si tienen escasos vértices, incluyen una nada despreciable área vacía que puede originar colisiones inexistentes (ver sección 4.5).

- Una técnica lenta, pero que ofrece soluciones de aún mayor calidad en escenarios muy poblados de obstáculos, consiste en lo siguiente. Dadas las dimensiones máximas del robot, se construye el rectángulo definido por las mismas reducidas en alguna cantidad (por ejemplo un 10% de la dimensión del vehículo). Tomando como planta del vehículo el rectángulo calculado, se procede a obtener la trayectoria. Dado que la planta es menor, los polígonos asociados a las maniobras de conexión también serán menores, y la probabilidad de que sorteen los obstáculos aumentarán. Si no se obtiene solución se puede reducir aún más el rectángulo. Si se obtiene, se dispondrá de una trayectoria suave y que cumple las restricciones, proporcionando una base mejor que las trayectorias ofrecidas por el algoritmo RRT sin restricciones. Es decir, si se aplica de nuevo el postprocesado utilizando la planta real del vehículo sobre dicha trayectoria, las probabilidades de obtener una solución aumentan considerablemente. La razón estriba en que la trayectoria previa será muy similar a la que se desea obtener, y en aquellos tramos en los que esto no suceda, al menos se dispondrá de posiciones previas y posteriores mucho más acertadas que las que ofrece el RRT. Todo ello ha quedado corroborado con la experiencia obtenida en diferentes experimentos.

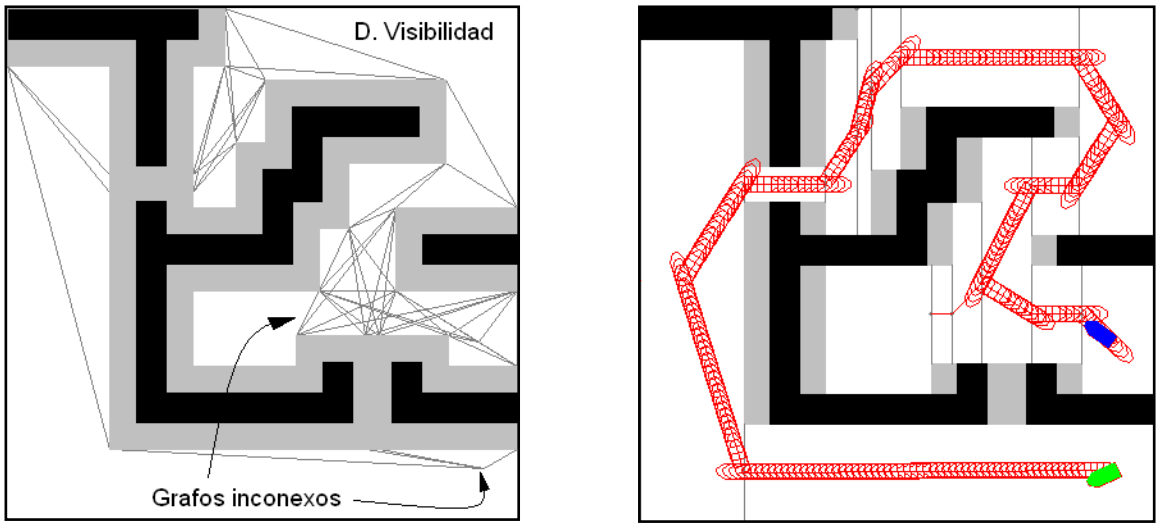
5.3 Postprocesado aplicado a otros planificadores

Como se ha podido comprobar, el algoritmo de postprocesado sólo requiere una trayectoria de partida, una sucesión de configuraciones próximas entre sí que bien puede obtenerse con cualquiera de los métodos de planificación expuestos en el capítulo 2. En concreto se han elegido dos para ilustrar las ventajas e inconvenientes del RRT con respecto a los demás. Uno de ellos es el diagrama de visibilidad, y otro la descomposición vertical, subtipo de la descomposición en celdas exacta. Para ilustrar esta comparativa se ha elegido un mismo escenario y se han ejecutado sobre éste los distintos planificadores (figura 5.14). Los diferentes aspectos estudiados se detallan en las siguientes secciones.

5.3.1 Eficacia

A la hora de combinar el algoritmo de planificación sin restricciones con el postprocesado, la primera cualidad de importancia es la capacidad de hallar soluciones si éstas existen. En el algoritmo de visibilidad y el de descomposición en celdas esta cuestión sería una certeza de no ser por la necesidad de revestir de un margen de seguridad el contorno de los obstáculos. Como se indicó en secciones anteriores, dicho margen consiste en la longitud del punto más alejado del robot a su centro de referencia. En un robot circular esto no supondría ninguna merma, pero en cualquier otro puede implicar la incapacidad de hallar una solución que existe. Esto se ilustra en la figura 5.14, donde para un mismo escenario se han ejecutado los tres planificadores sin postprocesado. En el primer recuadro (figura 5.14a) puede observarse el efecto del ensanche de los obstáculos para el diagrama de visibilidad. El resultado es la incapacidad del método para hallar una solución, aunque ésta exista.

En el método de descomposición en celdas el ensanche se toma sólo en la dirección de la abscisa, dado que la naturaleza del método asegura el máximo distanciamiento en el sentido de la ordenada. En el ejemplo de la figura 5.14b, se logra una solución, aunque ésta no es la más corta. Ello es debido a que el estrecho horizontal más próximo no puede ser franqueado al ensancharse los obstáculos y solaparse.



a) Diagrama de Visibilidad

b) Descomposición en celdas exacta

c) RRT

Figura 5.14: Comparativa entre distintos planificadores

Así queda demostrado que este método mejora la capacidad de hallar soluciones con respecto al anterior, pero no asegura totalmente su eficacia. El tiempo de cómputo de esta solución fue de 110 ms.

Por último, en el caso del RRT (figura 5.14c) sí se obtiene la solución más próxima, pero en un tiempo de cómputo ostensiblemente superior (504 ms). Era de esperar, debido a que la naturaleza aleatoria del método le otorga escasa probabilidad de hallar configuraciones específicas. Esta debilidad con los llamados “gaps” o pasos

estrechos ha sido resaltada y estudiada en publicaciones previas (LaValle y Kuffner, 2001).

No obstante, en cuanto a la capacidad general del método para hallar la solución, depende de las prestaciones del computador, ya que, según enunció su autor (LaValle y Kuffner, 2000), la probabilidad de que el RRT obtenga una solución depende del número de iteraciones. Es decir, con el suficiente tiempo y la necesaria capacidad de memoria para almacenar los árboles, siempre se podrá hallar la solución sea cual sea el escenario. Esta conclusión teórica deviene de que cuando el número de iteraciones tiende a infinito, la probabilidad de incorporar cualquier configuración tiende a uno.

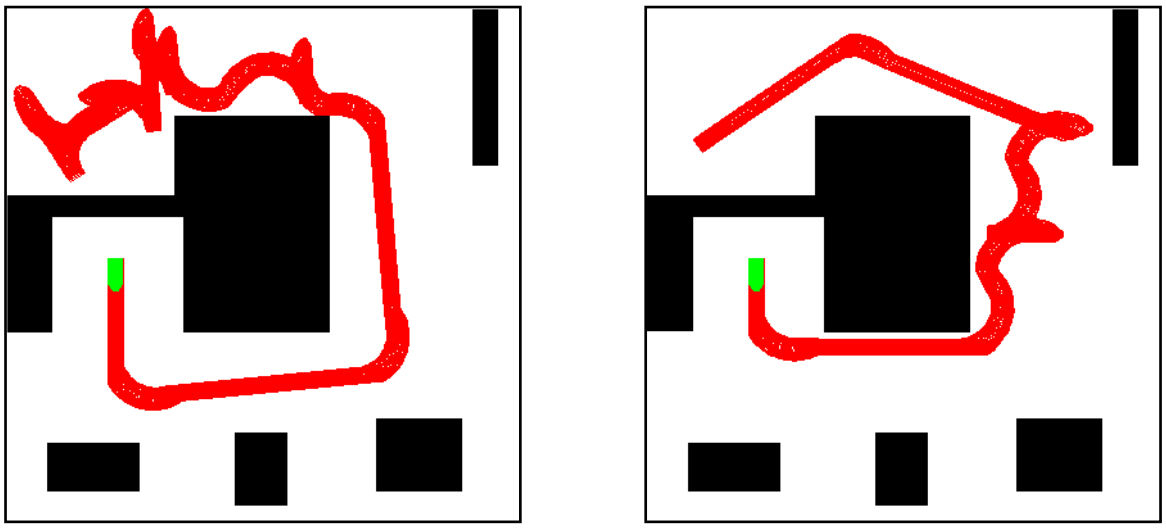
En la práctica, en según qué escenarios, los tiempos de cómputo pueden hacer inviable el uso de este método. Es el caso de abundancia de pasos estrechos o espacios de configuración muy restringidos.

5.3.2 Calidad

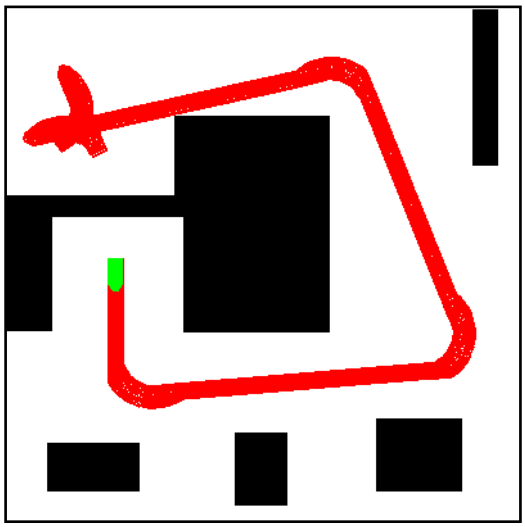
La siguiente virtud en un planificador ante el postprocesado consiste en la capacidad de generar rutas lo más alejadas posible de los obstáculos. Esta característica surgió durante la investigación de modo natural al advertir que en un mismo escenario las diferentes trayectorias ofrecidas por el RRT podían permitir o impedir el hallazgo de una solución en el postprocesado. La causa está en que éste utiliza las posiciones de la trayectoria como “puntos de apoyo” para colocar sus combinaciones de maniobras restringidas. Los polígonos que las envuelven siempre rodean a este punto de apoyo extendiéndose en alguna dirección. Cuanto más cerca estén los obstáculos de dichos puntos, mayor es la probabilidad de que los polígonos los intersequen imposibilitando así las maniobras. Es más, aun en el caso de que se permita alguna solución, la calidad de las mismas se verá seriamente afectada en virtud de lo apartadas que estuvieran las trayectorias de partida. El porqué reside en el mismo principio anterior, pues las posiciones apartadas permiten combinaciones que salvan tramos más largos, dejando en la solución final menos puntos de parada y cambios de sentido (inversores). De esta forma se logran trayectorias más suaves y continuadas. Se entiende por tanto en este apartado que una ruta aumenta su calidad cuando se aleja más de los obstáculos,

comprendiendo con ello la implicación en una calidad final en el sentido más convencional (más cortas, más suaves, con menos cambios de sentido, etc.).

En la figura 5.15 puede observarse un mismo escenario sobre el que se han practicado los distintos planificadores. En el primer caso (figura 5.15a) el algoritmo de visibilidad ofrece una trayectoria costosa de resolver para el postprocesado. Este comportamiento es característico del algoritmo si bien depende exclusivamente del escenario planteado.



a) Diagramas de Visibilidad *b) RRT*



c) Descomposición en celdas

Figura 5.15: Comparativa de calidad entre los distintos planificadores.

El siguiente recuadro (figura 5.15b) corresponde al resultado con el RRT. Aunque este método mejora al anterior, tiene también tendencia a acercarse a los obstáculos. Ello se debe a que la función Extiende, envía ramas en dirección al otro árbol al tratar de conectar. Si existe algún obstáculo, el RRT agregará ramas hasta alcanzarlo, llenándose de nodos en torno a éstos. No obstante, su naturaleza aleatoria le permite distanciarse eventualmente, ofreciendo puntos muy valiosos para el postprocesado.

Finalmente la descomposición en celdas se revela como el mejor planificador en este sentido, ya que por definición sitúa la trayectoria en los puntos medios del espacio libre (al menos en una de las coordenadas). Prueba de ello es el último de los recuadros (figura 5.15c), donde se muestra la trayectoria obtenida. Con respecto a las anteriores resulta la mejor (menor número de paradas, de cambios de sentido, de longitud recorrida marcha atrás, etc.) y además se ha logrado en un tiempo inferior a las otras (150 ms).

5.3.3 Tiempo de cómputo

Siempre que se comparan algoritmos se analiza su tiempo de cómputo. El del postprocesado depende fuertemente de la trayectoria de partida, además del grado de aglomeración de obstáculos. El método completo tendrá una duración que será la suma de la del planificador con la del postprocesado. De esta forma, un planificador que dé soluciones de excelente calidad pero sea ostensiblemente lento tampoco resulta conveniente.

En los experimentos anteriores se ha ido ofreciendo el tiempo de cómputo incluyendo el postprocesado, con lo que se podría deducir que el RRT es el más lento. Aunque en ciertas situaciones prácticas sea cierto, resulta ser una percepción engañosa. Como ejemplo de ello se propone el escenario de la figura 5.16, donde los tiempos de cómputo del RRT, Visibilidad y descomposición en celdas (sin postprocesado) han sido: 81, 651 y 401 ms respectivamente.

Puede observarse la ventaja del RRT en este problema particular. Esta diferencia se debe a la independencia del RRT con el número de vértices de los obstáculos y el tamaño general del escenario al contrario que los otros dos métodos que han de procesar todos los elementos del escenario (incluso los más periféricos que no afectan en absoluto a la trayectoria). Claramente se trata de un escenario atípico, pero sirve para

ilustrar de modo sencillo las fortalezas y debilidades de cada método. En concreto evidencia la idoneidad del RRT como planificador local.

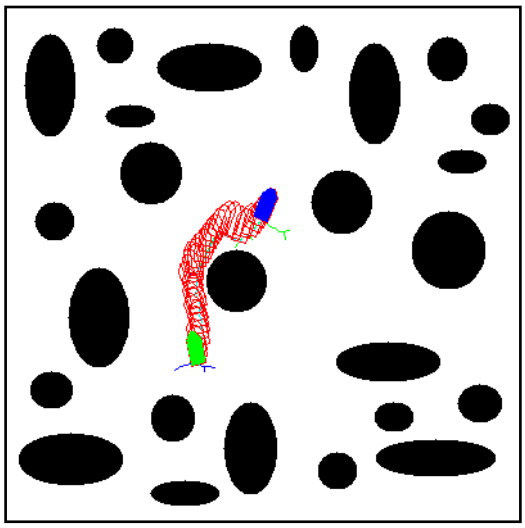


Figura 5.16: Independencia del RRT ante el número y forma de los obstáculos.

5.3.4 Conclusiones

Resulta difícil elegir de entre los métodos mencionados un planificador óptimo que preceda al postprocesado. Por un lado los diagramas de visibilidad son los más usados en robótica por su sencillez de implementación y velocidad de cómputo y, si bien es cierto que a veces no hallan la solución, en la mayoría de las situaciones reales sí que lo hacen. Era obligado estudiar su viabilidad para el postprocesado, sin embargo este trabajo concluye que no es el más idóneo.

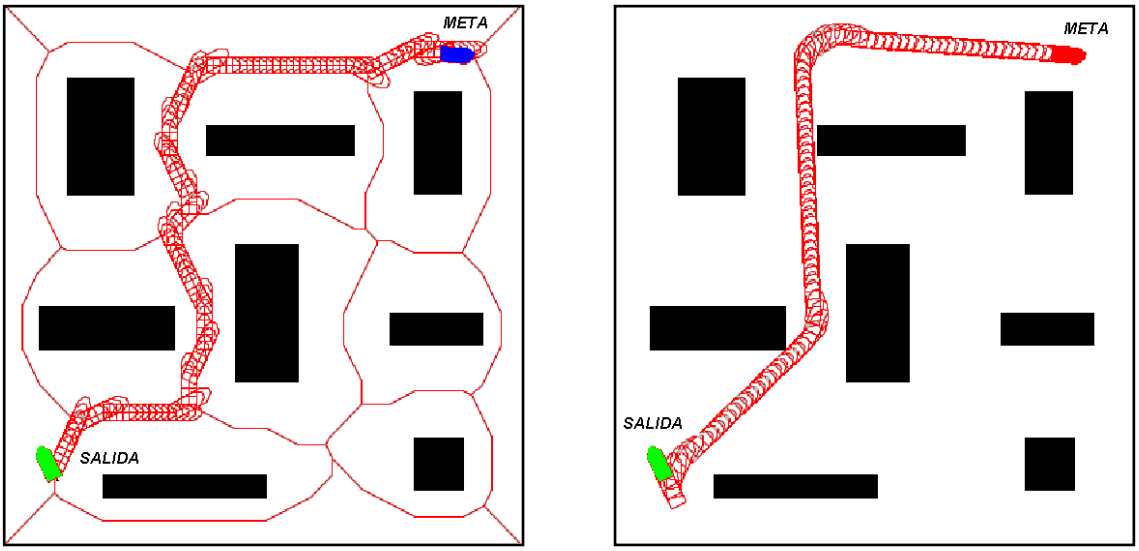
De entre los dos restantes la descomposición en celdas se sitúa como la más competitiva tanto en calidad como en velocidad en los escenarios habituales (bajo número de vértices). Su debilidad estriba en la eficacia. Este método no es capaz de adaptarse al espacio de configuraciones disponible, como sí hace el RRT, sino que ha de recurrir al ensanche de los obstáculos para asegurar una trayectoria válida. Para evitar esto, bastaría modificar la trayectoria que ofrece sin ningún ensanche, de modo que garantizase un alejamiento horizontal de los obstáculos. Con esta cualidad añadida no se precisaría ningún engrosamiento ficticio de los obstáculos y se tendría acceso al

conjunto total de soluciones posibles. Esta idea de hibridación de los diagramas de Voronoi con la descomposición en celdas fue la que motivo el método VODEC.

5.4 VODEC con maniobras restringidas

La aplicación del método VODEC como planificador previo seguido del postprocesado con maniobras restringidas puede observarse en la figura 5.17. Como características de este método se han de resaltar la calidad de sus soluciones y su velocidad en comparación con otros en la mayoría de escenarios reales.

Como se explicó en el capítulo 3, VODEC realiza una descomposición vertical en celdas, después las procesa para obtener el diagrama de Voronoi en métrica Manhattan. Este diagrama puede ser total (como el mostrado en la figura 5.17) o parcial, teniendo en cuenta sólo las celdas implicadas en la trayectoria. Finalmente, escoge entre las aristas de Voronoi aquellas que conforman un camino entre las posiciones origen y destino.



a) Resultado de VODEC *b) Postprocesado*

Figura 5.17: VODEC seguido de Postprocesado.

Ha de tenerse en cuenta el tratamiento de los pasos estrechos con este método. No es necesario engrosar los obstáculos con el consiguiente peligro de cegar esos

pasos estrechos. La detección de que un determinado paso es impracticable se puede efectuar en base a la información métrica del diagrama de Voronoi.

Las diferencias de VODEC con otros métodos de cálculo del diagrama de Voronoi, en general estriban en que los otros sólo almacenan la información de equidistancia mínima indispensable mientras que la descomposición en celdas lo hace de forma redundante a través de las funciones de flanco. Por un lado puede elevar el coste computacional, pero por otro constituye una ventaja de cara a cambios en el escenario. VODEC puede asumirlos conteniendo el desplazamiento de obstáculos, o su aparición, dentro de las celdas afectadas, no necesitando calcular de nuevo todo el escenario (ver sección 3.4.2).

5.5 Comparativa entre VODEC y otros planificadores

Una de las dificultades relevantes de los algoritmos de planificación consiste en los denominados "*gaps*" o pasos estrechos. En éstos el espacio de configuraciones se reduce a exiguas sucesiones de posiciones concretas, a las que es muy difícil llegar de modo aleatorio con el RRT. Como ejemplo véase la figura 5.18, donde el escenario sólo pudo ser resuelto con dicho método con tiempos superiores a los dos minutos.

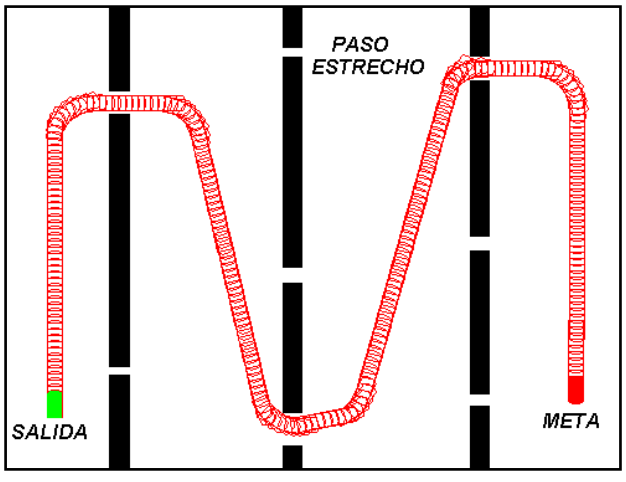


Figura 5.18: Escenario de pasos estrechos.

El algoritmo de Visibilidad implica el ensanche de los obstáculos previo a su aplicación, con lo que los "*gaps"* quedan bloqueados, imposibilitando el hallazgo de

alguna solución. El procedimiento por descomposición en celdas es dependiente de la coordenada utilizada en la descomposición, puesto que la ampliación de obstáculos se hace en el sentido de abscisas. Así, según el escenario, podrán existir estrechos cegados por la ampliación. Aunque mejora este aspecto con respecto al método anterior (López y otros, 2007), pierde cierta eficacia. En el ejemplo de la figura 5.18, este escenario girado 90 grados, tendría todos sus pasos cerrados al ampliar los obstáculos en el sentido de las abscisas. No daría por tanto ninguna solución. En cambio, tal y como se presenta en la figura, sería capaz de ofrecer una trayectoria.

VODEC es especialmente indicado en estos casos dado que, no sólo evita el cierre de los pasos estrechos (no engrosa los obstáculos), sino que además ofrece la trayectoria más alejada de los mismos. Esto es especialmente útil en el postprocesado, donde posiciones muy cerca de los obstáculos aumentan considerablemente la probabilidad de rechazo de las maniobras. Lo cual significa una descomposición de la trayectoria en tramos de búsqueda más pequeños y numerosos, y por tanto una calidad general de la solución obtenida bastante inferior.

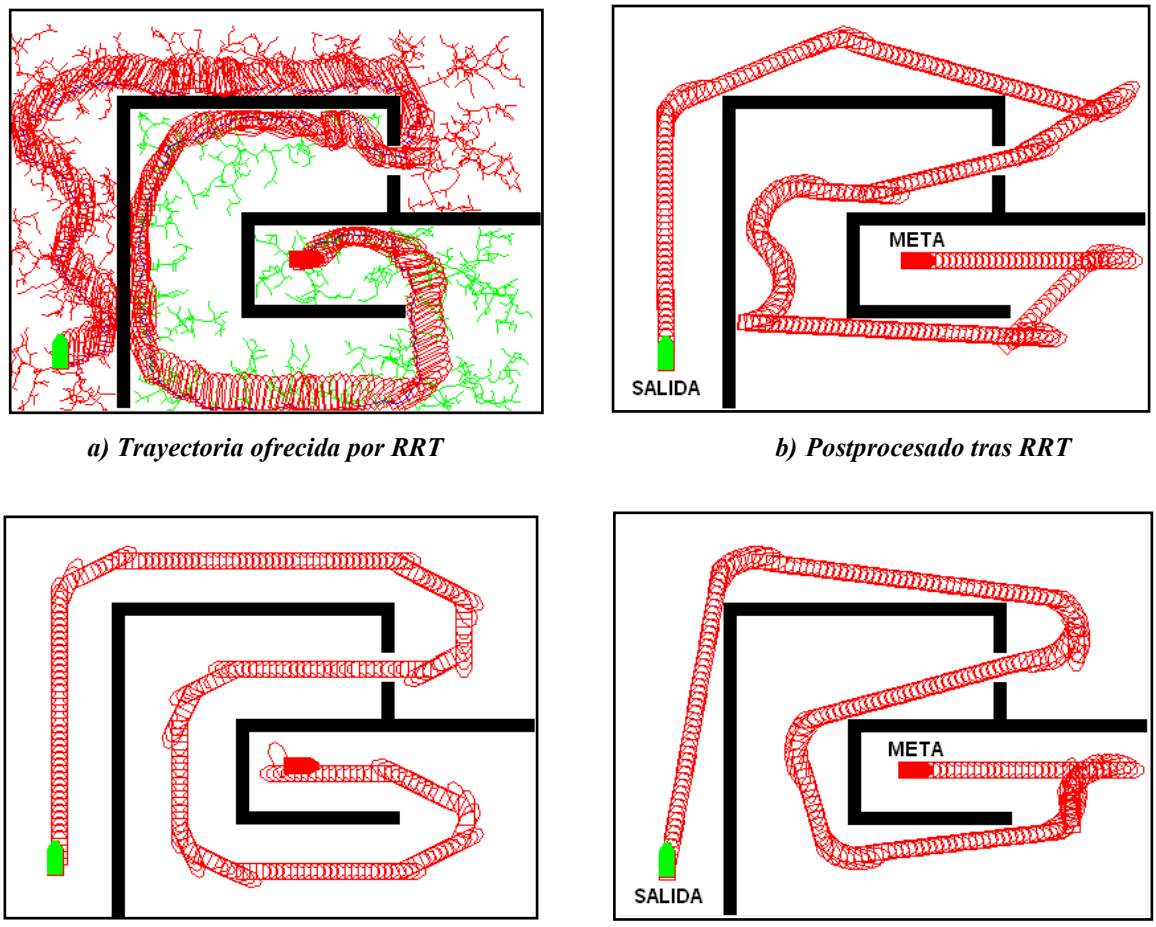
En colación con lo anterior, también se adhiere a esta problemática cualquier planificación que contenga posiciones sucesivas junto a obstáculos, aunque no sean pasos estrechos. Por ejemplo, el algoritmo RRT, en su tentativa de conectar árboles entre sí, los hace ramificarse en profusión junto a los obstáculos, por lo tanto suele ofrecer este tipo de trayectorias.

De modo más acusado ocurre con el algoritmo de visibilidad, donde la propia arista del obstáculo ampliado sirve de segmento de trayectoria. Como muestra de ello se ofrece el resultado del RRT en la figura 5.19a y su postprocesado en la figura 5.19b. En ella se puede apreciar el elevado número de maniobras requeridas.

Sin embargo, en la figura 5.19c, VODEC ofrece para el mismo escenario una línea poligonal claramente distanciada de los obstáculos, es decir, mucho más propicia al postprocesado, que devuelve una solución de calidad considerablemente superior mostrada en la figura 5.19d. El tiempo de este experimento es de 0’294 s. para VODEC, y de 15,525 s. para RRT.

En cuanto a los inconvenientes propios del Voronoi tradicional utilizado como planificador, pueden resaltarse la lentitud y la complejidad. A lo largo de esta sección se han ido ofreciendo los tiempos de los experimentos realizados, todos inferiores al

segundo. Su velocidad comparativa con respecto al RRT es sensiblemente superior en la mayoría de los escenarios.



a) Trayectoria ofrecida por RRT

b) Postprocesado tras RRT

c) Trayectoria ofrecida por VODEC

d) Postprocesado tras VODEC

Figura 5.19: Diferencia de calidad entre soluciones a partir de VODEC y RRT.

Sin embargo, VODEC es dependiente del número de vértices y no aprovecha la virtual cercanía de origen y meta, como sí hace el RRT (ver figura 5.16). VODEC comparte esta característica con el grafo de Visibilidad y la descomposición en celdas, con respecto a los cuales presenta, en las simulaciones efectuadas, velocidades muy similares. No obstante estos escenarios eran muy simples (bajo valor del número de vértices n). En el apartado 3.3.3 se indicó que VODEC oscilaría entre el O($nlogn$) propio de la de la descomposición en celdas y el O(n^2) de los grafos de visibilidad. Sin embargo aún puede mejorarse esta característica como se arguyó en el apartado 3.4.1 (equilibrado selectivo de celdas). En cuanto a la complejidad, al abandonar la métrica

euclídea se eliminan las formas cuadráticas (parábolas que equidistan entre vértices y segmentos), simplificando el resultado. La tabla 5.2 resume las características comparativas.

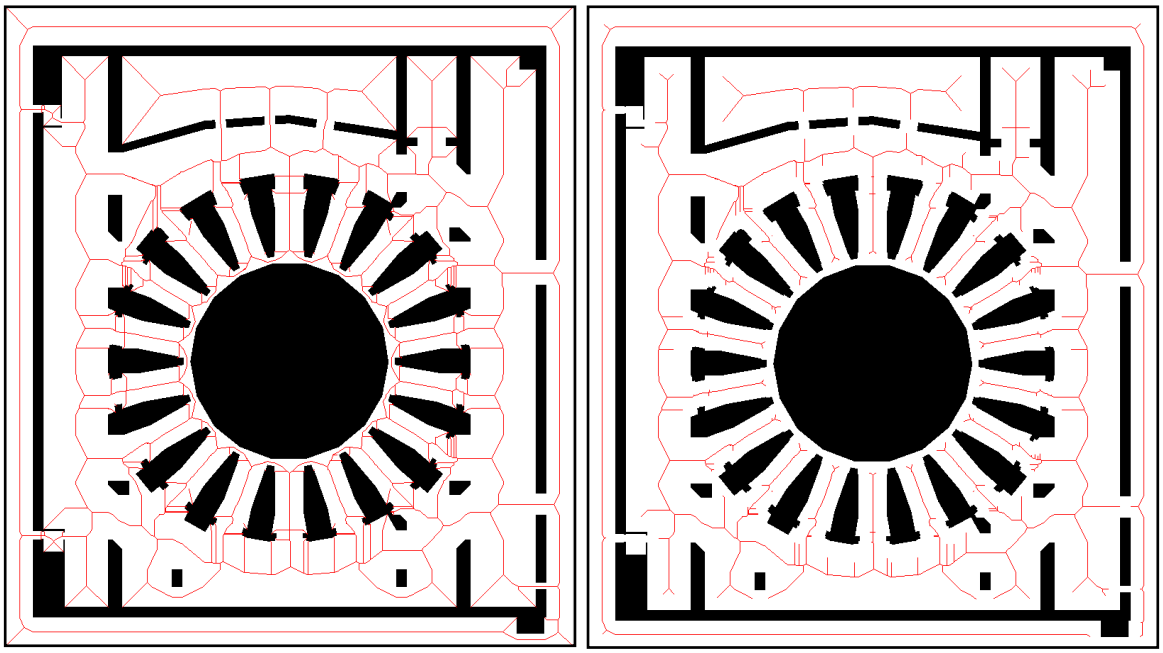
Tabla 5.2: Comparativa entre los diferentes planificadores.

	VODEC	Visibilidad	RRT	D. Celdas
Espacio de soluciones	*Completo*	*Bastante reducido*	*Completo*	*Reducido*
Tiempo de cómputo	*Medio*	*Elevado*	*Indefinido*	*Bajo*
Tratamiento de pasos estrechos	*Óptimo*	*Pésimo*	*Bastante malo*	*Indefinido*
Distanciamiento de los obstáculos	*Máximo*	*Mínimo*	*Suele acercarse*	*Elevado*

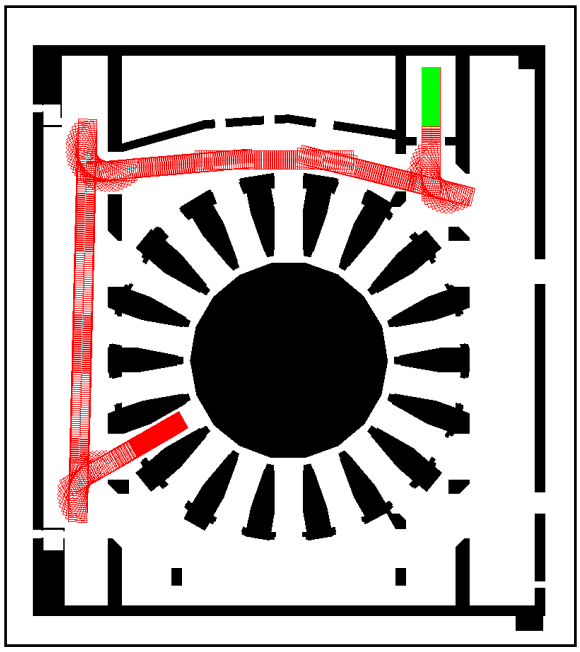
5.6 Aplicación de VODEC en un escenario real

A continuación se presenta la aplicación sobre un escenario real correspondiente al proyecto ITER. En efecto, el escenario planteado en la fig. 5.20 corresponde a una aproximación poligonal a los obstáculos de la planta de trabajo, datos suministrados por el IPFN (Instituto de Plasmas e Fusão Nuclear). En el centro de la figura el polígono mayor corresponde al denominado “tokamak”, es decir, el lugar donde tiene lugar la fusión nuclear. En su perímetro se sitúan unas planchas que han de ser reemplazadas periódicamente debido a su deterioro. Unos robots serán los encargados de realizar esta labor. Su trayectoria se inicia en el ascensor situado en la parte superior derecha de la figura 5.20. El robot debe abandonarlo, circular por el espacio libre perimetral, e introducirse en los huecos que circundan el tokamak para ejecutar la sustitución.

Estos robots levitan sobre un colchón de aire y disponen de dos ruedas (una bajo cada extremo del vehículo) para su guiado. Estas ruedas son totalmente orientables, por lo que el robot no es de tipo Ackerman, sin embargo, dada la magnitud de la carga que portan (varias toneladas de peso), resulta de interés atribuirles trayectorias que las cumplan, dado que éstas afectan en menor medida a la estabilidad.



a) Diagrama de Voronoi completo *b) Subgrafo con restricción de anchura.*



c) Postprocesado del grafo anterior

Figura 5.19: Simulación sobre un escenario complejo real.

La figura 5.20a muestra la planta del ITER y el resultado que devuelve VODEC como diagrama de Voronoi. Obsérvese que en este caso no se ha tomado cada obstáculo como un elemento único, sino como un conjunto de segmentos individuales

cada uno de los cuales tiene su propia región de Voronoi. No obstante se han despreciado aquellas aristas de Voronoi que no parten de oquedades. Esta selección de concavidades y convexidades para atribuir aristas de Voronoi emergentes en sus vértices, resulta práctica y útil a efectos de planificación y surge de forma natural al elegir los máximos de la función de flanco en la fase 3 del algoritmo, es decir, cuando una vez equilibradas las celdas se procede a construir el diagrama de Voronoi.

La fig. 5.20b muestra el resultado de eliminar del grafo anterior aquellos arcos cuya distancia a sus obstáculos más cercanos sea inferior al grosor del robot. Es decir, se eliminan los pasos imposibles, dejando el resto.

Finalmente sobre el grafo de la fig. 5.20b se aplica el algoritmo de búsqueda para obtener una trayectoria. Sobre dicha trayectoria se aplica el postprocesado y su resultado puede observarse en la fig. 5.20c.

5.7 Conclusiones

El algoritmo de postprocesado ha demostrado ser un método eficaz para hallar trayectorias de vehículos con restricciones no holónomas en un escenario con obstáculos. En concreto supera al método RRT adaptado (con las "*bang-motions*"). No obstante necesita un planificador previo sin restricciones. De entre éstos se ha probado con diagramas de Visibilidad, Descomposición en celdas y RRT, comparándolos entre sí. A resultas de esta comparación surge el distanciamiento de los obstáculos como una cualidad deseada para un mejor postprocesado. He aquí donde VODEC supera a los anteriores y se revela como un método ideal para el algoritmo de postprocesado.

CAPÍTULO 6 Evaluación y Optimización

6.1 Introducción

El algoritmo RRT presenta como ventajas reseñables su velocidad y su capacidad de generar distintas soluciones frente a un mismo escenario. Sin embargo, incluso utilizando el postprocesado, sólo algunas de estas soluciones son aceptables y en la mayoría de los casos susceptibles de mejora. Se presenta por tanto la necesidad de seleccionar las mejores soluciones dentro de un conjunto dado. La respuesta obvia sería la utilización de una función de coste. Sin embargo, no es ésta una tarea sencilla, debido fundamentalmente a la dificultad de ponderar adecuadamente parámetros subjetivos como la cercanía de obstáculos, íntimamente relacionada con la tolerancia asumible en el proceso de control. En el apartado siguiente se observa el uso de métodos de decisión multicriterio, más sencillos de implementar desde el punto de vista del experto, en la evaluación de las trayectorias.

También se sugiere la posibilidad de aproximarse a la solución óptima bajo unos determinados criterios. No existe hasta la fecha ningún método exacto para el cálculo de

la trayectoria óptima en vehículos no holónomos. El método heurístico propuesto en el próximo apartado 6.3 aprovecha la capacidad de generación de soluciones diferentes del RRT en combinación con un algoritmo genético común con la intención de buscar este óptimo. Como primer paso se aplica sobre sistemas holónomos, con idea de extender este trabajo en un futuro a los no holónomos.

Por último se ha probado sustituir la función de coste usada en dicho algoritmo genético por el método de decisión multicriterio definido anteriormente. Los resultados se exponen en el apartado 6.4.

6.2 Decisión Multicriterio y RRT

Una de las propiedades observadas en el uso del nuevo método RRT, proviene de su naturaleza aleatoria. Ésta implica la obtención de diferentes soluciones para un mismo problema. El postprocesado, con excepción de escenarios sin obstáculos o muy despejados, raramente se aparta de las zonas por la que discurre la trayectoria recibida del RRT. Es decir, si se tienen trayectorias del RRT que atraviesan áreas diferentes (áreas desvinculadas entre sí por obstáculos), el postprocesado de dichas trayectorias también constituirá soluciones diferentes.

Dichas soluciones no se distinguirán tan solo por las zonas de paso, sino que también lo harán por la longitud recorrida, y otros parámetros. Se presenta entonces el problema de elegir la trayectoria óptima. A este respecto lo primero que suele considerarse es la longitud, ya que de ésta depende el tiempo empleado en llegar a la meta y el consumo de energía del robot. Variables muy a tener en cuenta en problemas de optimización, pero no absolutamente relevantes como se mostrará a continuación.

Un conductor humano suele escoger a veces un camino más largo con tal de evitar trayectorias largas marcha atrás, muchos cambios de sentido, o continuos giros de volante. Esto se debe al mayor esfuerzo de control y a la menor calidad del mismo que suponen las maniobras que se desean evitar. Así pues, no resulta obvio elegir la trayectoria óptima. Es aquí donde resulta más apropiada la aplicación de algoritmos multicriterio. Se expresa a continuación la lista de criterios utilizada.

6.2.1 Criterios de estudio

Dadas varias trayectorias planificadas para un vehículo en un entorno determinado, se plantea la cuestión de hallar cuál es la mejor. Para ello se dispone de varios criterios de estudio de diversa ponderación, aplicables a cada una de dichas trayectorias (Gómez Bravo y otros, 2006). Ha de tenerse en cuenta que el formato con el que éstas se procesan consiste en una sucesión de posiciones del vehículo desde la ubicación inicial a la final, muy próximas entre sí. La intención es aproximar mediante una representación discreta los infinitos puntos que atraviesa el vehículo en el espacio de configuraciones al moverse. Entre una posición y otra existe una distancia euclídea fija, que se denotará por *Dmin*, y que no es más que el paso de discretización.

A continuación se exponen los distintos criterios. Si bien el primero de ellos es el más obvio, existen otros parámetros que incluso un controlador humano tiene en cuenta a la hora de elegir la planificación más adecuada. He aquí la relación:

Longitud: Es el criterio comparativo más inmediato: la longitud total recorrida. Se podría calcular multiplicando *Dmin* por el número de posiciones que componen el camino. Sin embargo, dadas las imperfecciones debidas a la resolución del problema (también el entorno viene dado en forma de matriz discreta), resulta una estimación más aproximada utilizar el cálculo de la distancia euclídea entre dos posiciones consecutivas y agregar el resultado en una suma global:

$$Longitud = \sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \qquad (6\text{-}1)$$

Donde *n* es el número de posiciones del vehículo en la trayectoria y (*x,y*) son las coordenadas cartesianas de cada posición.

Número de cambios de sentido: Es el número de veces que el vehículo varía su sentido de marcha, hacia atrás o hacia delante. Este parámetro influirá notablemente en el consumo del vehículo, ya que cada cambio de sentido implica una parada total, y por tanto elimina toda la energía cinética del mismo haciendo necesario un nuevo impulso para reiniciar el movimiento. Además el esfuerzo de control también se ve afectado (frenado, cambio de marcha, embrague, aceleración, etc.), por no hablar de los errores de deslizamiento que se generan en la localización por odometría en un robot autónomo.

Un ejemplo habitual en el que existen muchos cambios de sentido es la salida de un aparcamiento subterráneo, donde el margen de maniobra es bastante escaso. En dichas situaciones, el conductor procura apurar al máximo el recorrido factible del vehículo en cada maniobra, minimizando el número de veces que ha de cambiar el sentido de la marcha. En las siguientes simulaciones (fig. 6.5) puede verse cómo se prefiere aumentar la longitud total de la maniobra en pro de un menor número de cambios de sentido.

El cálculo de este parámetro es sencillo, basta con almacenar en una variable el sentido del vehículo y constatar cuando se altera (ver figura 6.1).

```
Para i = 1 hasta n
        sentido_nuevo = (Δx_i>0)
        Si (sentido_nuevo≠sentido_anterior) Entonces
                incrementa NCS
        sentido_anterior =sentido_nuevo
Siguiente i
```

Figura 6.1: Algoritmo de cálculo del número de cambios de sentido.

Donde Δx_i es el incremento de *x* en coordenadas locales y *NCS* es el número de cambios de sentido.

Longitud marcha atrás: indica la distancia recorrida en esta situación. Aquí es el control el que resulta más dificultoso. Esto es obvio en conductores humanos, pero también afecta a la navegación de robots según sea su diseño. Así, es habitual disponer más sensores en la parte frontal que en la trasera. También las prestaciones motrices pueden diferir según la marcha utilizada, siendo habitual varias relaciones de potencia hacia adelante, y sólo una hacia atrás. Incluso a veces la inversión del sentido de giro en motores de explosión se implementa mediante cambios en la sincronización del encendido, provocando más ruido y mayor deterioro del motor. Éstos y otros detalles pueden aumentar el peso de este criterio.

Se halla por el mismo procedimiento que la longitud total, sólo que ahora sólo se sumarán las distancias cuando el sentido de la marcha sea hacia atrás.

$$Longitud\_marcha\_atrás = \sum_{i=2}^{n} (\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}) \cdot a \qquad (6\text{-}2)$$

Donde *a* es cero si el vehículo avanza y 1 si retrocede.

Separación media: indica la proximidad de la trayectoria a los obstáculos del entorno. Se calcula determinando para cada paso del camino la distancia al obstáculo más próximo. Después se halla la media de estos valores y se obtiene un índice global de dicha proximidad. No es necesario conocer si un vértice u otro del vehículo resulta más próximo, es decir, no interesa ser exacto en el cálculo de dicha distancia. Tan sólo resulta relevante de un modo aproximado. Así, un algoritmo genera rectángulos crecientes centrados en la posición y con la orientación del vehículo. En cuanto alguno solape con cualquier obstáculo, se detiene el proceso anotando el ensanche máximo logrado (R_{max}). En la figura 6.2 puede observarse el rectángulo logrado en cuatro ubicaciones diferentes. En las marcadas con las letras 'a' y 'c', se muestra el segmento cuyo valor es utilizado. La media de estos valores R_{max} a lo largo de toda la trayectoria será la separación media.

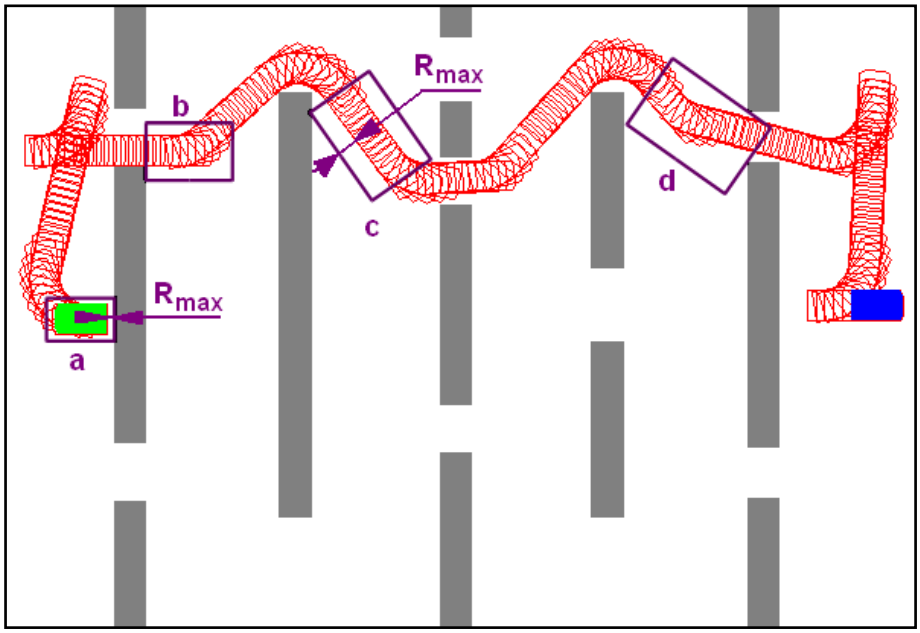


Figura 6.2: Cálculo de la separación media.

Es obvio que los conductores prefieren evitar trayectorias que discurran por lugares angostos y con obstáculos muy próximos, pues ello los obliga a circular con menor velocidad y mayor atención.

En vehículos autónomos los pasos estrechos suponen un mayor riesgo debido a las tolerancias de los sensores, los errores de localización y las imprecisiones inherentes al planificador (resolución). Además se dispone de menor flexibilidad ante eventuales cambios del entorno, dado que se limita el número de maniobras disponibles y éstas suelen proporcionar resultados de baja calidad (recuérdese el caso de aparcamiento subterráneo comentado con anterioridad).

$$Separación\_media = \frac{\sum_{i=1}^{i=n} R\max_i}{n} \qquad (6\text{-}3)$$

Índice de suavidad. Entre dos posiciones consecutivas del camino se halla la diferencia angular, al resultado se le aplica el valor absoluto y se suma a los anteriores, obteniéndose al final este parámetro.

$$suavidad = \sum_{i=2}^{n} \left| \theta_i - \theta_{i-1} \right| \qquad (6\text{-}4)$$

Donde θ_i es la orientación del vehículo en la posición *i*-ésima.

Puede inferirse que las maniobras rectas no incrementan el índice, sólo lo hacen las maniobras curvas. Tanto más cuanto más cerrado sea el radio de giro, y especialmente en cambios bruscos del mismo. Así, con este valor se obtiene una apreciación del esfuerzo de control en la dirección, es decir, la dificultad de manejo del volante que entrañan las maniobras a desarrollar, además de las limitaciones de velocidad asociadas a los giros, mayores cuanto más cerrados sean.

Si lo que se pretende es medir el esfuerzo de control de dirección, en el caso más simple, éste será proporcional a dicho ángulo, sin considerar su signo. Por tanto la función de coste para una trayectoria dada será proporcional a la integral a lo largo de toda la trayectoria y en el tiempo, del valor absoluto del ángulo de dirección. Si se asume que el vehículo se desplaza a velocidad constante, el ángulo de dirección provocará un incremento constante de la orientación absoluta del vehículo. Es decir:

$$\alpha = \frac{d\theta}{dt} \qquad Función\_de\_\cos te = k\int |\alpha| dt = k\int \left|\frac{d\theta}{dt}\right| dt \tag{6-5}$$

Siendo α el ángulo de dirección y θ el de la posición. Trasladando el resultado al ámbito discreto:

$$Función\_de\_\cos te = k\sum_{i=2}^{n} \frac{|\theta_i - \theta_{i-1}|}{\Delta t}\Delta t = k\sum_{i=2}^{n} |\theta_i - \theta_{i-1}| = k \cdot suavidad \tag{6-6}$$

Donde *k* es una constante de proporcionalidad. Su valor reflejará la relación entre los incrementos angulares y el coste o esfuerzo utilizado para realizarlos.

Dependiendo del sistema, podrá ocurrir que mantener una dirección determinada no suponga ningún esfuerzo, mientras que los cambios sí. Éste y otros supuestos podrán generar diferentes funciones de coste y requerir por tanto implementaciones más o menos complejas. En cualquier caso se aprecia la relevancia de este índice.

6.2.2 Algoritmo de decisión

Una vez establecidos los criterios que permiten ponderar la mayor o menor conveniencia de las trayectorias obtenidas es necesario establecer un procedimiento sencillo y rápido por el cual se determine, o bien cual trayectoria es la mejor, o bien una clasificación de las mismas de mejor a peor para realizar posibles descartes.

Este tipo de procedimiento no es más que una Toma de Decisiones Multicriterio (TDM). Existen en la literatura científica infinidad de métodos de decisión multicriterio aplicados a numerosas áreas de conocimiento y problemas de decisión. De los abundantes métodos existentes, encontrar uno que resulte sencillo y rápido no es fácil. Para ello, los distintos métodos de decisión se han clasificado en las siguientes familias (Barba-Romero y Pomerol, 2000):

-Métodos Multicriterio Continuos. Son aquellos cuyo conjunto de posibles soluciones es infinito al problema.

-Métodos Multicriterio Discretos. Son aquellos cuyo conjunto de posibles soluciones es finito numerable.

-**Métodos Multicriterio Interactivos**. Son aquellos métodos que para obtener la solución al problema, necesitan interaccionar normalmente con una persona o sistema que pueda, en cada momento, proporcionarle información adicional que le permita ir descartando soluciones no validas.

-Teoría de la Utilidad Multiatributo. Son aquellos métodos que se basan no en el valor de cada criterio sino en la utilidad que éste tiene para la obtención de una solución al problema.

-**Métodos Multicriterio Difusos**. Son aquellos cuya información es parcial o bien es de naturaleza cualitativa (Martin y otros, 2001).

De todas las familias enumeradas anteriormente, se escoge la familia de métodos multicriterio Discretos por los siguientes motivos:

1. Se parte de un conjunto finito de posibles trayectorias, por lo que los métodos multicriterio continuos no son aplicables.

2. Se necesita un método automático para clasificar las distintas trayectorias, con lo cual, los métodos multicriterio interactivos no son aplicables.

3. Los métodos relacionados con la utilidad multiatributo, tampoco son aplicables ya que es difícil definir y obtener los mecanismos para que, de manera automática, el método ofrezca la utilidad de la trayectoria en función de de los valores de los criterios.

4. Los métodos multicriterio Difusos, si podrían ser aplicables, siempre y cuando los criterios para la selección de las trayectorias estén definidos a través de conjuntos borrosos (Martin y otros, 2001).

La familia de métodos multicriterio discretos es muy extensa. De entre éstos parecen idóneos los de las subfamilias Electre (Bernard, 1996; Yves y otros, 1994) y Promethee (Brans y Vincke, 1985; Vincke, 1992), ampliamente utilizadas, y que son representantes de la Escuela Europea de Toma de Decisiones.

Estas dos familias de métodos, aún no teniendo una base matemática tan sólida como otros y siendo métodos heurísticos, se han comprobado en innumerables problemas de toma de decisiones en diferentes ámbitos entre los que se pueden citar económicos, logísticos y medioambientales, obteniendo excelentes resultados en cada uno de ellos (Geldermann y otros, 2000). Además poseen las ventajas de ser muy

conocidos, ampliamente utilizados, con metodologías muy comprensibles, rápidos y fácilmente programables.

La mayoría de los métodos TDM construyen la "*matriz de decisión"* que se forma a partir de un conjunto de alternativas $A=\{A_1,\ldots,A_i,\ldots,A_m\}$ (trayectorias), un conjunto de criterios de decisión $C=\{C_1,\ldots,C_j,\ldots,C_n\}$ (en este caso corresponderían a la longitud total, número de cambios de sentido, etc.), un conjunto de pesos $w=\{w_1,\ldots,w_j,\ldots,w_n\}$ que cuantifican la importancia de cada criterio en la obtención de la solución final de la toma de decisiones (TD) y finalmente un conjunto de funciones o umbrales propios y característicos de cada método de ayuda a la decisión (Vincke, 1992). Cada elemento $a_{i,j}$ de la matriz de decisión refleja el valor que tiene la alternativa *i*-ésima (A_i) con respecto al criterio j (C_j).

De la familia de métodos de TDM Electre y Promethee se han seleccionado las primeras variantes, es decir, los métodos Electre I, II y Promethee I, II.

Las familias de métodos Electre y Promethee también son clasificados dentro de las familias de métodos relacionados con la sobreclasificación o superación. Los métodos de sobreclasificación son métodos que comparan, dos a dos, todas las alternativas o posibles soluciones (en este caso trayectorias) y determinan según una relación binaria de orden (denominada relación de superación o sobreclasificación $\mathcal{S}$), previamente construida, cuál o cuáles son las mejores alternativas a un problema de TDM dado.

Todos los métodos de sobreclasificación se pueden diferenciar en numerosos aspectos de su aplicación, pero verifican que su metodología se realiza en dos fases claramente diferenciadas.

A) Fase de construcción de la relación de sobreclasificación $\mathcal{S}$.

B) Fase de explotación de esta relación de sobreclasificación.

En los siguientes apartados se presenta para cada método su fase de construcción y su fase de explotación.

6.2.2.1. El método Electre I.

El método Electre I (Bernard, 1996), tiene como resultado la o las mejores alternativas que son solución al problema de TMD. Este método define y construye la relación de sobreclasificación de la siguiente declaración:

"Dadas dos alternativas A_i y A_k, elementos del conjunto A de alternativas, la relación de sobreclasificación S es una relación binaria definida en A tal que $A_i S A_k$ (A_i supera a A_k), si conocidas las preferencias del decisor, la calidad de la evaluación de las alternativas y la naturaleza del problema, hay suficientes argumentos para decir que A_i es al menos tan buena como A_k, mientras que no existan razones esenciales para rechazar esta declaración".

Esta definición es más una idea que una definición matemática precisa. La manera de construir la relación binaria S es lo que diferencia los distintos métodos relacionados con la sobreclasificación.

A) Fase de construcción

Los métodos Electre construyen la relación de sobreclasificación S a partir de dos matrices denominadas:

• Matriz de Concordancia *c*.

• Matriz de Discordancias *d*.

Estas dos matrices reflejan el concepto de sobreclasificación definido por Bernal Roy (Bernard, 1996), es decir, la matriz de concordancia c, cuantifica la cantidad de criterios para los cuales se puede decir que una alternativa A_i es mejor que otra A_k, y la matriz de discordancia cuantifican las razones para decir lo contrario aunque sea en al menos un criterio, es decir, indica cuánto es de inferior una alternativa A_i con respecto a otra alternativa A_k. Ambas matrices de dimensión $m \times m$ se construyen a partir de la matriz de decisión del siguiente modo:

$\forall A_i, A_k \in A$ se define la matriz de concordancia c como $c : A \times A \to [0,1]$, donde:

$$c(A_i, A_k) = c_{i,k} = \frac{\sum_{j:g_j(A_i) \geq g_j(A_k)} w_j}{\sum_{j=1}^{n} w_j} \qquad (6\text{-}7)$$

$\forall A_i, A_k \in A$ se define la matriz de discordancia d como $d : A \times A \rightarrow [0,1]$, donde:

$$d(A_i, A_k) = d_{i,k} = \begin{cases} 0 & g_j(A_i) \geq g_j(A_k), \forall\, j \\ \dfrac{\operatorname{Max}_j [g_j(A_k) - g_j(A_i)]}{\operatorname{Max}_{i,k} [g_j(A_k) - g_j(A_i)]} & g_j(A_i) < g_j(A_k), \forall\, j \end{cases} \qquad (6\text{-}8)$$

Con $g_j(A_i) = a_{i,j}$

Así pues, $\forall A_i, A_k \in A$, la relación de sobreclasificación S se define como:

$$\textit{la alternativa } A_i \textit{ sobreclasifica a la alternativa } A_k\ (A_i S A_k) \Leftrightarrow \begin{cases} c_{i,k} \geq \hat{c} \\ d_{i,k} \leq \hat{d} \end{cases} \qquad (6\text{-}9)$$

donde $\hat{c}$ y $\hat{d}$ son dos umbrales denominados respectivamente de concordancia y de discordancia que cuantifican las razones para poder decir cuando una alternativa sobreclasifica a otra. Normalmente $\hat{c}$ es un umbral relativamente elevado, y $\hat{d}$ es un umbral relativamente bajo.

B) Fase de explotación

La fase de explotación, es aquella en la que se utiliza la relación de sobreclasificación S calculada en la fase de construcción, para determinar cuál o cuáles son las mejores alternativas al problema de TDM dado o bien una ordenación de mejor a peor alternativa.

A partir de la relación de sobreclasificación S, el método Electre I, la utiliza para obtener lo que denomina el núcleo o kernel de un grafo de sobreclasificación N. Este grafo dirigido asocia sus vértices a cada una de las alternativas $A_i \in A$. Dados dos vértices v_i y v_k del grafo, existirá una arista dirigida de v_i a v_k ($v_i \rightarrow v_j$), si la alternativa A_i sobreclasifica a la alternativa A_k, $A_i S A_k$.

Una vez definido el grafo, para obtener el kernel del mismo N, se determina el conjunto de alternativas que superan o sobreclasifican al resto y entre ellas no exista ninguna combinación de sobreclasificación, es decir son incomparables:

$$\begin{cases} \forall\ A_k \in A - N, \quad \exists\ A_i \in N : A_i \succsim A_k, \\ \forall\ A_i, A_k \in N,\ A_i \not\succsim A_k \end{cases} \qquad \textit{(6-10)}$$

6.2.2.2.- El método Electre II.

El método Electre II obtiene como resultado la ordenación de las distintas alternativas presentes en el problema de TDM. Este resultado es mucho más rico para la selección de trayectorias ya que ofrece la posibilidad de elegir no solo cual sería la/s mejores alternativas sino las siguientes en orden de importancia para el caso que las mejores no sean de aplicación por cualquier motivo.

El desarrollo del método es muy parecido al método Electre I por ser de la misma familia, es decir, construye de manera idéntica las matrices de concordancia c y discordancia d a partir de la matriz de decisión inicial.

La diferencia estriba en que no construye una única relación de sobreclasificación, sino que construye dos, una denominada relación de sobreclasificación fuerte (S^F) y otra débil (S^f). La relación de sobreclasificación fuerte es mucho más restrictiva por lo que el conjunto de alternativas de kernel es mucho menor y al contrario para la relación de sobreclasificación débil. La utilización de ambas relaciones permite al método obtener una relación de orden de todas las alternativas del problema.

A) Fase de construcción

A continuación tiene lugar la fase de construcción de las relaciones de sobreclasificación fuerte (S^F) y débil (S^f). Las relaciones de sobreclasificación fuerte y débil se construyen a partir de las matrices de concordancia c y discordancia d vistas en el método Electre I, (ecuaciones (6-7) y (6-8)). Las dos relaciones de sobreclasificación S^F y S^f cuantifican, a distinto nivel de incertidumbre, la veracidad de

la hipótesis de sobreclasificación dada en la declaración por Bernal (1996). Así pues, las relaciones de sobreclasificación fuerte y débil se definen como:

$\forall A_i, A_k \in A$

$$A_i S^F A_k \Leftrightarrow \begin{cases} c_{i,k} \geq \hat{c}_F \\ \sum_{j:g_j(A_i)>g_j(A_k)} w_j \geq \sum_{j:g_j(A_i)<g_j(A_k)} w_j \\ d_{i,k} \leq \hat{d}_F \end{cases} \quad \text{(6-11)}$$

$$A_i S^f A_k \Leftrightarrow \begin{cases} c_{i,k} \geq \hat{c}_f \\ \sum_{j:g_j(A_i)>g_j(A_k)} w_j \geq \sum_{j:g_j(A_i)<g_j(A_k)} w_j \\ d_{i,k} \leq \hat{d}_f \end{cases} \quad \text{(6-12)}$$

donde $\hat{c}_F, \hat{d}_F, \hat{c}_f$ y $\hat{d}_f$ son los umbrales de concordancia y discordancia fuerte y débil respectivamente verificando que $\hat{c}_F > \hat{c}_f$ y $\hat{d}_F < \hat{d}_f$.

B) Fase de explotación

A partir de estas relaciones de sobreclasificación, el método Electre II obtiene dos relaciones de preorden total de las alternativas, mediante un procedimiento que se resume en los siguientes pasos:

1. Se utiliza la relación de sobreclasificación fuerte S^F para obtener el conjunto B de las alternativas que no son fuertemente sobreclasificadas por cualquier otra alternativa.

2. Se utiliza la relación de sobreclasificación débil S^f con el conjunto previamente obtenido para obtener el conjunto A_1 de alternativas que no son débilmente sobreclasificadas por otra alternativa de B. El conjunto A_1 es la primera clase de alternativas dentro del preorden total de las alternativas.

3. El procedimiento se reitera con el conjunto de alternativas restantes ($A - A_1$) obteniendo sucesivos conjuntos A_i de alternativas hasta que no queden más por ordenar.

El procedimiento anterior nos proporciona un preorden descendente de todas las alternativas del problema de decisión. Un segundo preorden es construido de manera análoga, pero comenzando con la clase de las peores alternativas, es decir, aquéllas que no sobreclasifican a ninguna alternativa y continuando hasta llegar a las mejores alternativas.

Los preordenes obtenidos por lo general no son los mismos, con lo cual se ofrece al decisor un preorden parcial construido a partir de los dos preordenes totales, mediante el siguiente procedimiento:

1. Si la alternativa A_i es preferida a la alternativa A_k en ambos preordenes totales, entonces también será preferida para el preorden parcial final.

2. Si la alternativa A_i es equivalente a la alternativa A_k en un preorden y preferida en el otro, entonces también será preferida para el preorden parcial final.

3. Si la alternativa A_i es preferida a la alternativa A_k en un preorden y al revés en el otro preorden total, entonces ambas alternativas serán incomparables en el preorden parcial final.

En el preorden parcial obtenido, es posible que existan muchas alternativas que son incomparables. Este resultado no es producto de una inconsistencia del método, sino que es una forma de detección de problemas entre alternativas, debido a que aparecen en distintos lugares dentro de los preordenes totales.

6.2.2.3 Métodos Promethee

De la familia de métodos TDM se han seleccionado los métodos Promethee I y II (Brans y Vincke, 1985), por modelar las preferencias humanas lo que hace que sus resultados sean más realistas para el ser humano.

Los métodos Promethee (Barba-Romero y Pomerol, 2000) construyen la matriz de sobreclasificación mediante la relación valuada $\pi(A_i, A_k) \in [0,1]$ que representa la credibilidad de la sentencia "*la alternativa A_i sobreclasifica o supera a la alternativa A_k*" de manera que a mayor valor, mayor es la certeza o credibilidad de la anterior expresión. Esta relación es definida como:

$$\pi(A_i, A_k) = \pi_{i,k} = \frac{\sum_{j=1}^{n} w_j \cdot F_j(A_i, A_k)}{\sum_{j=1}^{n} w_j} \tag{6-13}$$

Donde F_j modela las preferencias que una persona tendría si tuviera que comparar ambas alternativas desde el punto de vista del criterio *j*. Existe un conjunto de funciones de preferencias predefinidas estándar aunque el método no impone restricciones para definir nuevas funciones de preferencias propias. En la figura 6.3 se puede observar el conjunto de funciones de preferencias humanas que definen la familia de métodos Promethee.

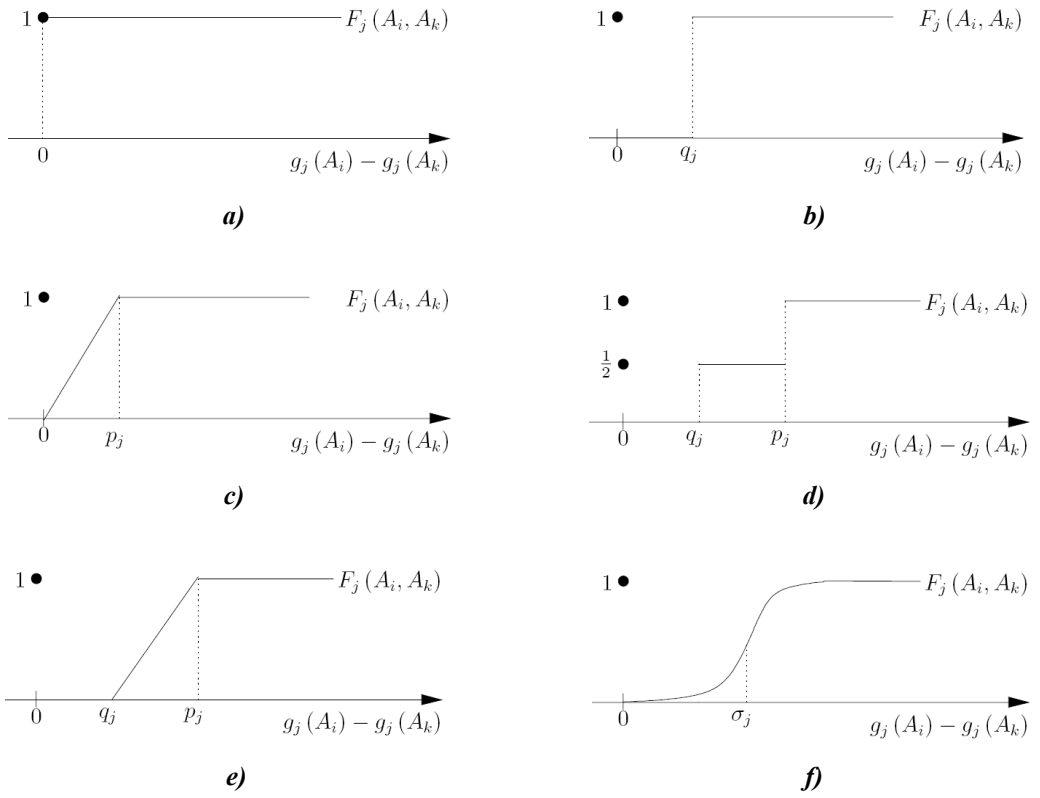


Figura 6.3: Funciones de preferencia más comunes, (a) de preferencia inmediata, (b) de estricta preferencia con un umbral de indiferencia, (c) de preferencia lineal con umbral de preferencia, (d) de preferencia escalonada con umbrales de preferencia e indiferencia, (e) de preferencia lineal entre umbrales de indiferencia y preferencia y (f) de preferencia con distribución normal..

Este conjunto de funciones es suficiente para modelar las preferencias humanas en cualquier problema real de TDM, pero los métodos Promethee no restringen este conjunto pudiendo utilizar cualquier función de preferencias que se considere adecuado. Los valores q_j y p_j definidos en algunas funciones de preferencias, son respectivamente

los umbrales de indiferencia y preferencia estricta respectivamente. Indican en el primer caso el valor a partir del cual las alternativas dejan de ser indiferentes, y en el segundo el valor a partir del cual se tiene una preferencia absoluta sobre una de ellas. Las funciones de preferencia F_j comparan ambas alternativas, restando los valores que tienen en la matriz de decisión para el criterio *j* y devolviendo un valor real perteneciente al intervalo unidad que modela la preferencia. Dicha preferencia sería la que experimentaría un ser humano si la diferencia entre ambas alternativas (para un criterio *j* determinado) aumentase desde cero hasta un valor infinito, es decir, su preferencia creciente o inmediata por la alternativa A_i en detrimento de la alternativa A_k. Un valor 0 para F_j expresa la indiferencia por ambas alternativas. El valor unidad significa una preferencia absoluta por la alternativa A_i.

A partir de la matriz de sobreclasificación, se definen, según el método Promethee, un conjunto de funciones denominadas *flujos* que nos ayudan a determinar las mejores trayectorias. Estos flujos son:

Flujo de Entrada para una alternativa A_i, permite determinar cuántas alternativas A_k superan o sobreclasifican la alternativa A_i.

$$\phi^{-}(A_i) = \sum_k \pi(A_k, A_i) \quad \forall A_k \in A \tag{6-14}$$

Flujo de Salida para una alternativa A_i, permite determinar cuántas alternativas A_k son superadas por la alternativa A_i.

$$\phi^{+}(A_i) = \sum_k \pi(A_i, A_k) \quad \forall A_k \in A \tag{6-15}$$

Flujo de Neto para una alternativa A_i, permite determinar si la alternativa supera más o es más superada por el resto de alternativas.

$$\phi(A_i) = \phi^{+}(A_i) - \phi^{-}(A_i) \quad \forall A_i \in A \tag{6-16}$$

Mediante los distintos flujos podemos determinar si una trayectoria es mejor que otra si el número de trayectorias a la cuales supera es mayor que el número de trayectorias que superan a la primera.

Todos los métodos de TD relacionados con la sobreclasificación definen una relación de superación $\mathcal{S}(A_i,A_k)$ que determina para dos alternativas cuál de ellas es preferida. Como consecuencia, a partir de esta relación $\mathcal{S}$ se puede determinar cuál o cuáles son las mejores alternativas y/o una ordenación de las mismas.

En concreto el método Promethee I define la relación $\mathfrak{S}$ como:

$$S(A_i, A_k) = \phi^+(A_i) \geq \phi^+(A_k) \wedge \phi^-(A_i) \leq \phi^-(A_k) \tag{6-17}$$

Y el Promethee II como:

$$S(A_i, A_k) = \phi(A_i) \geq \phi(A_k) \quad \forall A_i, A_k \in A \tag{6-18}$$

A partir de la relación de superación construida para el problema siguiente, se han obtenido resultados acordes con las decisiones que tomaría un ser humano.

6.2.3 Resultados

En la figura 6.4 se exponen seis alternativas generadas con algoritmo RRT seguido de postprocesado con maniobras restringidas en un mismo escenario con múltiples puntos de paso alternativos y con las mismas posiciones inicial y final. Como se puede observar, las soluciones difieren entre sí afectando a todos los criterios mencionados. La tabla 6.1 recoge los valores obtenidos para cada criterio.

Tabla 6.1: Valores de los criterios de estudio para cada trayectoria

	Longitud	Longitud marcha atrás	Inversores	Suavidad	Separación media
Trayectoria 1	1226.64	94.12	4	122.72	15
Trayectoria 2	940.60	37.74	2	52.63	13
Trayectoria 3	1153.50	94.11	4	113.36	15
Trayectoria 4	1043.31	87.54	4	49.06	14
Trayectoria 5	1360.8	540.95	10	82.76	15
Trayectoria 6	872.33	3.26	1	49.53	14

Para obtener una clasificación, se establecieron las funciones de preferencia utilizando la representada en la figura 6.3c. Los umbrales de preferencia se definieron de modo intuitivo, decidiendo a partir de lo que un conductor haría. Seguidamente se aplicó el método TDM en sus variantes Electre I y II. La figura 6.5 recoge las clasificaciones obtenidas tras procesar la tabla 6.1 con dichos métodos.

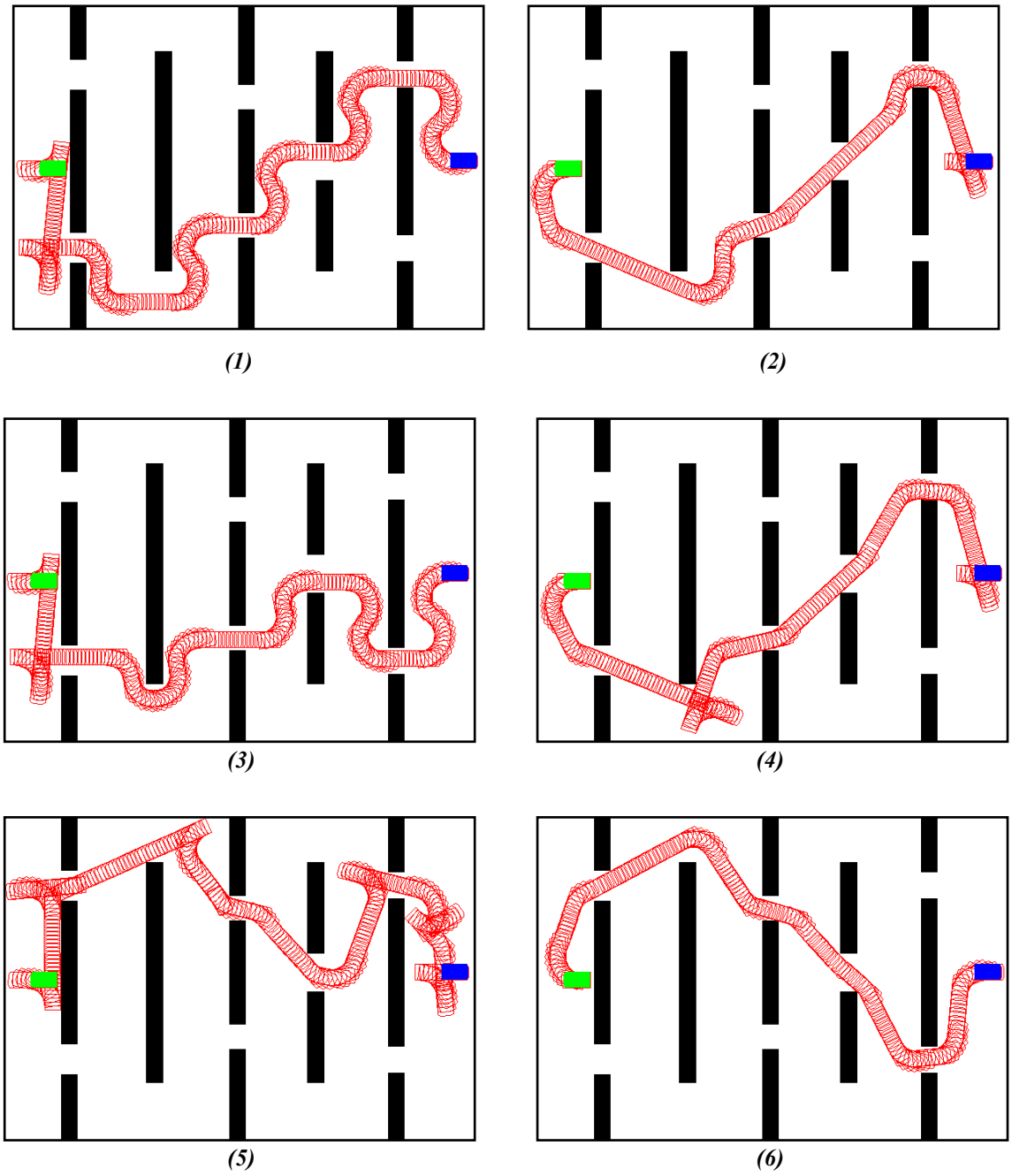


Figura 6.4: Alternativas para un mismo escenario.

Observando la preferencia de alternativas calculada y comparándola con la clasificación que de forma intuitiva un conductor humano elegiría, podemos apreciar una solución bastante satisfactoria. Se corrobora por tanto la propiedad del método de ser capaz de trasladar las preferencias humanas y su idoneidad para jerarquizar las diferentes soluciones que aporte el planificador.

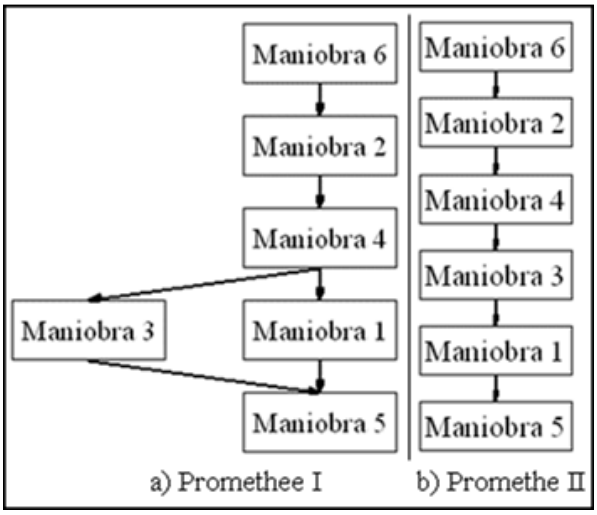


Figura 6.5: Resultados del algoritmo de decisión

6.2.4 Aplicación

Entre los objetivos de esta tesis está la aplicación a escenarios reales. Se ha escogido el entorno del hospital Juan Ramón Jiménez (fig. 6.7) y un robot de servicio de reparto real como sujeto de las planificaciones (figura 6.6). Dicho robot se sitúa bajo los carritos de reparto, cuyas medidas están normalizadas; se adhiere mecánicamente y a continuación puede desplazarse con él al lugar programado.

Figura 6.6: Plataforma autónoma para la automatización de reparto en entornos hospitalarios.

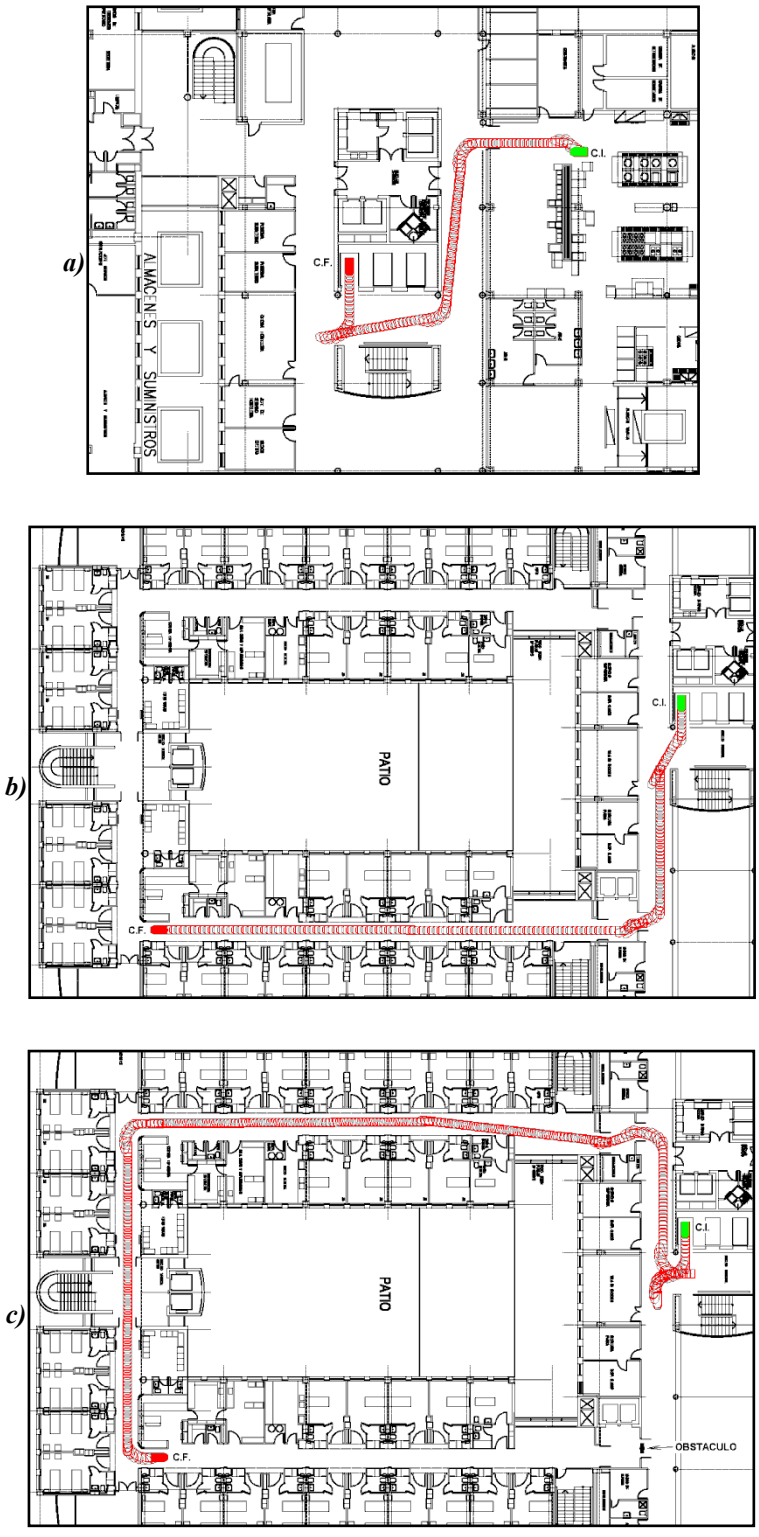


Figura 6.7: Trayectorias planificadas para el traslado del menú: a) salida del almacén hasta los ascensores; b) camino desde los ascensores hasta el punto de reparto; c) ruta alternativa por oclusión de la ruta principal.

Hay que destacar la escasez de espacio en este entorno concreto, donde prácticamente la totalidad del recorrido se realiza en pasillos estrechos, lo cual ralentiza considerablemente el algoritmo RRT.

En la figura 6.7 se muestra el resultado del planificador para tres casos diferentes. En el primero (fig. 6.7a) se muestra la trayectoria obtenida para el acceso del robot desde las cocinas situadas en la planta inferior a los ascensores. En el segundo (fig. 6.7b) se traslada el robot desde los ascensores hasta un punto de reparto al que acuden las enfermeras para distribuir el menú por las habitaciones.

La última (fig. 6.7c) muestra la respuesta de la aplicación cuando existe un obstáculo que impide el regreso por el camino más corto.

La planificación en los escenarios de las figuras explicadas con anterioridad ofrece escasas alternativas. No hay variedad de soluciones entre las que elegir. Todo lo contrario de lo que ocurre en la figura 6.8. En este caso se requiere un desplazamiento del robot en una sala de paso. En todas las soluciones mostradas en la figura la posición de partida está marcada en verde, y la de destino en rojo.

El algoritmo RRT seguido del postprocesado ha proporcionado siete formas diferentes de realizar este traslado. Algunas valorables a simple vista como excesivamente intrincadas, con innecesarios cambios de sentido o recorridos inútilmente largos. Otras parecen más lógicas o incluso cercanas a la optimalidad. Nuevamente se plantea la aplicación de los métodos TDM para clasificarlas.

Antes de la aplicación de los distintos métodos de TDM, se deben analizar para cada criterio los objetivos (maximizar o minimizar) así como los pesos que determinarán cuantitativamente la importancia que tiene cada criterio en la valoración global de cada alternativa (trayectoria). En la tabla 6.2 se pueden ver los objetivos y los pesos utilizados en todos los métodos de decisión.

Además, cada familia de métodos, debido a su distinta metodología, necesita una información adicional para ser aplicados como son los umbrales y tipos de preferencia. En el caso de la familia de métodos Promethee en la tabla 6.3 se muestran estos umbrales y para la familia de métodos Electre se han recogido en la tabla 6.4.

El objetivo de este experimento (fig. 6.8) consiste en ajustar dichos parámetros de modo que la clasificación obtenida sea satisfactoria. A continuación, en un segundo experimento (fig. 6.10) se aplicará el método ya ajustado a otro escenario diferente,

comprobando la validez del procedimiento si las nuevas clasificaciones son coherentes con la preferencia de un conductor humano. Así pues los valores mostrados en las tablas 6.2, 6.3 y 6.4 provienen de una estimación intuitiva inicial, de acuerdo al conocimiento de un conductor experto, reajustada en función de las clasificaciones obtenidas (fig. 6.9).

Tabla 6.2: Información común (rango, objetivos y pesos) a todos los métodos de decisión.

Matriz de Decisión		C1	C2	C3	C4	C5
Información común a todos los métodos	Nombre Criterios	Longitud Total	Cambios de Sentido	Longitud Marcha Atrás	Índice de Suavidad	Separación Media
	Valor Máximo y Mínimo	343- 607	1-10	14- 427	18-47	9-17
	Maximizar ó Minimizar	Minimizar	Minimizar	Minimizar	Minimizar	Maximizar
	Pesos [0,100]	40	30	30	15	15

Tabla 6.3: Tipo de Preferencia y umbrales utilizados en cada criterio aplicados a los métodos Promethee.

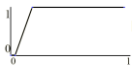
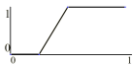
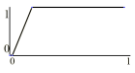
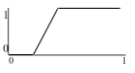
Matriz de Decisión		C1	C2	C3	C4	C5
Información Métodos Promethee I y II	Nombre Criterios	Longitud Total	Cambios de Sentido	Longitud Marcha Atrás	Índice de Suavidad	Separación Media
	Tipo de Preferencia	Lineal Entre Umbrales				
						
	Umbral de Indiferencia	**10**	**2**	**10**	**0**	**2**
	Umbral de Preferencia	**50**	**4**	**80**	**29**	**4**

Tabla 6.4: Umbrales utilizados en los métodos Electre para cada criterio.

Matriz de Decisión	Umbral de Concordancia		Umbral de Discordancia	
Electre I	0.5		0.5	
Electre II	Fuerte	Débil	Fuerte	Débil
	0.6	0.5	0.4	0.5

Figura 6.8: Experimentos para ajuste de umbrales y pesos para los TDM

En la tabla 6.3 se puede observar que se han utilizado, para todos los criterios, la función de preferencia lineal entre dos umbrales, debido a que proporciona un cambio paulatino y suave de la preferencia de una alternativa sobre otra en función del aumento de la diferencia entre ambas.

Es obvio que los resultados de los distintos métodos de decisión, dependen directamente de los valores utilizados en sus metodologías, de hecho es su principal crítica, pero hay que indicar que son métodos de toma de decisiones extensamente

aplicados a problemas de la vida real con muy buenos resultados. Haciendo un buen análisis inicial del problema y una elección lógica de los objetivos, pesos, y parámetros, los resultados de estos métodos son muy robustos en el sentido de que coincidirían con la elección que haría una persona sobre el mismo problema.

Esto queda ilustrado en la figura 6.9, donde sendos métodos reflejan la clasificación que un conductor humano haría. Como se ha comentado con anterioridad, este experimento tuvo como objetivo el ajuste y comprobación de los parámetros elegidos.

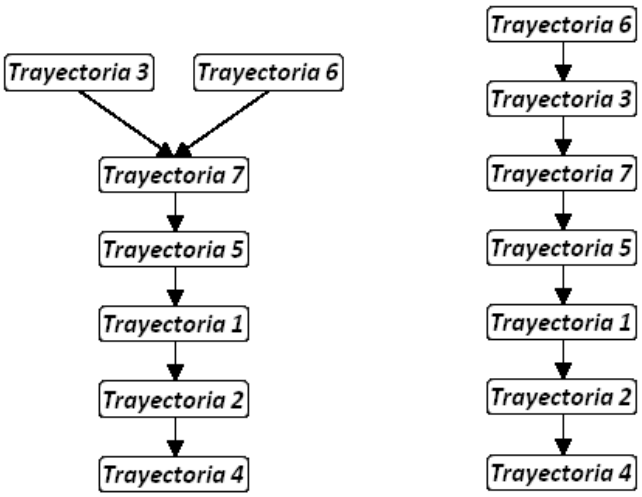


Figura 6.9: Resultados de los métodos Promethee I (izquierda) y Promethee II (derecha).

A continuación, con los valores establecidos, se procedió a aplicar los métodos TDM en un escenario diferente. En este caso la cocina del hospital, donde el robot debe acudir para ser aprovisionado.

Nuevamente el algoritmo RRT seguido de postprocesado proporcionó siete trayectorias que se muestran en la figura 6.10. Los valores que arroja cada trayectoria para los diferentes criterios quedan recogidos en la tabla 6.5

Si se analizan las distintas trayectorias de la figura 6.10, podemos observar que desde el punto de vista de un conductor humano, una ordenación lógica podría ser la siguiente: trayectoria 2, 6, 1, 5, 4, 7, y por último la trayectoria 3, siendo las tres últimas prácticamente iguales en cuanto a su deficiente calidad.

Figura 6.10: Diferentes trayectorias para las mismas configuraciones inicial y final.

Tabla 6.5: Información de las distintas alternativas de selección (trayectorias) común a todos los métodos.

Matriz de Decisión		C1	C2	C3	C4	C5
Criterios / Alternativas		Longitud Total	Cambios de Sentido	Longitud Marcha Atrás	Índice de Suavidad	Separación Media
Trayectorias	1	395.677	2	50.014	30.642	9
	2	343.274	2	14.484	23.628	11
	3	596.035	7	364.682	41.372	11
	4	575.256	6	426.773	46.337	15
	5	522.088	5	391.235	43.512	17
	6	413.241	1	350.968	18.5	14
	7	606.871	10	221.645	40.161	13

Los resultados obtenidos por las distintas familias de métodos multicriterio (figuras 6.11, 6.12 y tabla 6.6) no difieren de la elección humana antes realizada al observar las distintas trayectorias de la figura 6.10. En la figura 6.11 los métodos Promethee ofrecen prácticamente la misma ordenación de las trayectorias que la indicada anteriormente. Además, el método Promethee I genera como resultado adicional que las trayectorias 3, 4 y 7 son exactamente iguales ya que no cuenta con suficiente información para determinar cuál de ellas es la mejor.

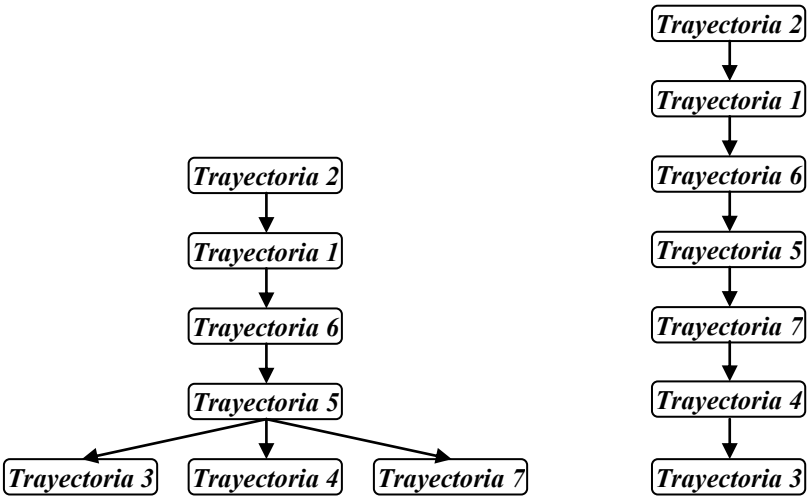


Figura 6.11: Resultado de los métodos Promethee I (Izquierda) y Promethee II (Derecha).

En la figura 6.12 se observan los grafos de superación obtenidos por la familia de métodos Electre, en los que se aprecia que la trayectoria 2 es indiscutiblemente la mejor ya que es la alternativa que supera al resto y no hay ninguna otra que la supere. A partir

de estos grafos de la figura 6.12, los métodos Electre proporcionan como resultado la ordenación de las distintas trayectorias (tabla 6.6), y se puede observar que las trayectorias 2, 6, 1 y 5 son las mejores y las trayectorias 3,4 y 7 son las peores.

Del análisis de los resultados de los distintos métodos, podemos deducir que independientemente de la metodología de los métodos de decisión, la ordenación de las distintas trayectorias es prácticamente la misma, coincidiendo en las mejores trayectorias a realizar y en las peores trayectorias a evitar.

Para un análisis exhaustivo de los resultados obtenidos de los distintos métodos de TDM presentados, se pueden ver en las tablas 6.7 y 6.8 la matriz de sobreclasificación y la matriz de superación para los métodos Promethee I y II. La matriz de superación es una matriz obtenida a partir de la matriz en la que se observan por filas las trayectorias que más superan o sobreclasifican y por columnas las trayectorias que son más superadas por otras. En las tablas 6.9 y 6.10 se pueden apreciar, respectivamente, las matrices de concordancia y discordancia de los métodos Electre I y Electre II.

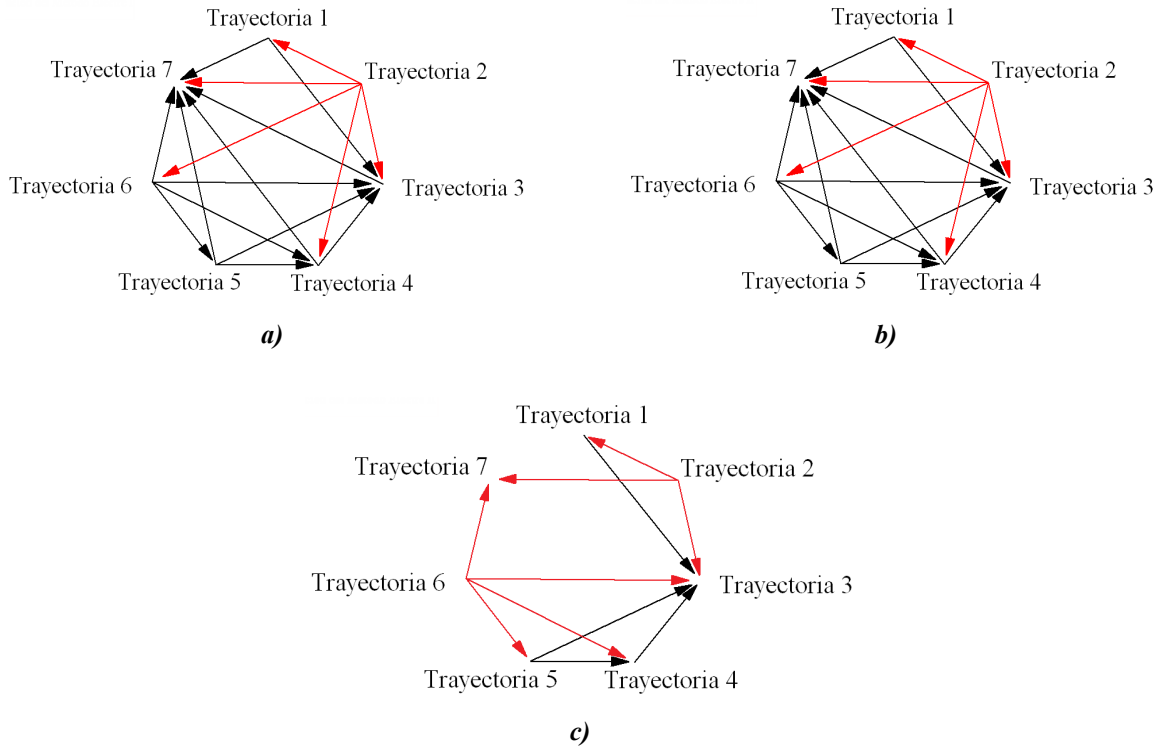


Figura 6.12: Grafos de superación de la familia de métodos Electre. (a) Grafo de superación del método Electre I, (b) Grafo de superación (débil) del Electre II y (c) Grafo de superación fuerte del Electre II.

Tabla 6.6: Resultados de los métodos Electre I y II obtenidos a partir de los grafos de superación.

Resultados	Kernel del Grafo				
Electre I	Trayectoria 2				
Electre II	Grafo Fuerte	Grafo Débil	Ordenación		
			Ascendente	Descendente	Final
	Trayectoria 2 Trayectoria 6	Trayectoria 2	Trayectoria 2 Trayectoria 1 y 6 Trayectoria 5 Trayectoria 4 Trayectoria 3 Trayectoria 7	Trayectoria 6 Trayectoria 2 y 5 Trayectoria 1 y 4 Trayectoria 3 Trayectoria 7	Trayectoria 2 y 6 Trayectoria 1 y 5 Trayectoria 4 Trayectoria 3 Trayectoria 7

Tabla 6.7: Matriz PI resultado de la aplicación de los métodos Promethee.

Matriz de PI		Trayectorias							Flujo de Salida
		1	2	3	4	5	6	7	
Trayectorias	1	-	0	0.812	0.832	0.705	0.289	0.807	3.445
	2	0.42	-	0.84	0.86	0.733	0.538	0.835	4.226
	3	0	0	-	0.191	0.063	0	0.122	0.376
	4	0.115	0.115	0.198	-	0	0	0.397	0.826
	5	0.115	0.115	0.423	0.403	-	0.058	0.654	1.769
	6	0.164	0.078	0.699	0.866	0.738	-	0.625	3.17
	7	0.115	0	0.236	0.255	0.244	0.231	-	1.081
Flujo de Entrada		0.93	0.309	3.208	3,407	2.483	1.116	3.439	-
Flujo Neto		2.515	3.917	-2.832	-2.581	-0.714	2.054	-2.358-	-

Tabla 6.8: Matriz de Superación del Método Promethee I y II donde se observa en ambos métodos que la alternativa 2 es la mejor y supera a al resto, además no es superada por ninguna otra alternativa.

Método Promethee I									Método Promethee II								
Matriz de Sobrecla-sificación		Trayectorias							Matriz de Sobrecla-sificación								
		1	2	3	4	5	6	7			1	2	3	4	5	6	7
Trayectorias	1	-		S	S	S	S	S	Trayectorias	1	-		S	S	S	S	S
	2	S	-	S	S	S	S	S		2	S	-	S	S	S	S	S
	3			-						3			-				
	4				-					4			S	-			
	5			S	S	-		S		5			S	S	-		S
	6			S	S	S	-	S		6			S	S	S	-	S
	7							-		7			S	S			-

Tabla 6.9: Matriz de Concordancia de los métodos Electre I y II

Matriz de Concordancia		Trayectorias						
		1	2	3	4	5	6	7
Trayectorias	1	-	0.231	0.855	0.855	0.855	0.538	0.885
	2	1	-	1	0.885	0.885	0.538	0.885
	3	0.115	0.115	-	0.346	0.346	0	0.538
	4	0.115	0.115	0.654	-	0	0.115	0.654
	5	0.115	0.115	0.654	1	-	0.115	0.654
	6	0.462	0.462	1	0.885	0.885	-	0.769
	7	0.115	0.115	0.462	0.346	0.346	0.231	-

Tabla 6.10: Matriz de Discordancia de los métodos Electre I y II.

Matriz de Discordancia		Trayectorias						
		1	2	3	4	5	6	7
Trayectorias	1	-	0.25	0.25	0.75	1	0.625	0.5
	2	0	-	0	0.5	0.75	0.375	0.25
	3	0.762	0.957	-	0.5	0.75	0.789	0.346
	4	0.912	0.998	0.171	-	0.25	0.96	0.497
	5	0.826	0.912	0.074	0	-	0.862	0.411
	6	0.729	0.815	0	0.125	0.375	-	0.313
	7	0.889	0.998	0.333	0.444	0.556	1	-

El objetivo final es proporcionar una medida de la mayor o menor bondad de las trayectorias encontradas. Los resultados experimentales presentados ratifican el buen funcionamiento del método. Estos resultados, permiten el uso de los métodos de toma de decisiones multicriterio para realizar en tiempo de real, analizar distintas trayectorias y decidir en función de los criterio prefijados, cuál o cuáles son las mejores trayectorias o desde otro punto de vista, cuál o cuáles trayectorias a evitar.

Con la unión de técnicas de planificación de maniobras y de metodologías de toma de decisión, se dota a los vehículos no holónomos de cierta Inteligencia Artificial para analizar en cada momento cual es su mejor trayectorias hacia un destino indicado y sobre todo a realizar replanificaciones cuando el vehículo está en escenarios dinámicos o cambiantes por lo que se necesitará constantemente generar y seleccionar las mejores trayectorias según vaya mutando el entorno real donde se desplace el vehículo.

6.3 Algoritmos genéticos y RRT

Uno de los resultados observados durante la investigación consiste en la multiplicidad de caminos y de maniobras alternativas durante el cómputo de la trayectoria. Esta disparidad nos induce a pensar en los algoritmos genéticos como herramienta de discriminación entre las diversas soluciones. Incluso, con las debidas modificaciones, el algoritmo genético podría crear la planificación completa combinando distintas maniobras (genes) y mediante la función de detección de colisión, ir rechazando los individuos no aptos.

Sin embargo no es fácil la implementación de estos métodos. Para que los algoritmos genéticos funcionen los genes deben representar una cercanía a la solución, un valor de sintonización más o menos óptimo de la trayectoria final buscada. No obstante, una misma maniobra ubicada en el mismo orden sobre una secuencia de otras dada, puede dar trayectorias que colisionen o trayectorias óptimas dependiendo de la secuencia anterior. Es decir, la simple asociación gen-maniobra no es factible a primera vista.

Se han de buscar por tanto otros parámetros para la identificación de los genes. Como trabajo preliminar se han estudiado en primer lugar los robots manipuladores, dado que éstos no tienen las restricciones no holónomas consideradas hasta ahora, con la intención de extender sus resultados en un futuro a los robots móviles. Para los robots manipuladores, la identificación de los genes se ha basado en las posiciones (un subconjunto de ellas) que definen las trayectorias. De este modo se pueden aprovechar las características de los algoritmos genéticos en cuanto al hallazgo de soluciones próximas a la óptima.

6.3.1 Aplicación a un robot manipulador simple

Para mostrar la aplicación de algoritmos genéticos sobre poblaciones obtenidas con el RRT, se ha escogido un robot compuesto por dos elementos articulados. El primer elemento se ha modelado como un rectángulo alargado que puede girar paralelo al plano horizontal utilizando como eje uno de sus extremos. El otro extremo conforma una

segunda articulación al estar unida al segundo elemento. Dicho elemento dispone en su extremo libre un dispositivo actuador que se desea posicionar en el plano horizontal de trabajo. En dicho plano existen tres obstáculos rectangulares. La figura 6.13 representa la situación planteada.

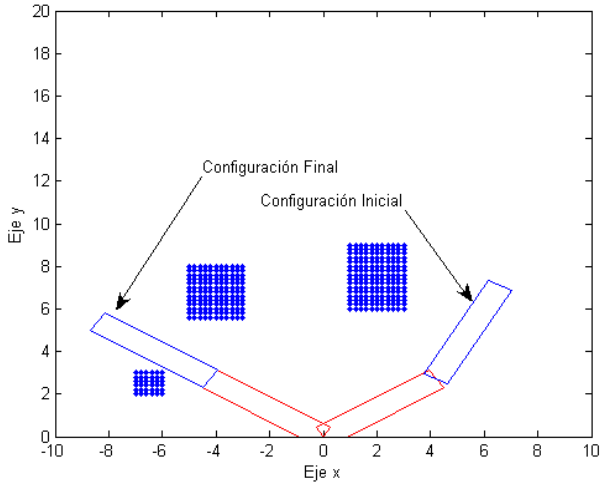


Figura 6.13: Problema de planificación propuesto

La posición del robot manipulador descrito queda definida por dos parámetros de configuración, que en concreto son los ángulos de sus dos articulaciones. Por tanto, el espacio de configuraciones del robot puede ser representado como un diagrama cartesiano cuyas abscisas corresponden con los ángulos de la primera articulación, y cuyas ordenadas contengan los ángulos de la segunda articulación. Este diagrama también es denominado “espacio articular”. Un punto en este plano se corresponde con una, y sólo una, posición del robot. Estas posiciones o configuraciones pueden entrar en colisión o no con los obstáculos del escenario. Para saberlo es necesario comprobar que los polígonos que modelan los brazos del robot intersecan con los que representan a los obstáculos. Así pues, para cada punto de dicho diagrama se marcan aquellos que implican colisión (el conjunto de los que representan colisión con un mismo obstáculo sería el “obstáculo-C” definido en la sección 2.1), quedando sin marcar el espacio de configuraciones libres de colisión. Este espacio se muestra en la figura 6.14, donde además se han señalado como puntos ‘A’ y ‘B’ los correspondientes a las posiciones inicial y final del robot representadas en la figura 6.14.

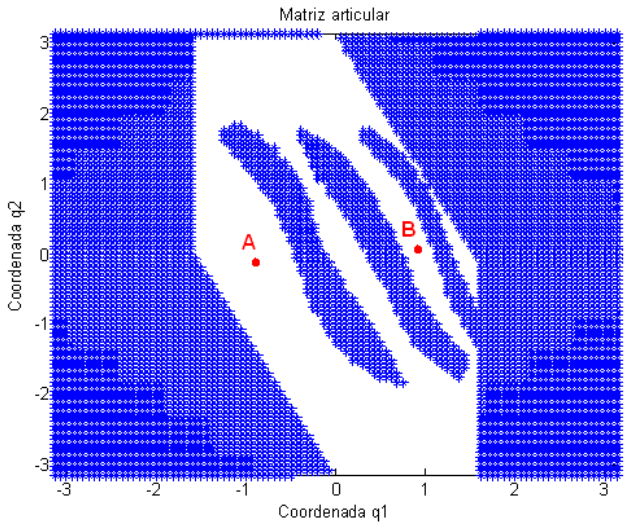


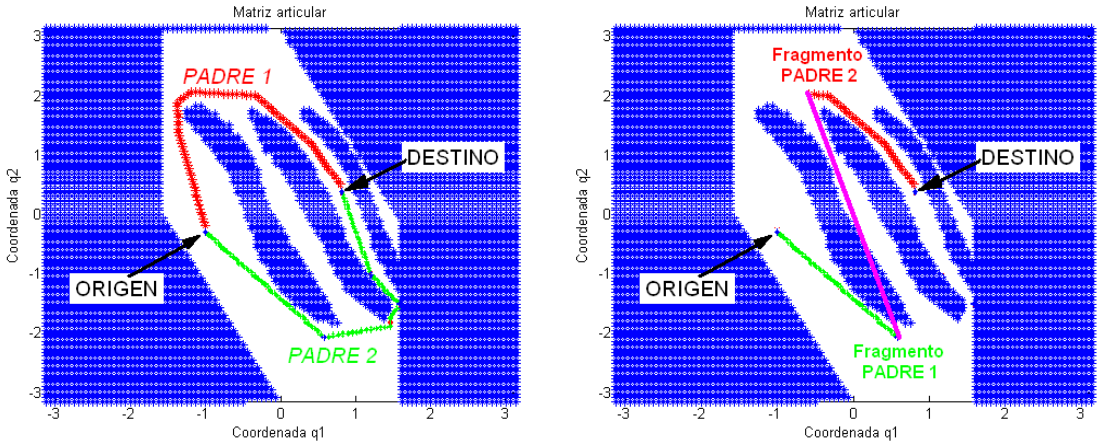
Figura 6.14: Espacio de configuraciones del robot

El procedimiento que a continuación se expone consiste en utilizar el RRT como generador de la población inicial a partir de la cual aplicar el algoritmo genético. Como ya se ha detallado en las secciones precedentes, el algoritmo RRT parte de las posiciones inicial y final dadas (puntos A y B), genera dos árboles a partir de éstas y devuelve una solución cuando se encuentran. Dicha solución consiste en una secuencia de puntos entre A y B, separados entre sí no más de un determinado valor ε dado, dentro del espacio de configuraciones libres de colisión. Al ejecutar el RRT de nuevo sobre el mismo problema, dada la naturaleza aleatoria del crecimiento de los árboles, se obtiene otra solución diferente a la anterior. De este modo se pueden generar con facilidad conjuntos de soluciones alternativas para llegar desde A hasta B.

En el problema planteado no existen restricciones no holónomas, con lo que en principio no sería necesario un postprocesado. Sin embargo, el inconveniente de estas soluciones estriba en la complejidad de las mismas, compuestas habitualmente de segmentos diminutos que cambian continuamente de dirección y que se aproximan excesivamente a los obstáculos. De ahí la necesidad un tratamiento posterior. En este caso se ha optado por aplicar un algoritmo genético. El algoritmo seleccionado es un Algoritmo Genético Básico (Goldberg, 1989). A partir de una población inicial (conjunto de individuos) se realizan una serie de operaciones: cruce, mutación, evaluación y selección. El resultado de estas operaciones será una nueva población cuyos individuos estarán más próximos a la solución óptima, y con ella se vuelve a iterar. El número de

iteraciones estará limitado bien con un valor dado, bien con con un criterio de calidad o convergencia definido.

En este problema particular, cada posible solución (individuo) estará compuesta por una secuencia de configuraciones (genes) que permiten conectar la configuración inicial con la final. El número de genes no está limitado, de forma que se permite obtener individuos de distinta longitud. Dada una población de partida (obtenida con el RRT), la primera operación es el cruce. En la operación de cruce los individuos de la población son emparejados al azar. De cada "progenitor" (ver Fig. 6.15) se extrae un fragmento de trayectoria definido por un punto de la misma, escogido de forma aleatoria, y un extremo (la configuración inicial o la final). Cada par de estos fragmentos, correspondientes a progenitores diferentes y a extremos diferentes, se unen utilizando segmentos rectos para lograr continuidad (ver Fig. 6.15). Estos segmentos rectos son conjuntos de puntos obtenidos mediante interpolación entre sus extremos aplicando una distancia de separación ε Consecuentemente, por cada pareja se generan dos hijos, cada uno con fragmentos de ambos padres.



a) *b)*

Figura 6.15: Operación de cruce

A continuación, con una probabilidad *'p'*, tiene lugar la mutación. En la operación de mutación se elimina un fragmento de un individuo al azar. Para ello se generan dos índices aleatorios dentro de la secuencia que compone el individuo. En la figura 6.16 corresponden con los puntos *A* y *B*. El fragmento comprendido entre *A* y *B* es eliminado. Seguidamente se genera un punto aleatorio dentro del espacio de configuraciones libre

de colisión (punto P de la Fig. 6.16). Una vez obtenido, se trata de unir dicho punto a los restos del individuo con segmentos rectos ($\overline{AP}$ y $\overline{PB}$). Al igual que en el caso anterior los segmentos rectos se construyen por interpolación. De este modo el individuo mutado consistirá en el individuo de partida sustituido en parte por los segmentos $\overline{AP}$ y $\overline{PB}$.

Esta operación no tiene lugar en todas las iteraciones. Esto es debido a que, realizada en exceso, la mutación no favorece la convergencia del método. Sin embargo, usada con una probabilidad *'p'* adecuada, puede aportar nuevos y valiosos genes, inexistentes en las poblaciones iniciales, y determinantes en la progresión hacia la solución óptima.

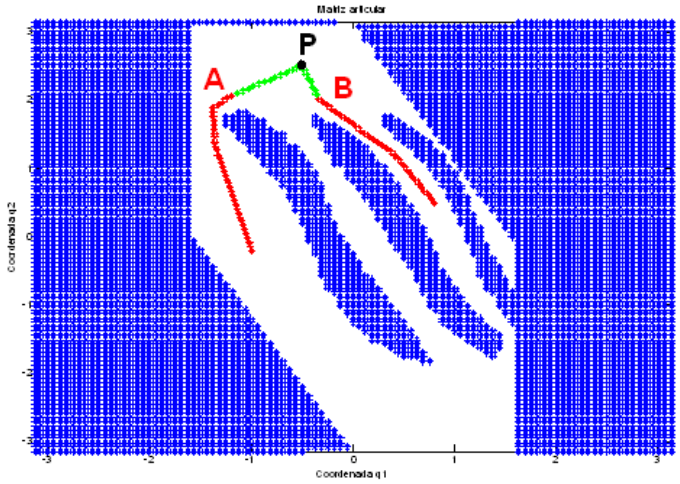
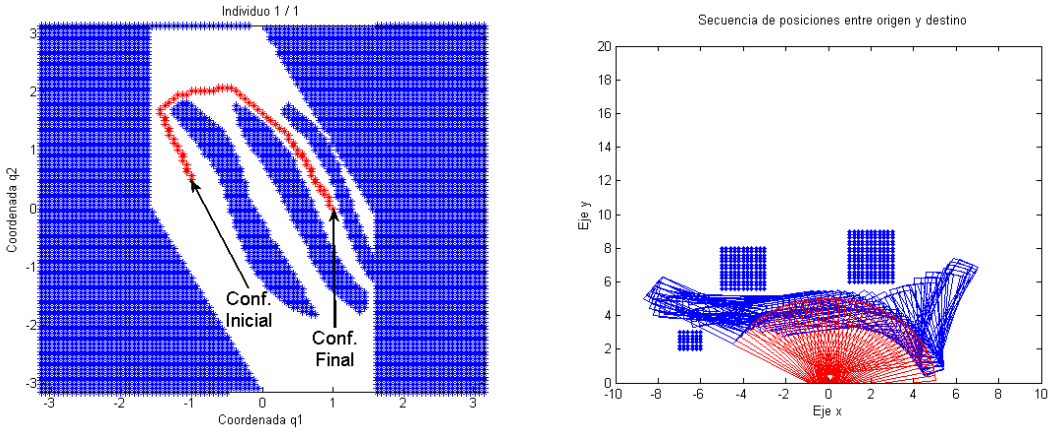


Figura 6.16: Operación de mutación

Finalmente se realiza la selección a partir de la población alcanzada, conjunto compuesto por los individuos padres, los individuos hijos obtenidos y el individuo mutado si eventualmente fue generado. Dicha selección se realiza merced a una función de coste. En este caso la función de coste se ha elegido considerando sólo dos aspectos: la distancia total recorrida y la cercanía a los obstáculos. Una vez evaluados todos los individuos, se clasifican en función de dicho coste y se eliminan del conjunto los inferiores hasta obtener un conjunto resultado de la misma dimensión que el de partida. Con esta nueva población se vuelve a iterar.

Cuando se realiza un número determinado de iteraciones o se cumple un determinado criterio de calidad, el algoritmo devuelve el mejor individuo como solución (ver figura 6.17).



a) Espacio articular *b) Espacio cartesiano*

Figura 6.17: Resultado del algoritmo genético.

En los resultados expuestos se han utilizado poblaciones de 20 individuos, produciéndose la convergencia en torno a las 14 iteraciones. También se ha probado un algoritmo de postprocesado sencillo, basado en la rectificación de la trayectoria por tramos rectilíneos, y se ha comprobado que la solución del algoritmo genético es de mayor calidad. Su desventaja, como era de prever, es que resulta ser un método más costoso desde el punto de vista computacional.

6.3.2 Aplicación al robot CAPAMAN

A continuación se muestra una aplicación del método anterior sobre un robot real. El manipulador paralelo CaPaMan (Cassino Parallel Manipulator) ha sido desarrollado en el Laboratorio de Robótica y Mecatrónica (LARM) de la Universidad de Casino. Dicho robot es un manipulador paralelo con tres grados de libertad, compuesto por una base fija y una plataforma móvil conectadas entre sí por tres brazos, como se puede ver en la fig. 6.18.

Cada brazo consta de un paralelogramo articulado que lleva una barra acopladora que se une a una varilla vertical mediante un par prismático. La varilla vertical se conecta a la plataforma móvil a través de una junta esférica. Los motores actúan sobre los ángulos de los paralelogramos, haciendo bascular la plataforma. Cada uno de los paralelogramos puede moverse con independencia de los otros.

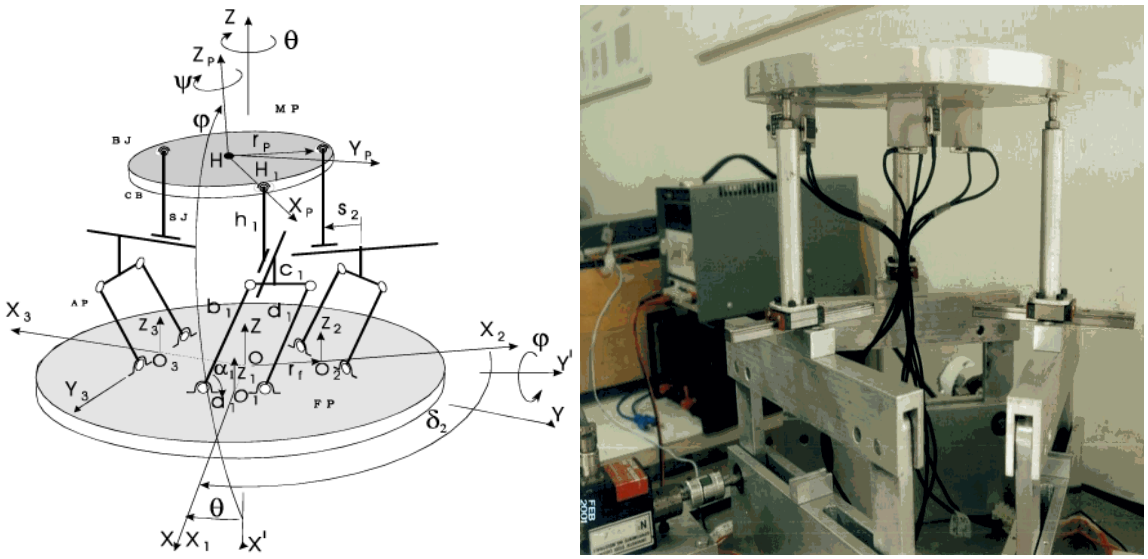


Figura 6.18: Robot CAPAMAN

La representación geométrica de CaPaMan utilizada en este trabajo para la detección de colisiones puede observarse en la figura 6.19. En ella se ha añadido sobre la plataforma un cilindro y un prisma representando una herramienta.

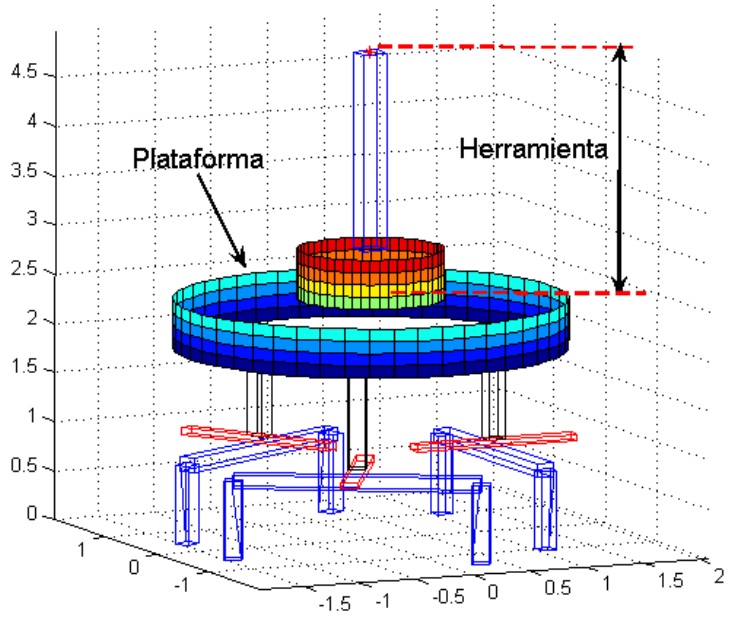


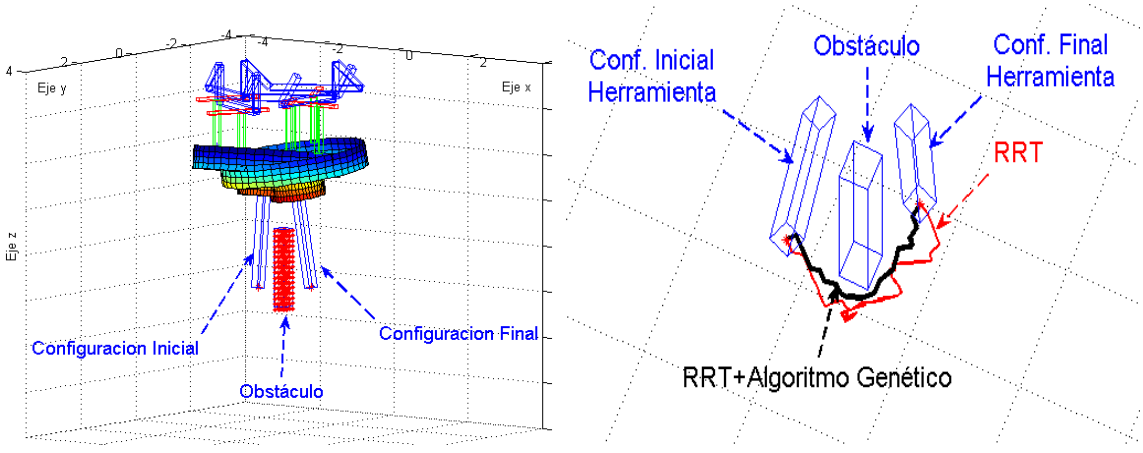
Figura 6.19: Modelado de CAPAMAN

Los algoritmos genéticos se aplican en este caso de la misma forma que en la sección anterior. Así cada trayectoria entre las posiciones origen y destino será un

individuo, y cada configuración, un gen. La función de coste, sin embargo se ha sustituido aquí por los métodos de decisión multicriterio. La intención es facilitar la incorporación del conocimiento experto en cuanto a la valoración de los criterios de calidad (longitud de trayectoria, cercanía a los obstáculos, etc.). En este sentido se ha dado un paso más, permitiendo el uso de etiquetas lingüísticas al utilizar el método de decisión multicriterio FPromethee. Este método de decisión (Martin y otros, 2003), es la extensión difusa del método Promethee (Brans y Vincke, 1985) visto en la sección anterior, para poder utilizar, dentro de su metodología, tanto información cualitativa como cuantitativa o crisp. Para ello se ha utilizado la teoría de conjuntos difusos (Zadeh, 1975).

La ilustración 6.20a muestra la descripción geométrica de la configuración inicial y la configuración final que se desea alcanzar. Se representa también un obstáculo que impide el movimiento directo de la plataforma desde una configuración a otra.

En la Fig. 6.20b se adelanta el resultado de este método presentando en color negro la trayectoria del punto final de la herramienta que proporciona el método propuesto (algoritmo genético con evaluación TDM). Para simplificar el dibujo sólo se han dibujado los prismas de las herramientas y el obstáculo Esta solución puede compararse con una de las trayectorias proporcionadas por el RRT que se ha representado en color rojo.



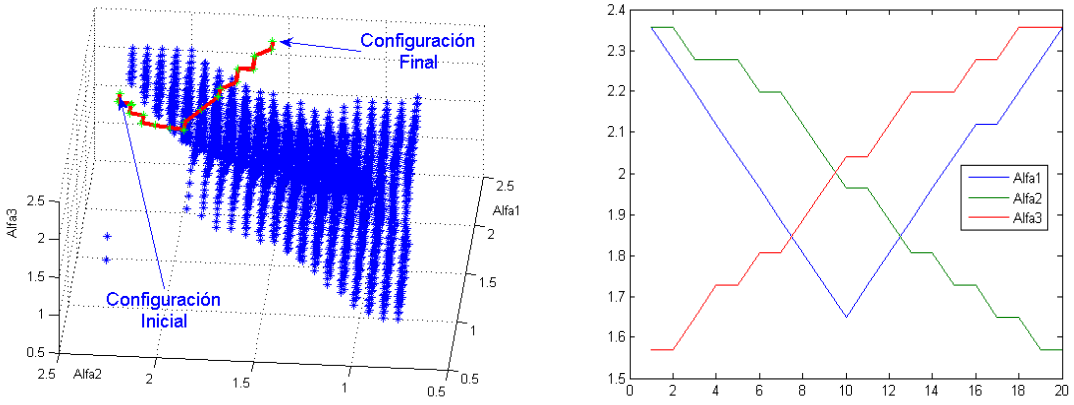
a) Planteamiento *b) Soluciones*

Figura 6.20: Trayectorias del robot CAPAMAN

En la Fig. 6.21a se ilustra la trayectoria de la solución obtenida en el espacio de configuraciones. Obsérvese que dicho espacio es de dimensión tres (cada uno de los

ángulos de los paralelogramos del robot), que en color azul se representan las configuraciones con colisión y que en color rojo se presenta la trayectoria generada.

En la Fig. 6.21b se muestra la evolución de las variables articulares desde su valor inicial a su valor final, y que desde el punto de vista del control contiene la solución en el formato más fácil de implementar.



a)Espacio de configuraciones *b) Evolución temporal*

Figura 6.21: Solución obtenida del AG con TDM

La calidad de las soluciones ofrecidas es similar a la del método de algoritmos genéticos con función de coste. La mejora introducida ha consistido en la facilidad para definir los criterios de calidad de las soluciones ofrecidas, dado que con FPromethee basta con el uso de etiquetas lingüísticas y funciones de preferencia sencillas.

6.5. Conclusiones

En este capítulo se ha explotado la capacidad del algoritmo RRT para generar trayectorias diferentes ante un mismo problema propuesto. Para ello se necesitaba un método de selección en base a ciertos criterios como la longitud recorrida, el número de inversores, etc. En lugar de definir una función de coste se prefirió utilizar los métodos de decisión multicriterio debido a su capacidad para trasladar el conocimiento experto y

proporcionar resultados semejantes a los que ofrecería un decisor humano. Todo ello se ha constatado con experimentos sobre escenarios y robots reales.

Dada la disparidad de soluciones existentes se plantea el problema de hallar la óptima. En una primera aproximación se han utilizado algoritmos genéticos sobre un robot manipulador comprobando la convergencia hacia trayectorias de elevada calidad.

Como último paso se ha sustituido la función de coste por métodos de decisión multicriterio con definición de los criterios de calidad en términos borrosos. Se ha pretendido dar un paso más en la facilidad de traslación de conocimiento experto. Para ello se han planteado experimentos sobre un robot manipulador paralelo. Como resultado se ha obtenido una trayectoria de calidad superior a las generadas con el algoritmo RRT.

Queda por tanto la aplicación de estas técnicas a robots móviles con restricciones no holónomas. Sin embargo aún quedan cuestiones que resolver, como por ejemplo la definición de los genes para este caso. No obstante, del trabajo realizado se desprende que los métodos TDM suponen una técnica fácil y eficaz para la selección de las poblaciones.

CAPÍTULO 7 Aplicaciones y experimentación

7.1 Introducción

Los objetivos perseguidos en este capítulo han sido estudiar la viabilidad de las trayectorias ofrecidas por los planificadores (RRT y VODEC seguidos de postprocesado), su implementación práctica en robots reales, y la capacidad y precisión del robot al seguirlas. En este sentido se han probado trayectorias suavizadas con *spline* cúbicas y *β-spline* (Gómez-Bravo y otros, 2008c) con resultados similares a la aplicación directa de las maniobras restringidas, por lo cual en esta tesis no se hace referencia al uso de las mismas.

Dada la velocidad de cómputo de los planificadores, se ha podido aprovechar esta cualidad para escenarios dinámicos, es decir, escenarios cuyos obstáculos pueden cambiar. Por tanto, se ha incluido como objetivo el estudio del comportamiento del robot cuando el algoritmo de planificación altera la trayectoria inicial ante este tipo de eventos. Dichos eventos son detectados por una red inalámbrica de sensores distribuida capaz de informar de cambios en el escenario.

El robot utilizado en los experimentos ha sido el Romeo4R, de la universidad de Sevilla. Este robot consta de dos motores eléctricos vinculados a sendas ruedas traseras. La implementación del control establece su configuración como de tipo Ackerman. En cuanto a sensores, el Romeo4R cuenta con encoders para odometría, GPS, giróscopo, láser y varios dispositivos ultrasónicos capaces de distinguir la proximidad de objetos. No obstante, en los experimentos sólo se han utilizado los encoders (asociados a las ruedas y al volante del vehículo), el giróscopo y el GPS para establecer su localización. En la figura 7.1 se muestra el Romeo4R.

Figura 7.1: Robot Romeo4R

Este capítulo continúa con tres apartados. En el primero se procederá a explicar los recursos utilizados para realizar los experimentos. Se detallarán los elementos integrantes (Robot, red de sensores, y estaciones de control), y su interconexión. Además se expondrán los algoritmos de seguimiento utilizados y los planificadores (variantes aplicadas entre las vistas a lo largo de esta tesis), El segundo apartado muestra los experimentos realizados y sus resultados. El tercero y último resume las conclusiones.

7.2 Control y ejecución de maniobras

7.2.1. Arquitectura de control

Los experimentos realizados han necesitado dos elementos: el robot y un PC portátil donde se ha ejecutado el planificador. La comunicación entre ambos se ha establecido mediante wifi. Dentro del robot, una CPU integrada en el mismo recibía la información del planificador y ejecutaba las labores de control precisas. Además, los experimentos relacionados con escenarios dinámicos han requerido otros elementos adicionales: una red inalámbrica de sensores y un PC portátil adicional para su gestión. A continuación se detallan cada uno de estos elementos.

El robot utilizado ha sido el "ROMEO-4R" (fig. 7.1) de la universidad de Sevilla (Ollero y otros, 1999), de configuración tipo Ackerman (fig. 4.3), de dimensiones 2'8 m. x 1'4 m. (eje de ruedas trasero a 0'5 m. del borde posterior), con un radio de curvatura mínimo de 2,5 metros. Este robot está equipado con diferentes sensores entre los que se encuentra un GPS (Daily y Bevly, 2004), un giróscopo y encoders asociados a las ruedas y al volante. Todos estos sensores concurren a una tarjeta de adquisición de datos unida a la CPU, cuyo sistema operativo es linux. La estimación de la posición se obtiene utilizando un filtro de Kalman extendido (Grewal y Andrews, 1993). Como sistemas de comunicación cuenta con una tarjeta IEEE 802.11g (calidad wifi) y un radio-modem de uso exclusivo para el sistema GPS. En la CPU del robot se ha realizado la estimación de la posición y el seguimiento de las trayectorias recibidas a tramos desde el PC principal, donde se ejecutaban los algoritmos de planificación. Todas las aplicaciones en la CPU del romeo han sido programadas en C++.

El PC portátil dedicado a las tareas de planificación posee un Intel Core Duo T2300 (1.66 GHz) y 1024 MB de RAM DDR II. Funciona bajo el sistema operativo Windows XP y las aplicaciones han sido implementadas en JAVA. Para las conexiones con los otros equipos (CPU del robot y portátil de la red de sensores) dispone también de una tarjeta IEEE 802.11g. Este equipo ha centralizado el control de alto nivel, solicitando al robot información de posición y a la red de sensores su estado, procesándola, realizando la planificación cuando la situación lo requería y ordenando su ejecución por

tramos al robot. También se ha incluido la capacidad de detener al robot y de reajustar sus parámetros de control.

El sistema de sensores utiliza radiofrecuencia (900 MHz) para comunicación inalámbrica con una estación base, de características similares al portátil utilizado en planificación. Dicha estación se encarga de monitorizar constantemente el estado de los sensores, registrando los cambios y actualizándolos sobre un mapa del escenario donde se reflejan los obstáculos (Gómez Bravo y otros, 2007a). En este caso particular los sensores detectan la presencia de vehículos en las distintas plazas de aparcamiento, por tanto, los obstáculos detectados consisten en los límites de dichas plazas cuando un automóvil las ocupa.

El escenario de todos los experimentos ha sido un aparcamiento donde hay vías de paso libre, plazas que pueden estar ocupadas o no, y zonas de acceso prohibido, ya sea por la ausencia de señal GPS o por la impracticabilidad del terreno para el robot (figura 7.2).

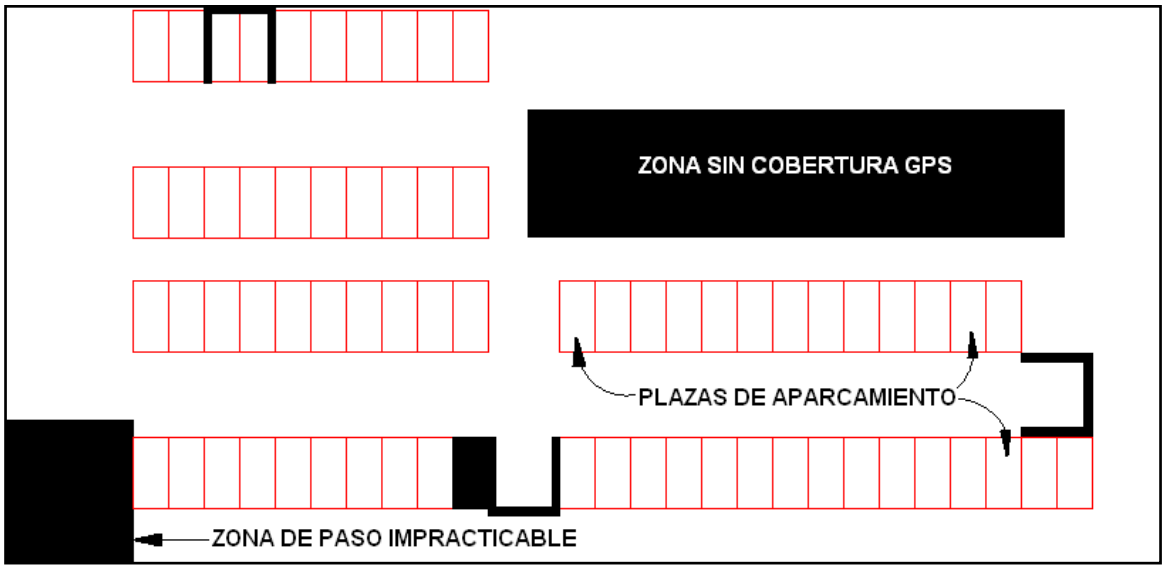


Figura7.2: Escenario de los experimentos.

Como se ha comentado, el planificador se ejecuta en un PC independiente, que se comunica por un lado con la estación base de los sensores, y por otro con la CPU del robot. Esta segmentación de los procesos sobre distintos microprocesadores resulta útil a efectos de velocidad de cómputo. Un esquema de esta estructura de control se presenta en la fig. 7.3.

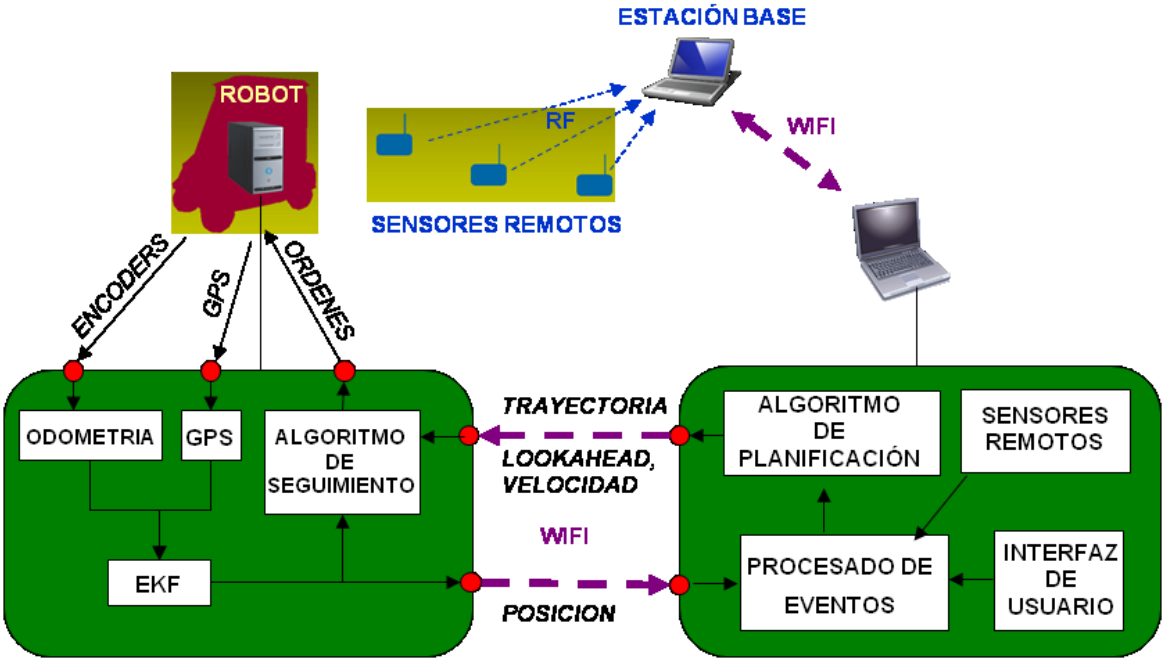


Figura 7.3: Esquema de control adoptado.

Como puede observarse el control se realiza a dos niveles. En un primer nivel se realiza un seguimiento de trayectorias (conocido como “path-tracking”) donde el robot tiene como misión acercarse lo más posible a una trayectoria dada. Para ello cuenta con un algoritmo de seguimiento que iterativamente evalúa la posición del vehículo, calcula el error existente entre dicha posición y el punto de trayectoria más próximo, y basándose en ello y en la trayectoria pendiente de recorrer, calcula las órdenes de control más adecuadas.

En un segundo nivel se realiza la planificación de las trayectorias. El algoritmo de planificación actúa al principio y eventualmente durante el recorrido. Esto ocurre cuando los sensores detectan obstáculos nuevos que interrumpan la trayectoria o bien cuando el robot se aleje de la trayectoria en demasía (tanto como para colisionar con otros obstáculos o como para que el algoritmo de seguimiento no pueda funcionar adecuadamente). Los cambios en los obstáculos son advertidos por la estación base de la red de sensores y los distanciamientos excesivos por el propio robot. Por su parte el planificador remite la trayectoria, o los fragmentos modificados de la misma, además de ciertos parámetros de configuración específicos para los diferentes tramos a recorrer según sus características.

7.2.1.1 Comunicaciones con la red de sensores.

La red de sensores se comunica con la estación base (un PC portátil) a través de señales de radio en la banda de los 900 MHz. El tamaño de la información es mínimo, dado que sólo tienen dos estados: o detectan presencia de vehículos, o no. Por tanto la actualización del estado es muy rápida. Se realiza a través de rondas de muestreo periódicas. La estación base debe comunicar cualquier cambio en el escenario tan pronto este ocurra. Dada la infraestructura disponible, se disponía de dos alternativas de implementación a nivel de transporte de acuerdo a la pila OSI: TCP o UDP. Por encima, a nivel de aplicación, tanto en la estación base de los sensores como en el PC portátil para planificación, se diseñaron aplicaciones en JAVA. Por debajo, a nivel de enlace se utilizó wifi. En general, cuando el tiempo es un factor crítico o bien se requiere cierto ritmo en la llegada de información con independencia del posible deterioro de segmentos puntuales, se opta por UDP. Es un protocolo más rápido, pero carece de los servicios de confirmación de tramas de TCP. En este caso los tiempos del planificador y de respuesta del robot estaban en torno al segundo, mientras que el envío de tramas TCP sólo acapara unos pocos milisegundos. Consecuentemente se optó por este protocolo debido a su fiabilidad.

Las rutinas en JAVA destinadas a la comunicación comparten un protocolo sencillo en código ASCII. La estación base transmite los cambios detectados en el escenario mediante una simple línea con 5 números separados por espacios. El primero se deja para posibles ampliaciones futuras. El segundo estipula el tipo de servicio a realizar. Los tres últimos consisten en la posición (abscisas y ordenadas) y el estado del sensor situado en esta posición. Al recibir esta información, el algoritmo de planificación localiza la plaza de aparcamiento ocupada por el sensor. Si ésta está libre y el sensor indica presencia de vehículo, la transforma en obstáculo; y viceversa (ver fig. 7.4).

Figura 7.4: Protocolo de comunicación para la red de sensores.

Aunque se podían haber realizado todas estas operaciones en un mismo equipo, se consideró interesante desvincular ambos procesos dado que, si bien en estos

experimentos el número de sensores era pequeño, en aplicaciones genéricas este número podría ser elevado y su información mucho más abundante y compleja. Así pues, el reparto de tareas entre CPUs diferentes mejora las prestaciones del sistema. De este modo, en los experimentos, la aplicación principal (la dedicada a las tareas de planificación) ejecuta un proceso en un hilo independiente dedicado a la escucha continua al socket utilizado para la comunicación con la estación base. Sólo cuando surge un evento de cambio en el escenario, este hilo comprueba si la trayectoria vigente es afectada. Si esto no sucede, el planificador anota el cambio y vuelve a quedar a la espera. Si por el contrario los sensores informan de un nuevo obstáculo en el camino del robot, el planificador ejecuta el algoritmo correspondiente para corregirlo, detiene el robot si existe peligro de colisión y le envía la nueva trayectoria a seguir. Para mayor aclaración véase la figura 7.5. Además, en la sección 7.3.2.4 se ampliará con mayor detalle las interacciones con la red de sensores.

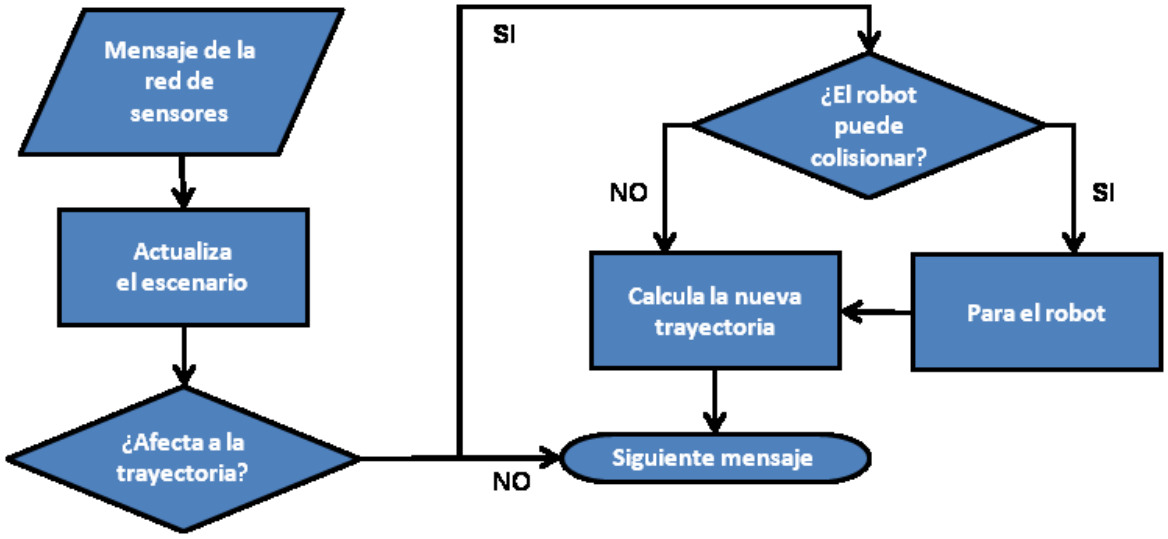


Figura 7.5: Procesado de eventos de la red de sensores.

7.2.1.2 Comunicaciones en el robot.

Las comunicaciones entre el PC portátil dedicado a la planificación y el robot resultaban más complejas. En primer lugar existían diferencias entre el software utilizado. Los algoritmos de planificación se habían programado sobre JAVA, y el sistema operativo del PC portátil era Windows XP. Por el contrario la CPU del robot tenía instalado un sistema operativo LINUX y sus algoritmos estaban programados en C++.

Además, a priori, no se sabía a qué recursos de los que disponía el robot iba a ser necesario acceder desde el algoritmo de planificación. Incluso las estructuras de datos (clases de C++) a intercambiar podían sufrir cambios debido a las necesidades detectadas en los experimentos. Por tanto se necesitaba una interconexión software flexible a todos estos cambios.

Por otro lado, la arquitectura de comunicaciones interna del robot estaba diseñada de forma abierta, capaz de contender con un gran número de sensores y actuadores y con una amplia diversidad de necesidades de tráfico de información diferentes. Esta arquitectura de protocolos de comunicación se conoce como YARP.

YARP corresponde a las siglas de "*Yet Another Robot Platform*", y consiste en un conjunto de librerías, protocolos y herramientas para lograr desacoplar los diferentes dispositivos que entran en juego en un sistema robótico. Nació de la necesidad de mantener el código desarrollado para robots, en constante cambio debido a la evolución continua de la tecnología en sensores y actuadores, especialmente en robots humanoides. La forma de hacerlo es desvincularlo al máximo de los dispositivos utilizados. Así pues YARP proporciona un protocolo de comunicación situado en capa 4 de la pila OSI, que solventa la comunicación entre equipos. Con este sistema, las rutinas en C++ sólo han de incorporar unos identificadores de puertos de comunicación y utilizar instancias de clases diseñadas de forma específica para incorporar los datos a intercambiar. YARP puede implementarse sobre diferentes sistemas operativos, es software libre y abierto, y sus librerías están escritas en C++.

Existen alternativas a YARP, definidas como "middleware", aludiendo a su ubicación entre los dispositivos físicos utilizados por el robot (sensores y actuadores de diverso tipo) y el software de alto nivel donde se realiza la toma de decisiones. Algunos ejemplos es estas arquitecturas alternativas son *Miro*, desarrollado por la universidad de Ulm (Alemania), *Orca*, *UPnP Robot Middleware* (Instituto de Ciencia y Tecnología de Korea), *Aware* (universidades de Stuttgart y Sevilla), *Wurde* (universidad de Washington), etc. (Nader y otros, 2008). En general son variados en cuanto a prestaciones, aunque todos ellos persiguen facilitar la interacción entre dispositivos robóticos.

La ausencia de un sistema como YARP implicaría la elaboración específica de funciones dedicadas a la comunicación entre dispositivos. Esto podría suponer la rigidez del sistema a futuros cambios, tales como ampliaciones o renovaciones en la tecnología.

A la hora de implementar YARP sobre un sistema, es necesario establecer un esquema de puertos de comunicación. Estos puertos, similares a los equivalentes en la capa de transporte de la pila OSI, se identifican con nombres y se vinculan a cada dispositivo de forma jerárquica. Por ejemplo “romeo/gps/data” especifica la salida de la información de posición que proporciona el dispositivo GPS instalado sobre el robot Romeo4R. Cada puerto tiene su clase asociada con la estructura de datos a intercambiar.

Era por tanto imprescindible poder acceder a esta infraestructura de comunicaciones para disponer de la flexibilidad necesaria durante la investigación. Así pues, la solución adoptada consistió en crear un programa en C++ que ejecutase las librerías YARP en el portátil destinado a la planificación, que se comunicara a través de sus propios puertos a los del robot, y que contactase con el planificador a través de un socket interno a la aplicación de JAVA. De este modo el PC portátil ejecutaría dos procesos: el algoritmo de planificación y el proceso de intermediación con los puertos YARP del robot. Esta solución demostró ser eficaz y flexible pues no costó demasiado realizar cambios cuando fue necesario. En la figura 7.6 se muestra un esquema de los puertos utilizados. Los procesos que toman parte en la comunicación YARP son (figura 7.6):

-DCX: Se refiere a la tarjeta de adquisición de datos unida a la CPU del Romeo4R. En ésta se leen los valores de los encoders (en concreto velocidad y curvatura) a través del puerto “/romeo/dcx/data” y también se establecen las referencias a seguir por los servomecanismos del robot (puerto “/romeo/dcx/ref”).

-GPS indica el sensor homónimo. Transmite en coordenadas UTM la posición del robot por el único puerto definido en la figura.

-GIROSCOPO: detecta la orientación y la transmite.

-PLANIFICADOR: Se refiere a la aplicación en C++ que se ejecuta en el PC portátil dedicado a las tareas de planificación. Por un lado recibe la posición estimada del robot (“uhuplanner\position”) y el estado del algoritmo de seguimiento (en el puerto “uhuplanner\state” detecta si está en funcionamiento, inversor alcanzado, etc.), información que transmitirá a través de un socket interno al algoritmo de planificación. Por otro envía la trayectoria (“uhuplanner\trajectory”) y los parámetros de seguimiento

(lookahead y velocidad con el puerto “uhuplanner\lookahead”). También a través de este último puerto puede obligar al robot a detenerse.

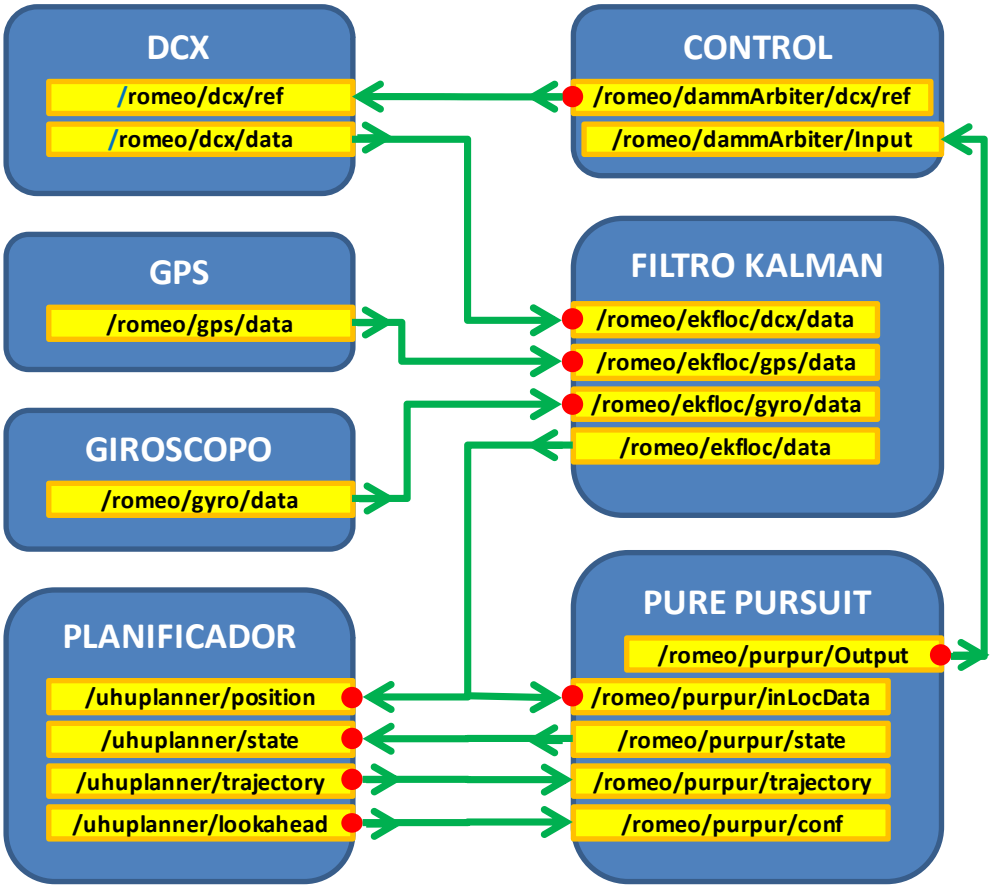


Figura 7.6: Esquema de comunicaciones YARP.

-CONTROL: Arbitra entre las distintas órdenes de control a seguir por diferentes procesos concurrentes. Este proceso en realidad actúa de mero mensajero, dado que los procesos dedicados a la evitación de obstáculos que podrían interferir se desactivaron en los experimentos.

-FILTRO KALMAN: Su función es proporcionar la estimación de la posición a través del puerto “romeo/ekfloc/data” en función de la información obtenida por el GPS (“romeo/ekfloc/gps/data”), el giróscopo (“romeo/ekfloc/gyro/data”) los encoders (“romeo/ekfloc/dcx/data”).

-PURE PURSUIT: Es el proceso dedicado al seguimiento de trayectorias. Las recibe a través del puerto “/romeo/purpur/trajectory”, y las ejecuta de acuerdo a los parámetros recibidos por “romeo/purpur/conf”, y a la posición que le llega por

"romeo/purpur/inLocData". Su resultado altera de forma continua las referencias de curvatura y velocidad a través del puerto "romeo/purpur/Output". Además, periódicamente actualiza su estado para que el planificador pueda leerlo a través del puerto "romeo/purpur/state".

Las flechas indican la dirección de la información. Es necesario indicar en YARP si un puerto es de salida de datos o de entrada de los mismos. También se distinguen entre puertos servidores (continuamente abiertos a la espera de solicitud de transmisión) y puertos clientes (los que solicitan la información eventualmente). Estos últimos son los que se han marcado en la figura 7.4 con un pequeño círculo rojo.

7.2.2 Algoritmo de seguimiento

A continuación se detalla el algoritmo utilizado para el seguimiento de las trayectorias. Este algoritmo parte del conocido como "persecución pura", sobre el que se realizaron modificaciones específicas para el seguimiento de las maniobras restringidas. El primer apartado corresponde con los conocimientos teóricos generales que se pueden hallar en la documentación científica relativa al algoritmo de persecución pura. El segundo explica las singularidades aportadas durante la investigación destinadas a mejorar su comportamiento.

7.2.2.1 Persecución pura

El método utilizado para el seguimiento de trayectorias en este robot ha sido una modificación del de persecución pura ("*pure pursuit*" en nomenclatura inglesa) (Ollero, 2001). La idea es establecer un punto objetivo dentro de la trayectoria a seguir, se calculan las órdenes de control necesarias para alcanzarlo, y se transmiten. Esta secuencia se repite cambiando el punto objetivo tal y como si se estuviera persiguiendo a un móvil que se desplaza por la trayectoria a la misma velocidad del robot.

Este sistema se explica con el modelo geométrico que se muestra en las figuras 7.7 y 7.8. Cuando el robot realiza un camino circular de radio *r* como el mostrado en la figura 7.7, se irá alejando de su trayectoria rectilínea (punteada en la figura 7.7) de modo progresivo hasta alcanzar el punto *B* pasados α radianes, donde esta desviación resulta

ser Δ. La distancia entre los puntos A y B se denomina "*Lookahead*" y se denota con L_H, su valor será:

$$L_H = 2r \cdot sen(\alpha/2) = 2r\sqrt{\frac{1-\cos\alpha}{2}} = \sqrt{2r^2(1-\cos\alpha)} \qquad (7\text{-}1)$$

Por otro lado:

$$\Delta = r \cdot (1-\cos\alpha) = r \cdot \frac{L_H^2}{2r^2} = \frac{L_H^2}{2r} \qquad (7\text{-}2)$$

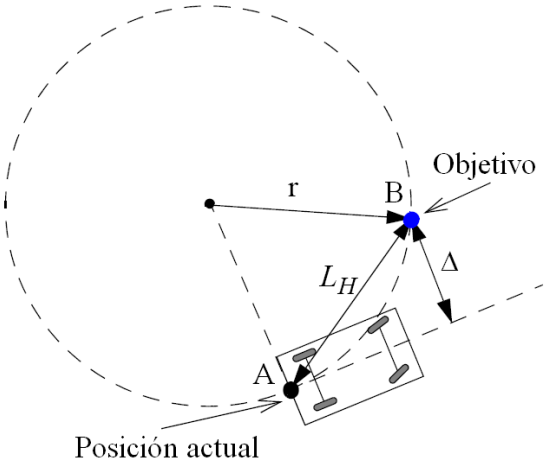


Figura 7.7: Trayectoria Circular.

A la inversa, si se desea alcanzar un punto *B* desde el actual *A*, que se aparta una distancia Δ de la trayectoria y situado a una distancia L_H, se necesita conocer el radio de curvatura *r* de la maniobra que lo logre. A partir de este radio de curvatura puede deducirse el ángulo de giro del volante del robot ϕ tal y como aparece en la figura 4.3. Por tanto se tiene:

$$r = \frac{L_H^2}{2\Delta} \qquad (7\text{-}3)$$

$$\tan\phi = \frac{l}{r} \qquad (7\text{-}4)$$

$$\phi = \arctan \frac{l}{\frac{L_H^2}{2\Delta}} = \arctan \frac{2l\Delta}{L_H^2} \qquad (7\text{-}5)$$

La última ecuación (7-5) define una ley de control para el vehículo en función de dos parámetros: L_H y Δ, que sitúan el punto objetivo *B*. Para que el robot pueda realizar un seguimiento de la trayectoria, este punto objetivo debe ir variando constantemente. De hecho se impone un valor fijo de L_H y se procede de la siguiente forma (figura 7.8):

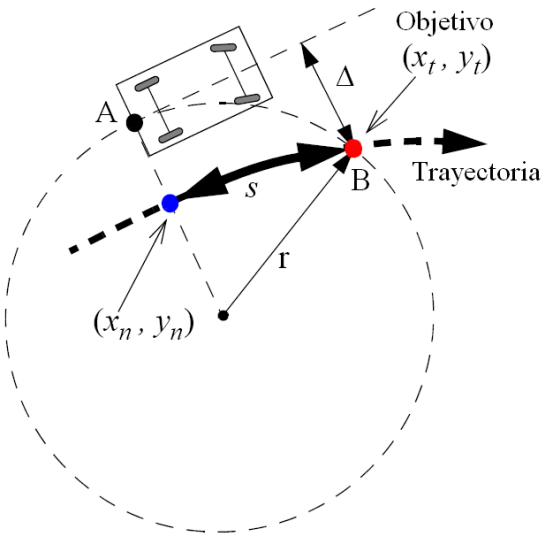


Figura 7.8: Método de seguimiento "Pure Pursuit".

-En primer lugar se determina, a partir de la posición actual *A* del robot, el punto más próximo (x_n,y_n) de la trayectoria a seguir.

- Desde dicho punto, en la dirección de avance de la trayectoria se busca el punto objetivo *B*, utilizando para ello la condición de que diste de *A* una longitud fija L_H.

- Una vez hallado *B*, se determina su desviación con respecto al eje longitudinal del vehículo Δ.

- Con la fórmula (7-5) se procede a calcular el ángulo de giro del volante ϕ y se efectúa el control.

-Pasado un tiempo determinado, el vehículo avanzará en esa trayectoria circular, el punto más próximo será otro y nuevamente se reitera el proceso.

El parámetro de configuración L_H está condicionado por la velocidad y por la naturaleza de la trayectoria. Por ejemplo, cuanto menor sea el radio de curvatura de la trayectoria, menor también ha de ser el valor de L_H, puesto que se desea que el vehículo reaccione lo antes posible a fin de no desviarse mucho del camino a seguir. Por otro lado en maniobras rectas, un valor excesivamente bajo de L_H implica oscilaciones entorno a la recta. Ello es debido a que pequeñas desviaciones del camino implican severos esfuerzos de control y aunque el robot corrige rápidamente el error, cuando lo consigue lo hace divergiendo en orientación con respecto a la recta. Consecuentemente al avanzar volverá a salirse del camino y nuevamente girará en demasía sin alinearse, al menos a corto plazo, con la recta. La velocidad acentúa este comportamiento, con lo que para cada velocidad existe un valor óptimo. En estos experimentos, tanto la velocidad como el valor de L_H han sido ajustados hasta obtener un comportamiento óptimo (Gómez-Bravo y otros, 2009).

7.2.2.2 Seguimiento de maniobras

Hasta aquí el procedimiento descrito corresponde con el de persecución pura tradicional. Sin embargo, la implementación práctica mostró algunas cuestiones que obligaron a modificar el algoritmo. En concreto los problemas más relevantes tenían que ver con los puntos de cambio de sentido conocidos también como “inversores”. La trayectoria puede fragmentarse entre estos inversores en tramos. En estos finales de tramo, el algoritmo de seguimiento tradicional suele imponer un margen de error aceptado. Es decir, si el robot alcanza una posición cuya distancia al inversor es inferior al margen de error prefijado, se detiene, dando por concluido el tramo. Mientras esto no ocurra, el algoritmo de seguimiento continúa funcionando.

Esta técnica tiene la desventaja de presentar comportamientos como el expresado en la figura 7.9. Esta situación anómala sucede siempre que el radio de curvatura mínimo del robot supera al calculado por el algoritmo de persecución pura. Al no poder establecer la curvatura solicitada, el robot mantiene la máxima posible. Con ésta, al avanzar, el robot describe un círculo durante el cual se mantiene la desigualdad entre el radio de curvatura solicitado y el admisible, no llegando jamás a disminuir la distancia con el inversor más allá de un valor dado.

La solución obvia a este problema consiste en aumentar la distancia de presunción de llegada, es decir, ese margen de error asociado al alcance del inversor por parte del robot. Esta medida generaba otros problemas. En primer lugar se aumentaba el error de seguimiento de la trayectoria al no cumplir con los puntos de inversión, y en segundo, este error perjudicaba el seguimiento del tramo siguiente.

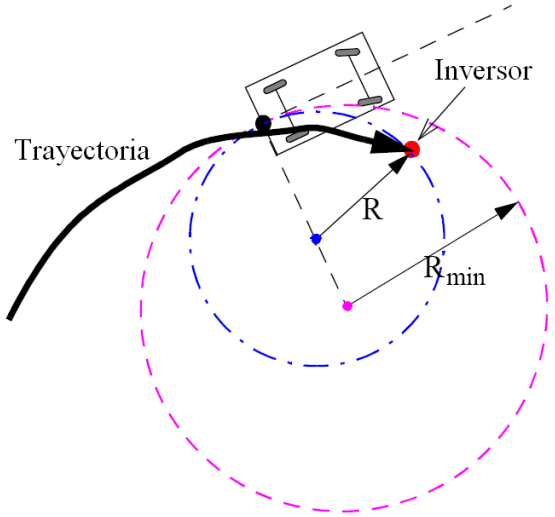


Figura 7.9: Problemas con inversores.

Otros problemas surgidos durante la experimentación revelaban la dependencia de los parámetros Lookahead y velocidad óptimos con respecto al tipo de trayectoria a seguir. Como se ha comentado anteriormente no es lo mismo seguir una curva cerrada, donde son necesarios valores bajos de ambos parámetros, que una trayectorias larga rectilínea, donde sucede lo contrario.

Todos estos aspectos incurrieron en la toma de las siguientes medidas:

1. Secuenciación de maniobras

En lugar de enviar toda la trayectoria al robot, se optó por subdividirla en los puntos de inversión, de modo que el robot sólo reciba un tramo cada vez. Esto permite reajustar parámetros al inicio de cada tramo e incluso replanificar el tramo si el robot se ha desviado demasiado.

2. Parámetros de seguimiento variables

En lugar de establecerlos de modo fijo para toda la trayectoria se realizó una asociación de parámetros Lookahead y velocidad al tipo de maniobra que se estaba

ejecutando. Estos valores quedan reflejados en la tabla 7.1. Ha de tenerse en cuenta que en el caso del parámetro Lookahead, se trata de distancias máximas en el tramo, dado que al acercarse al punto final del fragmento a seguir, este valor queda relegado a la distancia hasta dicho punto.

Tabla 7.1: Valores óptimos de Lookahead y velocidad.

	Lookahead (m.)	Velocidad (m/s)
Rectas	4	1
Curvas	0.8	0.5
Próximo a inversor	distancia al inversor	0.5

3. Criterio de alcance de inversores

Para evitar la caída en trayectorias circulares sin fin, se modificó el algoritmo de seguimiento midiendo además la proyección del segmento $\overline{AB}$ sobre el eje longitudinal del vehículo ($\overline{AB'}$). Si este valor cambiaba de signo (si avanzando pasaba a ser negativo, o retrocediendo a ser positivo), se entendía que el punto ya había sido alcanzado y se solicitaba un nuevo tramo.

Para ilustrar esta nueva técnica se dispone de la figura 7.8. En ésta se representa la posición del vehículo en dos puntos (A y A_n) escogidos dentro de una trayectoria circular. Dicha trayectoria sería la circunferencia de radio de giro mínimo del robot, es decir, aquella en la que quedaría el robot atrapado debido a la incapacidad de alcanzar el punto de inversión. La línea de color morado representa la trayectoria circular practicada por el robot. Por otro lado se ha marcado en color negro, a trazos gruesos, la trayectoria deseada, donde existe un punto de inversión (B). Cuando el vehículo está situado en la posición inicial, la proyección del punto de inversión (B) sobre el eje longitudinal del vehiculo (B') queda delante del mismo, es decir, el segmento $\overline{AB'}$ resulta positivo de acuerdo a las coordenadas locales del robot. Por el contrario, cuando el robot alcanza la posición final mostrada en la figura (A_n), la proyección de B sobre el eje en esta nueva ubicación resulta ser B'_n, y queda detrás del vehículo. En otras palabras, la proyección del segmento $\overline{A_nB}$ sobre el eje del vehículo, $\overline{A_nB_n'}$, es negativa de acuerdo a las coordenadas locales del robot. Así pues, existe un punto intermedio entre A y A_n en el cual se ha producido el cambio de signo. Dado que la evolución del segmento proyectado es continua, esto significa que en el punto de cambio $\overline{AB'}$ ha llegado a ser nulo. O lo que es lo mismo, $\overline{AB}$ ha llegado a ser ortogonal al eje del vehículo. Teniendo

en cuenta que la trayectoria seguida por el robot es circular, esto implica que en el cambio de signo *B* ha llegado a ser el punto más próximo a *A*. Por lo tanto se realiza la detención del robot en el punto de menor distancia al deseado.

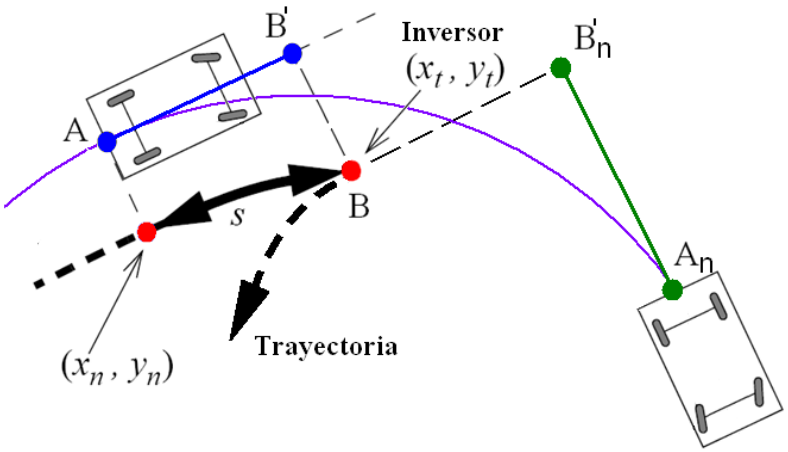


Figura 7.10: Criterio de alcance en los inversores.

Para expresarlo de modo analítico se recurre a las matrices de transformación entre sistemas de referencia (Ollero, 2001). Dados dos sistemas de referencia en el plano (denotados con los superíndices 'L' para el sistema local asociado al robot y 'G' para las coordenadas globales) separados por un vector ***O*** y rotado uno con respecto al otro un ángulo *θ*, cualquier punto del plano podrá expresarse en ambos. Así el punto B en coordenadas locales se puede deducir a partir de la expresión (7-6):

$$\begin{bmatrix} {}^{L}Bx \\ {}^{L}By \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sen\theta & {}^{L}Ox \\ sen\theta & cos\theta & {}^{L}Oy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{G}Bx \\ {}^{G}By \\ 1 \end{bmatrix} \qquad (7\text{-}6)$$

$${}^{L}Bx = {}^{G}Bx \cdot cos\theta - {}^{G}By \cdot sen\theta - {}^{L}Ox < 0 \qquad (7\text{-}7)$$

La condición de parada corresponde a que la primera componente de (7-6) sea negativa, es decir, la inecuación (7-7). También podría haberse impuesto la condición de nulidad de este valor. Sin embargo, las imprecisiones de los sensores y el hecho de que el robot percibe su posición entrecortada debido a los periodos de muestreo, convierten la técnica de la medida del signo como mucho más robusta y práctica que la anterior.

4. Extensión de la trayectoria en los puntos de inversión

Además de lo anterior, el margen de error provocaba deficiencias en la llegada a los puntos de inversión. Sin embargo el seguimiento a los puntos previos era excelente. Así pues se añadieron pequeños segmentos rectos (0,4 metros) unidos a los inversores y alineados con la orientación del vehículo en los mismos. De este modo se establecía un nuevo inversor artificial postergado en la trayectoria lo suficiente como para que cuando la condición de alcance se diera, el robot se hallase sobre el inversor real.

5. Uso de la planificación en línea

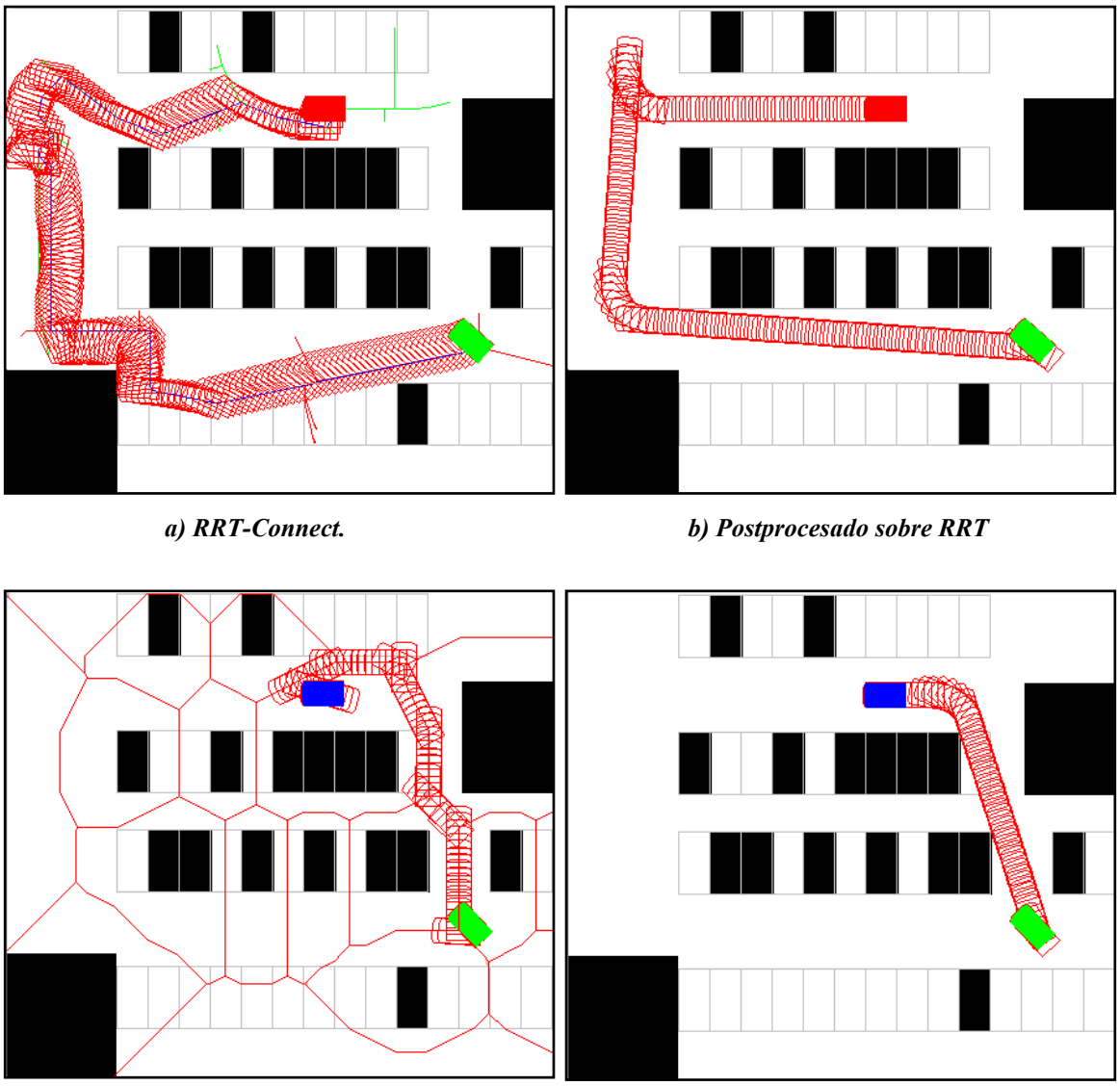
La existencia de un algoritmo planificador capaz de responder en línea, permite incorporar la posibilidad de reacción ante distanciamientos de la trayectoria excesivos. Toda vez que el algoritmo de seguimiento detecta esta eventualidad puede acudir al planificador para que genere un nuevo tramo a partir de la posición actual. Esto supone la detención del vehículo durante un breve segundo, sin embargo, dada la escasa frecuencia de dicha situación y el escaso tiempo de respuesta, apenas se ve perjudicado el comportamiento en comparación con el seguimiento sin incidencias.

7.2.3. Algoritmos de planificación

En estas experiencias se han manejado dos algoritmos de planificación sin restricciones no holónomas: RRT y VODEC. Sobre éstos se ha utilizado el algoritmo de postprocesado explicado en el capítulo 4, que transformaba las trayectorias anteriores en otras admisibles para el robot. Los dos algoritmos de planificación sin restricciones han resultado ser rápidos en comparación con el postprocesado, siendo indistinguibles en este aspecto.

No obstante, la naturaleza aleatoria del RRT derivaba a veces en la generación de trayectorias alejadas de las que utilizaría un ser humano. Como ejemplo obsérvese la figura 7.11. En ella puede apreciarse que el algoritmo RRT (en su variante RRT-Connect) ha encontrado la solución en espacios alejados de las posiciones del robot al inicio y al final de la maniobra (fig. 7.11a). Como consecuencia de ello el postprocesado ofrece una solución de largo recorrido (ver fig. 7.11b). En cambio Vodec proporcionará siempre la ruta más corta dentro del grafo de Voronoi calculado (ver fig. 7.11c). Y el

algoritmo de postprocesado devolverá en este caso la trayectoria que realizaría un conductor humano (fig. 7.11d).



a) RRT-Connect. *b) Postprocesado sobre RRT*

c) Vodec. *d) Postprocesado sobre Vodec*

Figura7.11: Comparativa RRT-Vodec en el escenario de los experimentos.

En estos experimentos los tiempos de ejecución de los algoritmos RRT y Vodec han sido de 142 y 91 ms respectivamente. El postprocesado sobre cada trayectoria ha tomado 1062 ms sobre la del RRT, y 701 ms sobre la de Vodec. Estos tiempos muestran el alto peso del algoritmo de postprocesado sobre el de los otros dos. En efecto, para este escenario concreto sucede que la labor de planificación sin restricciones resulta

rápida en comparación con el proceso de transformación en trayectorias que las cumplan.

En cuanto a la rapidez de Vodec frente a RRT hay que decir que en este caso, debido a los obstáculos y la longitud de la trayectoria, este algoritmo lo ha superado. En general, en la mayoría de los experimentos, los tiempos de ambos quedan muy próximos como para poder aseverar la ventaja de uno frente al otro.

7.3 Experimentación

A continuación se expone una selección de los experimentos efectuados. Éstos pueden dividirse en dos apartados. El primero se centra en el seguimiento de las trayectorias, ilustrando los aspectos técnicos cuya resolución se ha comentado anteriormente: segmentación de las trayectorias, ajuste de Lookahead, tratamiento de inversores, etc. El segundo involucra la red inalámbrica de sensores distribuidos, contempla escenarios dinámicos y muestra los experimentos en los que el robot reaccionaba a dichos cambios.

7.3.1 Experimentos de seguimiento de maniobras.

Dada la diversidad de elementos y la complejidad del sistema, fueron muchos los problemas surgidos, la mayoría de ellos intrascendentes a efectos de documentación en esta tesis. A continuación se exponen aquellos que merecen interés de acuerdo a la justificación de las modificaciones realizadas en el algoritmo de seguimiento.

La ubicación del robot ha sido uno de los problemas más complejos de resolver. El robot dispone de dos fuentes para lograr una estimación de su posición. La primera es el sensor GPS. Se utilizó un GPS cuyo error máximo teórico es de 2 metros. Para atenuar este error se utilizó un radio módem conectado a una estación base encargada de corregir el error percibido desde los satélites. De este modo el GPS, trabajando en modo diferencial, ofrecía mayor exactitud. Además existía una limitada resolución en cuanto a los valores ofrecidos.

Otro de los problemas detectados fue el efecto de los desniveles del terreno. La capacidad de control se ve alterada por la inercia que el robot adopta si el terreno está inclinado. Este efecto es aún más acusado cuando altera la posición de llegada a un inversor. Como muestra de ello véase la figura 7.12, donde la pendiente del suelo incitaba al robot a continuar hacia el lado inferior de la figura.

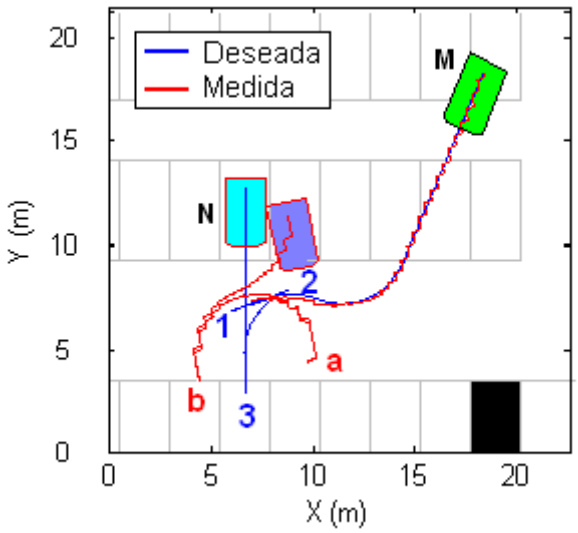


Figura 7.12: Problemas debido a la pendiente del terreno.

En el experimento de la figura 7.12 el robot ha de ir desde la posición marcada en color verde ('M') hasta la azul ('N'), superando tres puntos de inversión muy próximos entre sí (mostrados en la figura como *1, 2* y *3*). El robot realiza un seguimiento bastante preciso de la trayectoria (línea azul para la trayectoria deseada, y línea roja para la realmente seguida según el GPS), hasta que llega al primer punto de inversión (*1*). Debido a la tolerancia establecida en el algoritmo se detiene antes, quedando mal orientado respecto a la alineación ideal que sería la del segmento *1-2*. Por ello al intentar alcanzar el inversor *2* se desvía, y es aquí donde este error es aumentado considerablemente debido a la pendiente del terreno que arrastra al robot hasta el punto *a*. Desde ahí el punto *3* queda casi fuera del círculo de radio de curvatura mínimo. El robot trata de alcanzarlo y lo da por logrado (nuevamente debido al margen de error programado) al llegar al punto *b*. Desde ahí el error sobre la trayectoria a seguir es tan grande que al tratar de corregirlo sobrepasa la línea marcada. Al hacerlo queda mal orientado y precisa de nuevo corregir un error creciente. Si la línea continuase durante

un tramo recto más largo se podrían apreciar oscilaciones. En este experimento la posición final detiene el proceso antes.

Una de las soluciones adoptadas para resolver este problema consistió en la inclusión de unas prolongaciones rectilíneas en los inversores de la trayectoria propuesta. De este modo el robot lograba mantener el margen de tolerancia a la par que aumentaba la precisión en la superación del inversor. Incluso en el caso de que estas prolongaciones fueran excesivas, es decir, provocaran que el robot sobrepasase el punto de inversión, los resultados mejoraban. Como ejemplo de ello puede observarse la figura 7.13, donde se han utilizado estas prolongaciones.

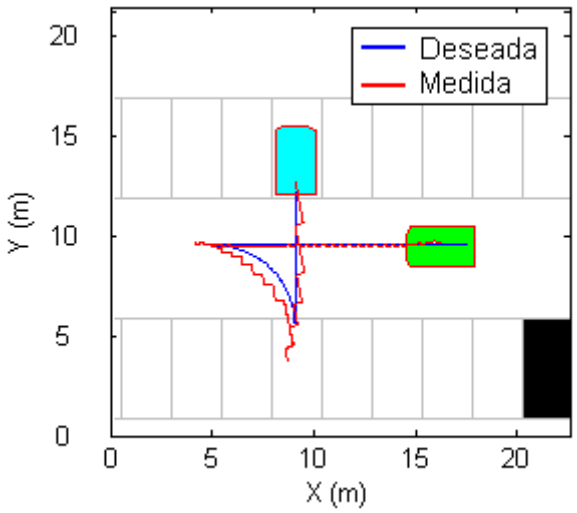
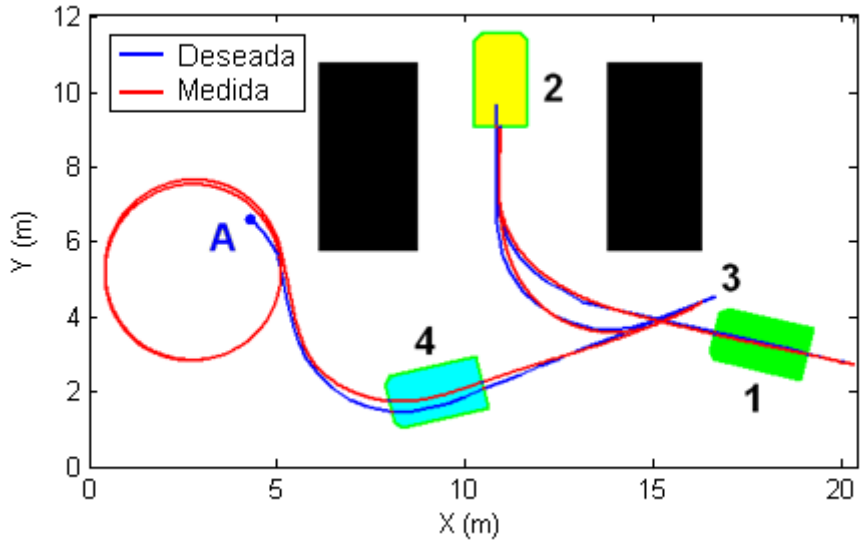


Figura 7.13: Agregación de extensiones en los puntos de inversión.

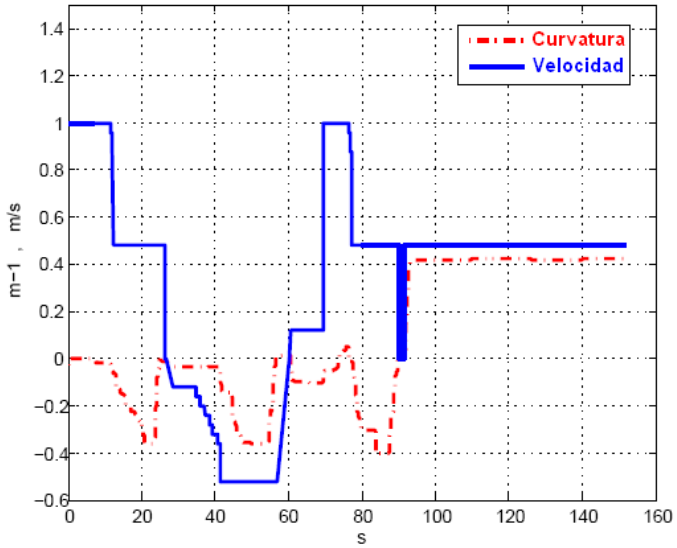
Nuevamente la línea azul indica la trayectoria deseada, donde no se han incluido las prolongaciones. La línea roja refleja la posición obtenida por el sensor GPS. En los dos puntos de inversión, el robot sobrepasa la posición deseada, sin embargo el resultado obtenido es satisfactorio. Ello se debe a la capacidad del robot de seguir un camino si está bien orientado y dispone de recorrido para corregir el error. En cambio, como ocurre en el caso anterior, aunque el robot se encuentre sobre la misma trayectoria, una mala orientación puede afectar de forma muy negativa.

Esta nueva estrategia incita a reducir el margen de error establecido para dar por alcanzado un inversor. No obstante este parámetro es crítico pues puede dar lugar a

trampas circulares como puede observarse en el siguiente experimento cuyos resultados se muestran en la figura 7.14.



a) Ejemplo de trayectoria circular sin fin



b) Curvatura y velocidad asociadas al experimento

Figura 7.14: Problema de seguimiento en inversores en un experimento real.

En la figura 7.14a se refleja la trayectoria seguida (medida según los sensores) del robot en una línea de color rojo, y en azul la trayectoria deseada. A lo largo de la

misma existen dos inversores, identificados con los puntos *2* y *3*, y el final de la trayectoria marcado como punto *A*. El robot viene desde el punto *1*, alcanza el primer inversor (*2*) y retrocede hasta el siguiente (*3*). Obsérvese como el ajuste del margen de tolerancia en la llegada y el uso de las prolongaciones han logrado aquí un seguimiento excelente. Sin embargo, cuando el robot trata de alcanzar el punto *A* falla. Tal y como se explicó en el apartado 7.2.2.2, esto es debido a que el radio de giro requerido para llegar al punto *A* es inferior al mínimo que el robot es capaz de ejecutar. Consecuentemente, el robot gira al máximo el volante y permanece describiendo circunferencias de forma continua sin lograr en ningún momento satisfacer la condición de llegada.

La curvatura está representada en trazos discontinuos de color rojo en la figura 7.14b, donde en torno a los 90 segundos pasa de ser negativa a positiva. Mientras el volante gira intentando efectuar este cambio, el robot se desplaza alejándose de la trayectoria lo suficiente como para no alcanzar nunca el punto *A*.

En este experimento se han incorporado otras mejoras. La continuidad de las líneas sugiere que las posiciones obtenidas ya no corresponden con el muestreo directo del GPS. En efecto, la otra fuente de información consiste en la odometría, es decir, los encoders asociados a las ruedas, y volante de giro. Toda esta información es agregada a la del GPS utilizando un filtro de Kalman extendido. De ahí la suavidad de las líneas. Las partes en las que la trayectoria seguida se aparta de la deseada se debe a varios factores, tales como los deslizamientos en puntos de parada o simplemente debidos a la naturaleza del terreno, la orografía del suelo, etc. En concreto, estos deslizamientos son especialmente perturbadores en lo que a la orientación del robot se refiere, de ahí la inclusión de un giróscopo para mejorar la estimación de posición.

Otra de las mejoras incluidas en este experimento ha sido el ajuste de la velocidad del vehículo de acuerdo al tipo de trayectoria a seguir. En la figura 7.14b, la velocidad del robot está representada por una línea azul continua. Los valores que alcanza este parámetro oscilan entre el metro por segundo de valor máximo, los $\pm$0,5 m/s y los puntos de parada (0 m/s), el resto de la gráfica muestra evoluciones bruscas entre estos valores. De hecho, estos han sido parámetros de diseño en el algoritmo de seguimiento. Es decir, cuando las maniobras son rectilíneas (fragmento inicial en torno al punto *1* y fragmento que parte del inversor *3*) se alcanza la velocidad máxima. Por el contrario cuando se necesita maniobrar en curvas cerradas la velocidad disminuye a los 0,5 m/s, permitiendo un seguimiento más preciso. Las detenciones corresponden a los

puntos de inversión. En efecto el robot comienza a frenar en cuanto la distancia al inversor alcanza un valor dado. El último valor nulo de la gráfica de velocidad muestra que el robot trató de detenerse junto al punto *A*, pero al final lo sobrepasa por no cumplir la condición de parada. Como se comentó en la sección 7.2.2.2 esto se resolvió con una mejora en dicha condición de parada.

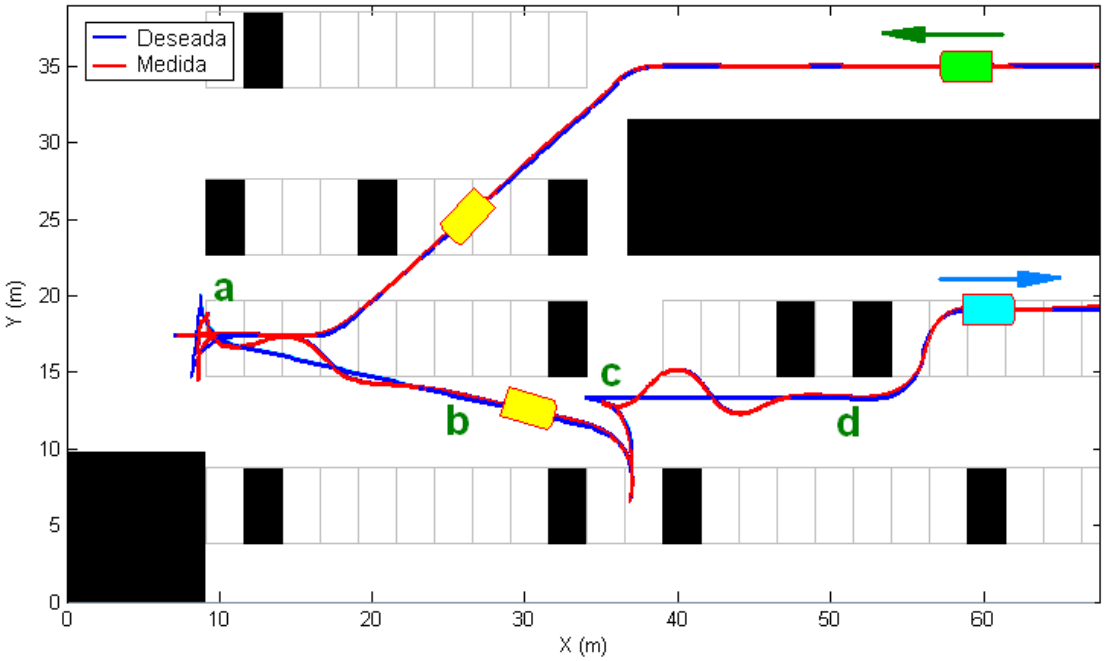
El ajuste de la velocidad, al igual que el ajuste del parámetro Lookahead se realizó en base a una serie de experimentos como los mostrados en la figura 7.15. En la figura 7.15a se refleja un experimento en el que el Lookahead se estableció en dos metros, y la velocidad a 1 m/s. Inicialmente el seguimiento de la trayectoria deseada (línea de color azul) es bastante preciso. La trayectoria realizada por el robot (el color rojo) está casi superpuesta a la deseada hasta que se alcanza la secuencia de inversores (zona en torno al punto *a*). El robot circula demasiado deprisa para jalonar adecuadamente los puntos de inversión. Cuando llega al último (punto *a*), queda mal orientado. Consecuentemente al avanzar cruza la trayectoria deseada y el esfuerzo de control se vuelca en volver al camino marcado. Al ser el Lookahead demasiado corto este esfuerzo implica un giro de volante excesivo, con lo que al alcanzar la trayectoria vuelve a cruzarla. Así se explican las oscilaciones entre los puntos *a* y *b* de la figura 7.15a, que vuelven a repetirse en el tramo entre el punto *c* y el punto *d*.

El la figura 7.15b se ajustó el parámetro Lookahead a una distancia mayor (3 m.), con lo que el esfuerzo de control era más suave y se reducían las oscilaciones. Véase la reducida longitud de los tramos *ab* y *cd* en comparación con los anteriores, así como el distanciamiento máximo con respecto a la trayectoria a seguir.

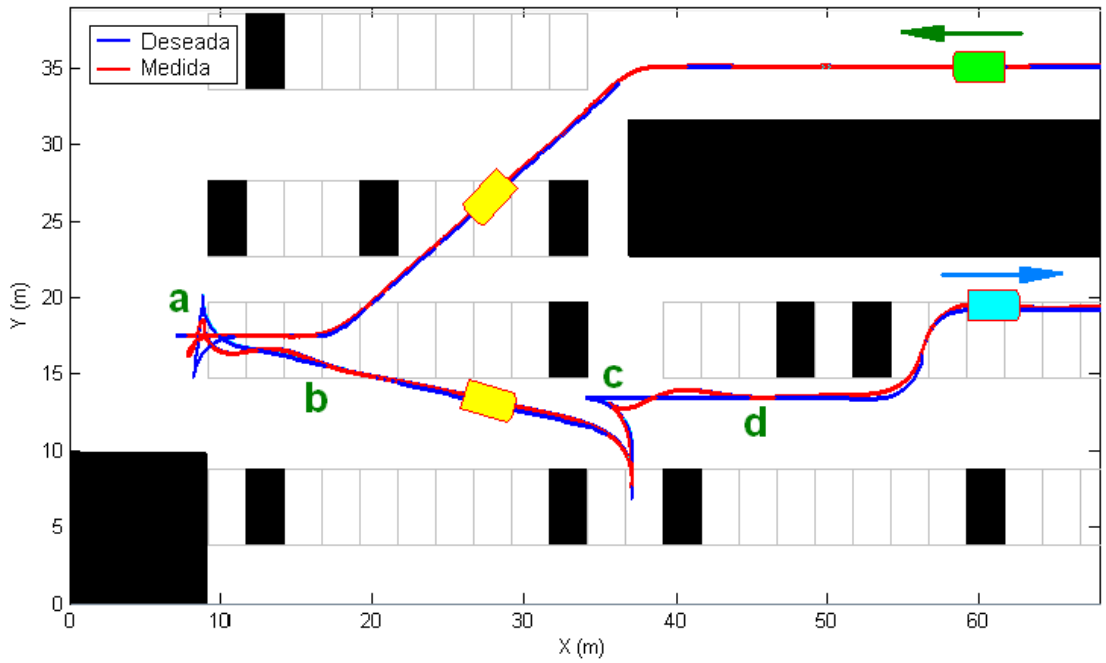
También resulta instructivo observar la sensibilidad del sistema en los inversores. Es el error en la llegada hasta los inversores lo que provoca el inicio de la oscilación. Como se comentó en la sección 7.2.2.2 este problema se trató con dos estrategias: la utilización de prolongaciones rectilíneas al final del segmento y la medición del signo del segmento $\overline{AB}$', donde *A* es el origen de coordenadas del robot y *B'* la proyección del inversor sobre el eje del vehículo.

En el experimento mostrado en la figura siguiente (7.16) se utilizan ambas técnicas, así como la segmentación de la trayectoria y la asignación de valores para los parámetros Lookahead y velocidad de acuerdo a la tabla 7.1. En la figura 7.16a, en línea roja continua se muestra la trayectoria deseada y en línea azul a trazos la ejecutada por

el robot. En la figura 7.16b la línea azul continua se utiliza para la velocidad, y en roja a trazos la curvatura.

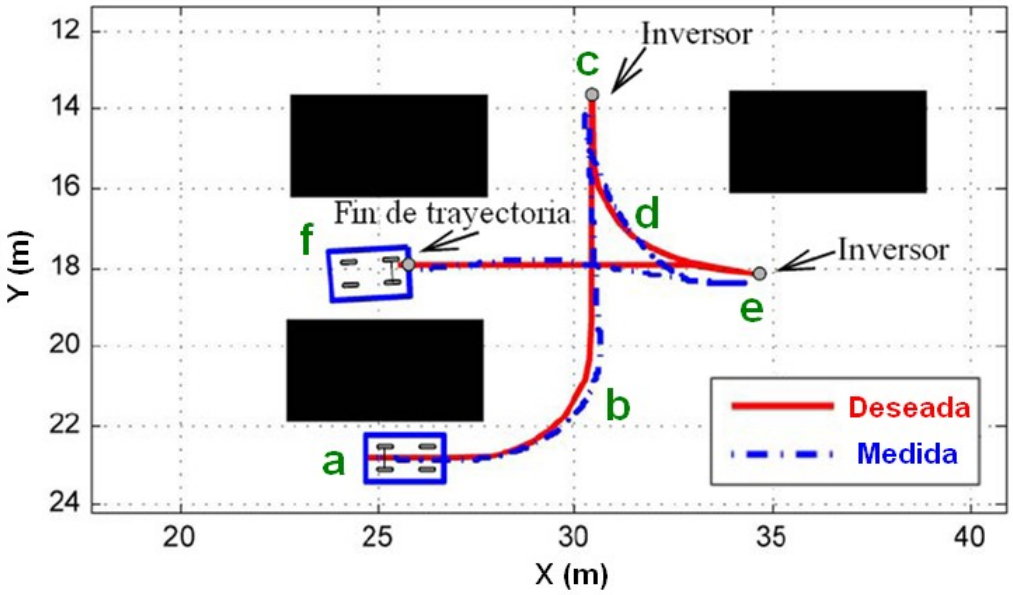


a) Lookahead=2m. Velocidad= 1m/s.

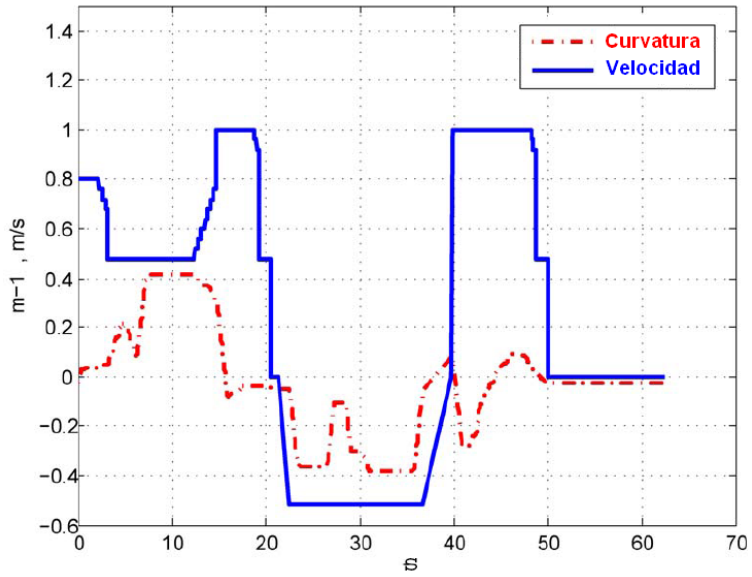


b) Lookahead=2m. Velocidad=0,5m/s

Figura 7.15: Experimentos con diferentes parámetros de Lookahead.



a) Trayectoria deseada y seguida



b) Curvatura y velocidad a lo largo del experimento

Figura 7.16: Trayectoria con dos puntos de inversión

El experimento comienza reduciendo la velocidad para acometer la primera curva. Ésta es ejecutada a velocidad constante de 0,5 m/s y Lookahead de 0,8 m. Le sigue un tramo recto, por lo que el robot acelera hasta alcanzar el m/s y el Lookahead se sitúa en 4 m. Conforme el robot se aproxima al primer inversor, el lookahead queda

reducido a la distancia hasta el inversor. Al acercarse a una determinada distancia, la velocidad se reduce de nuevo a 0,5 m/s, para facilitar la detención. Ésta se produce con bastante proximidad a su objetivo. A continuación tiene lugar el arco hacia atrás. Dado que es una curva cerrada, se utiliza de nuevo la velocidad de 0,5 m/s. Así permanece hasta el segundo inversor. La última parte de la trayectoria es una recta, por lo que se establece el lookahead en 4 m. y la velocidad en 1 m/s. El robot describe una leve oscilación, pero con escaso error sobre la trayectoria deseada. Finalmente, cuando está a punto de detenerse reduce la velocidad a 0,5 m/s durante un segundo, hasta que se para.

En las gráficas el seguimiento resulta excelente, pero no hay que olvidar que la posición obtenida es una estimación de la real, y que existe un error entre una y otra que depende de varios factores. En este caso el terreno presentaba cierta inclinación descendente en torno al punto final. Factor añadido es el hecho de que en este último tramo se circulaba hacia atrás.

Si se observa la figura 7.16a, aparece una oscilación pequeña en torno a la trayectoria deseada. Sin embargo, si se estudia la gráfica de curvatura entre los 40 y los 50 segundos, la variación de curvatura sí que es severa. Estas sobreactuaciones en el control implicaban aumentos de error en la odometría que afectaban consecuentemente a la estimación de la posición. De ahí que el vehículo no quede centrado con mayor precisión en la plaza de aparcamiento libre.

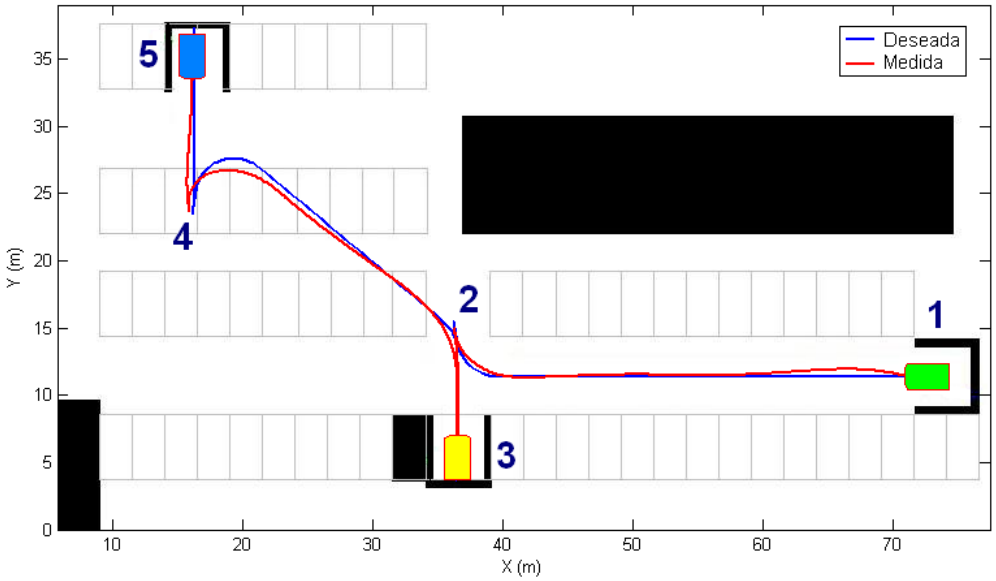
En la figura 7.17 se muestran una serie de fotografías asociadas al experimento anterior. Las letras bajo cada fotografía corresponden con las mostradas en la figura 7.16a. En la última de las fotografías (7.17f) puede observarse la desviación del robot con respecto a la línea media entre los dos vehículos estacionados.

Tras los problemas mencionados, el uso de de las técnicas explicadas y el ajuste de los parámetros de control permitió culminar con éxito este experimento y otros similares en los que el error de posición final era del orden de un metro.

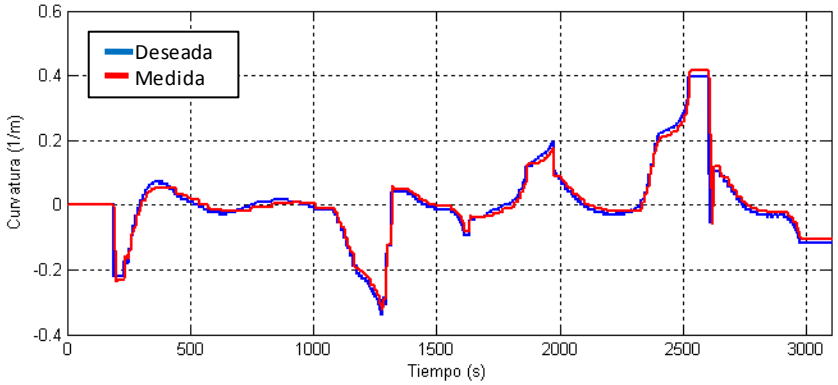
Al aumentar el recorrido de las trayectorias el efecto del error acumulado se hace más severo con lo que la precisión disminuye. Para reducirlos se recurrió a disminuir la velocidad. En el siguiente experimento se ubicaron puntos de parada alejados entre sí cubriendo la extensa área que constituyó el escenario de trabajo. La figura 7.18a los muestra en las posiciones *1*, *3* y *5*.

Figura 7.17: Imágenes del experimento de la figura 7.16.

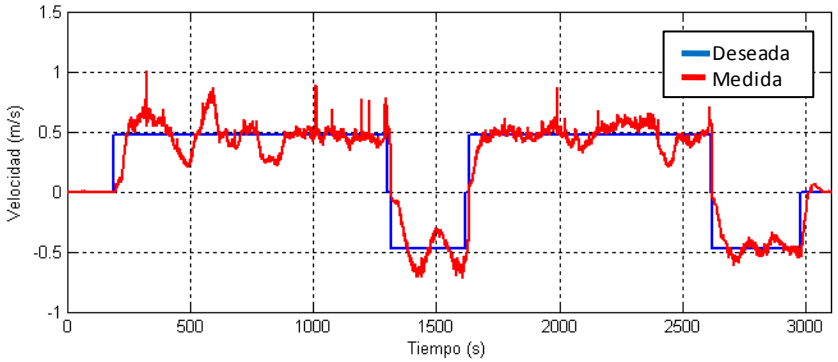
Nuevamente, el seguimiento sobre la posición estimada es razonablemente bueno. El algoritmo de seguimiento solía dar errores máximos en torno a los 0,7 m. A éstos habría que añadir los derivados de la estimación de posición. La figura 7.18b muestra la grafica de curvatura. En ella se puede apreciar la referencia en una línea azul, y la realizada en línea roja superpuesta. En la figura 7.18c puede observarse la velocidad deseada en línea azul y la medida en línea roja. En la primera el seguimiento es muy preciso y sin apenas desviaciones. En la segunda aparece un error de alta frecuencia derivado del proceso de estimación discreto.



a) Trayectorias



b) Curvatura



c) Velocidad

Figura 7.18: Seguimiento de trayectorias de largo recorrido.

Otra observación que se desprende de la gráfica de la velocidad es la diferencia entre los arranques y las paradas. El seguimiento en el arranque es más costoso pues depende de la potencia de los motores del robot. En cambio la parada se efectúa utilizando frenos, con lo que es bastante más brusca.

7.3.2 Método Distribuido de Planificación de Caminos

La interacción entre vehículos autónomos y redes de sensores vía wíreless está suscitando un especial interés en los últimos años (Venkitasubramaniam y Tong, 2004; Kansal y otros, 2004), particularmente en lo que se refiere al guiado de vehículos autónomos mediante este tipo de sensores (Li y otros, 2003; Batalin y otros, 2004; Moore y otros, 2004). Por ejemplo, un robot autónomo podría funcionar como un nodo móvil de la red navegando por el escenario recogiendo y trasportando la información (lo que se conoce como '*Data Mule*') proporcionada por la Red de Sensores Wireless (RSW) (Shah y otros, 2003; Kansal y otros, 2004).

Otra posible aplicación sería la de moverse siguiendo unos puntos determinados en el escenario o incluso navegar por un terreno peligroso ayudado por la información proporcionada por estos mismos sensores (Li y otros, 2003).

Existen varios trabajos que abordan el problema de la navegación de nodos móviles interactuando con una en una red de sensores y la generación de las rutas (Tong y Adireddy, 2003; Venkitasubramaniam y Tong, 2004). En Grandham y otros (2003), y en Wang y otros (2005), se utilizan técnicas de programación lineal con objeto de reducir el consumo de energía y maximizar, por tanto, el tiempo de operación de los sensores. Sin embargo, cuando se imponen restricciones a la movilidad del nodo, es menos frecuente encontrar trabajos que aborden el problema.

En este apartado se aplican las técnicas descritas hasta ahora para controlar el robot Romeo4R teniendo presente la información proporcionada por una RSW, con el objeto de que el robot realice diferentes cometidos. Sean cuales fueren las acciones a realizar se entenderá que para ello el robot ha de ejecutarlas detenido sobre una serie de puntos prefijados. El planificador por tanto deberá resolver la trayectoria entre los mismos.

7.3.2.1 Problema planteado.

Sea un vehículo autónomo (un robot móvil) navegando en un escenario conocido sólo en parte, en el que se ha dispuesto una RSW. También se considera que la evolución dinámica del escenario puede afectar a la conectividad de la red de sensores, pudiendo incluso llegar a perderse debido a diversas perturbaciones o sucesos (perturbaciones electromagnéticas, daños en algunos de los nodos de la red, descarga de las baterías,...). El robot se comporta como un nodo móvil con capacidad de comunicación, navegando siguiendo un camino definido previamente, aunque este camino puede ser modificado en función de los datos proporcionados por los sensores o para restablecer la comunicación en la red, en caso de que esta se perdiera. También se supone que tanto el robot como los sensores pueden ser localizados en un sistema de referencia global.

En principio, se considera conocida la existencia y ubicación de varios obstáculos fijos, y se calcula un determinado camino de forma que el vehículo pueda evitarlos y realizar determinadas tareas asignadas. Por ejemplo, el vehículo autónomo podría funcionar como un '*Data Mule*' recogiendo los datos de los sensores sin conectividad directa con el punto de acceso u otros sensores de la red. Para cumplir este propósito, debe alcanzar varios puntos de control situados entre los sensores. La ubicación de estos puntos puede ser determinada de diversas formas: por un operador humano, en función de unas tareas específicas, por la información sensorial, como consecuencia de algún evento en particular, o incluso determinada de tal forma que el robot pueda desempeñar su labor de '*Data Mule*' como se ha comentado.

Asimismo, las rutas para alcanzar estos puntos de control pueden ser calculadas aplicando diversas técnicas de optimización (Grandham y otros, 2003; Wang y otros, 2005). Sin embargo, ninguno de estos métodos tiene en cuenta las restricciones cinemáticas del vehículo. La consideración de estas restricciones ha sido uno de los ejes de esta tesis. Resulta lógico aplicar las técnicas de planificación descritas a este problema.

Como en los problemas planteados en secciones anteriores, en éste es necesario generar un camino admisible, libre de colisiones, con el que alcanzar los puntos prefijados y modificarlos, si fuese necesario, en función de la información sensorial. A medida que el vehículo se desplaza, los sensores proporcionarán información acerca de nuevos obstáculos cuya ubicación, incluso, podría ir cambiando, o zonas peligrosas que

el robot debe evitar. Adicionalmente, los sensores también pueden determinar nuevos puntos de paso que deben ser incorporados a los establecidos previamente, de forma que el camino original debe ser modificado de acuerdo a esta nueva información, respetando en todo momento las restricciones no holónomas del vehículo.

Por otro lado, los sensores tienen un rango de detección limitado, por lo que la información asociada a un sensor determinado sólo debe afectar a la parte del camino cercana al mismo.

7.3.2.2. Solución propuesta

A raíz de todo lo anterior se ha escogido la siguiente estrategia de planificación. En primer lugar se segmenta la trayectoria solicitada en tramos definidos por los puntos de paso. El planificador (RRT o Vodec seguido de postprocesado) podrá operar entre cada par de puntos de paso consecutivos de la misma forma en que se ha explicado en secciones anteriores. Es decir, un punto de paso actuará como posición origen, y el siguiente como destino. A partir de aquí el planificador genera una trayectoria teniendo en cuenta los obstáculos presentes. En segundo lugar se estudia el escenario, estableciendo las áreas en las que cada sensor tiene influencia. Concretamente lo que interesa es hallar los puntos de paso que limitan los fragmentos de trayectoria que pueden verse afectados por cada sensor. La aplicación desarrollada al efecto, vincula estos sensores a los fragmentos, de modo que cuando un sensor altera su estado el esfuerzo de cómputo se concentra sólo en el fragmento afectado y no en todo el itinerario.

Los eventos que puede generar el sensor son de dos tipos: nuevo punto de paso (la información implica la necesidad de que el robot acuda a un lugar determinado), o nuevo obstáculo (los datos del sensor indican un área que debe ser evitada). En el primer caso, el fragmento ha de ser desechado y sustituido por una nueva trayectoria que alcance el nuevo punto de paso. En el segundo se verifica si el nuevo obstáculo solapa la trayectoria actual. Si esto no ocurre, no habrá colisión, y por tanto no se actúa. Si por el contrario hay intersección, se vuelve a calcular el fragmento afectado.

Nótese la ventaja que este sistema supone para los planificadores en términos de ahorro de cómputo. Si se elige el algoritmo RRT; éste será más rápido cuanto más cercanos sean los puntos de paso a unir. Si además el escenario completo es muy

grande, el RRT será bastante más rápido que otros algoritmos que deban procesarlo en su totalidad. Vodec no se encuentra entre estos últimos, dado que en un escenario dinámico donde los nuevos obstáculos suponen una pequeña área con respecto al total, requieren tan sólo procesar las celdas afectadas. Por lo que resulta igualmente efectivo.

7.3.2.3. Ejemplos de aplicación

Esta sección está dedicada a presentar los resultados de diversos experimentos relacionados con el enfoque propuesto. Las simulaciones se han ejecutado en un PC con un procesador AMD a 1.8Ghz, bajo un entorno de programación JAVA ejecutado sobre el sistema operativo Windows XP.

Se ha considerado un escenario con diversos obstáculos y se han dispuesto varios puntos de paso por el mismo, como se observa en la fig. 7.19, de forma que el vehículo autónomo pueda llevar a cabo diferentes tareas al alcanzarlos.

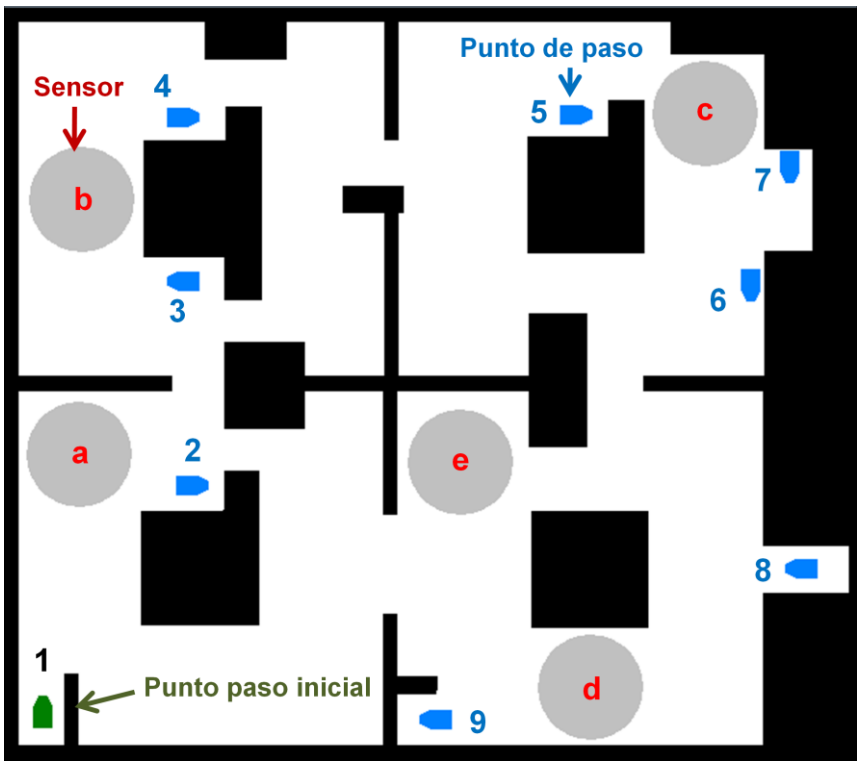


Figura 7.19: Escenario de planificación distribuida.

Cada punto de paso está representado en la figura por una silueta azul que indica la posición y la orientación deseada del vehículo al alcanzar el punto de paso. El número

junto a cada una de ellas indica el orden en que han de ser jalonadas. Por otro lado, se ha dispuesto en el escenario un conjunto de sensores representados por círculos de color gris que indican el rango de detección de los mismos. Cada sensor está asociado con varios de los puntos de paso, esto es, la información de cada sensor influirá en la generación de la sección del camino que une los puntos de paso que están bajo su influencia. Por ejemplo el sensor a está asociado con 1,2 y 3; el sensor b con 2,3 y 4; el c con 5,6 y 7; etc.

La fig. 7.20 muestra los resultados de la planificación realizada a partir de los datos del mapa original sin usar la información proporcionada por los sensores.

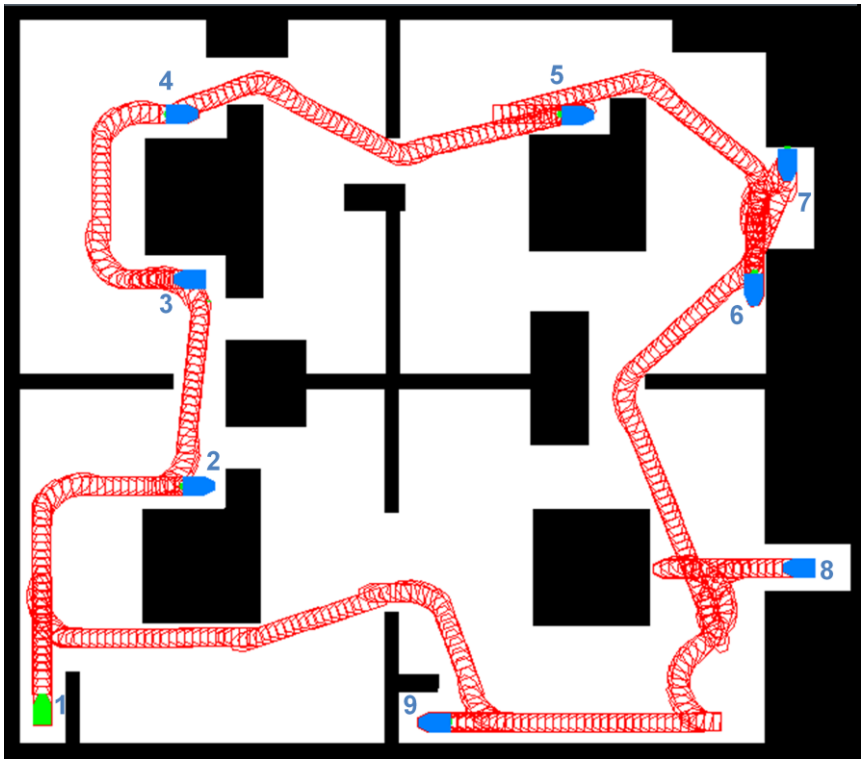


Figura 7.20: Planificación completa sin sensores.

En la fig.7.21 se muestra cómo se ve modificado este camino cuando se atiende la información proporciona por un sensor (el sensor c de la figura 7.19). Queda patente que sólo la sección del camino sobre la que tiene influencia este sensor se ve modificada al detectar el sensor la presencia de un nuevo obstáculo, representado por un círculo de color rojo oscuro. El tiempo necesario para modificar el camino original es de 2.2s.

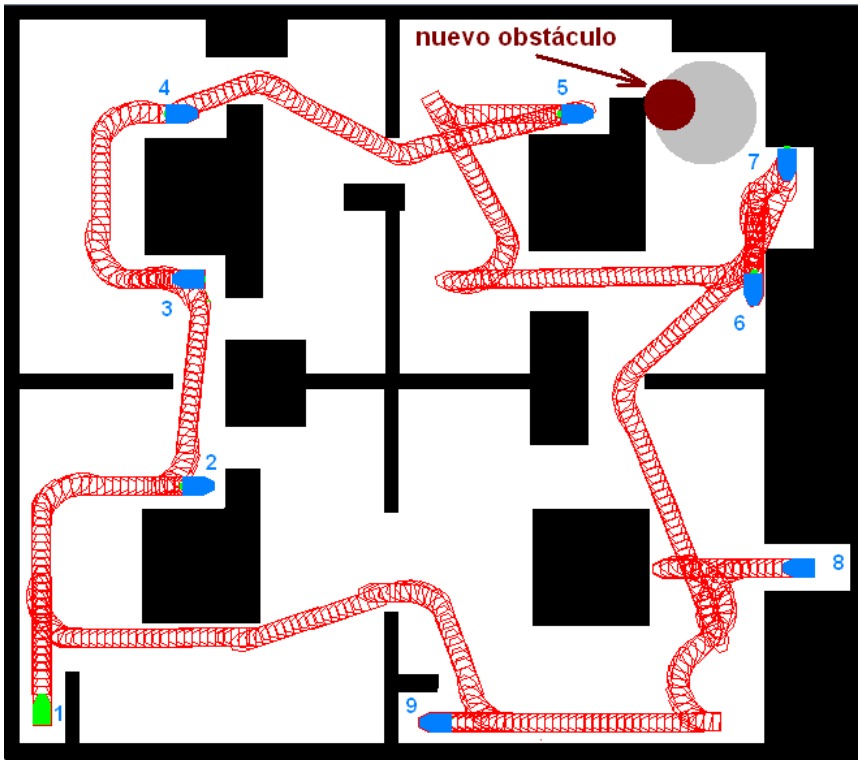


Figura 7.21: Modificación de la trayectoria con un nuevo obstáculo.

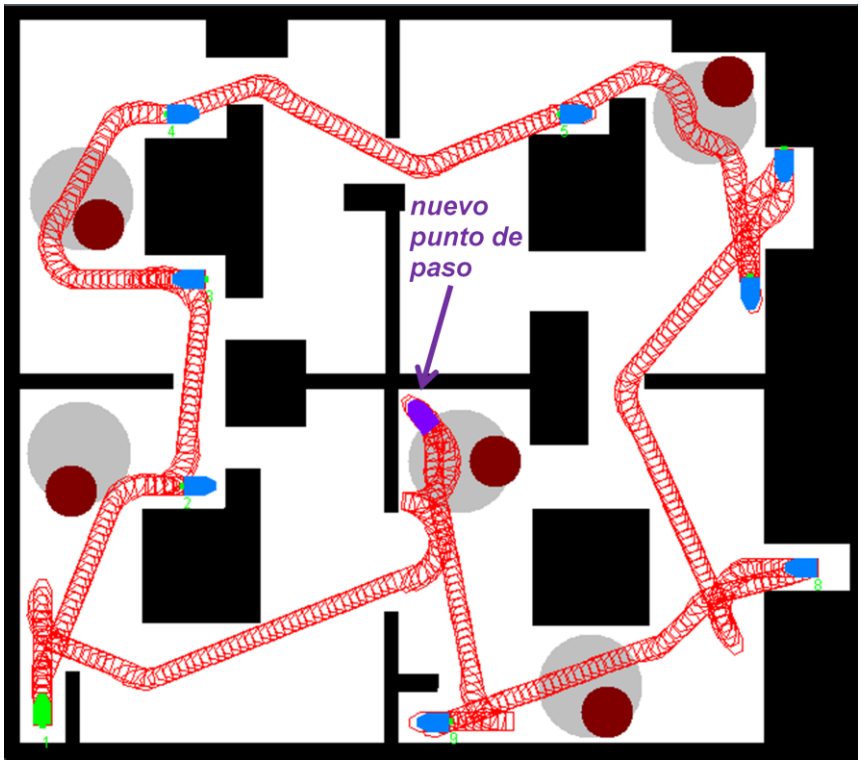


Figura 7.22: Modificación causada por todos los sensores, incluido un nuevo punto de paso.

Finalmente, la Fig. 7.22 muestra los resultados de un experimento en el que se ha considerado la información de todos los sensores. Caso cada sección es calculada independientemente respondiendo al evento del sensor asociado con los puntos de paso incluidos en la sección. Adicionalmente, se generó un nuevo punto de paso (sensor e de la figura 7.19) como consecuencia de un alerta demandando la presencia del robot en ese punto. El tiempo medio para calcular una sección es de 2.5s.

7.3.2.4 Aplicación al robot ROMEO4R

La aplicación práctica de lo anterior utiliza la arquitectura mostrada en la figura 7.23, que supone una ampliación más detallada del esquema presentado en la figura 7.3, sobre todo en lo que a los procesos concierne. El software utilizado ha sufrido algunos cambios con respecto al empleado en los experimentos de la sección 7.3.1.

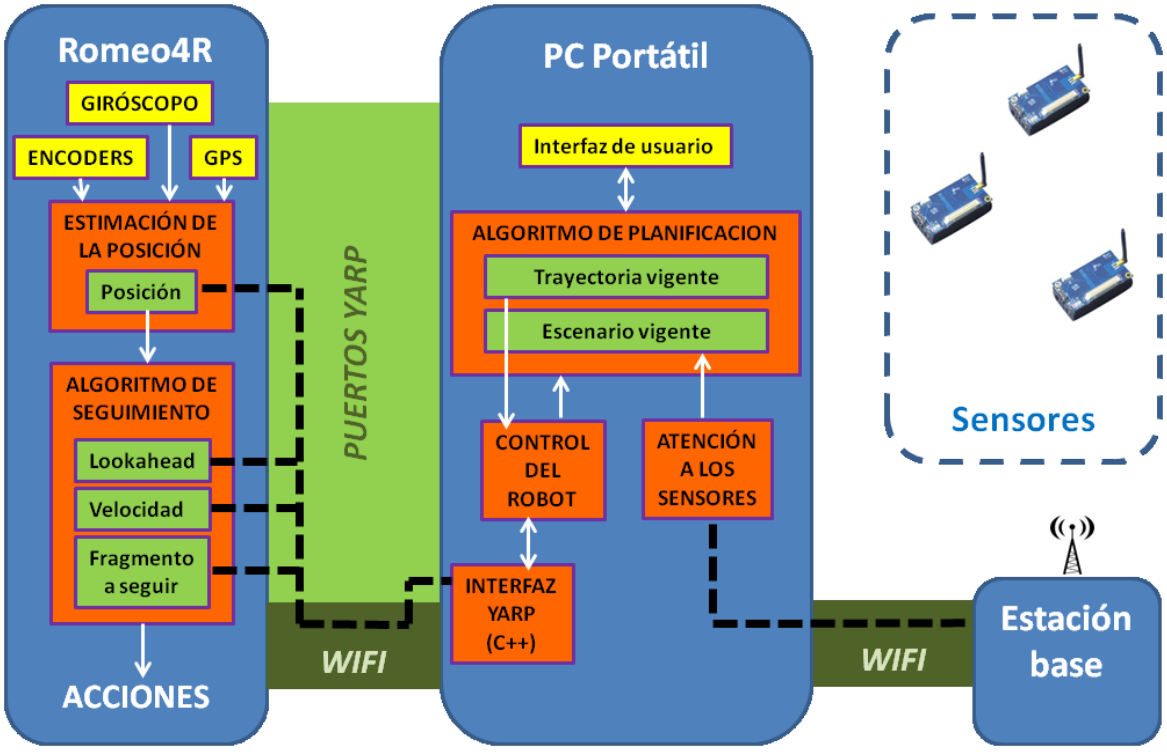


Figura 7.23: Arquitectura de comunicaciones y control con RSW.

En la figura 7.23 se muestran los elementos que entran en juego en estos experimentos: el robot, el PC portátil para tareas de planificación, la estación base y los sensores.

En el robot se sigue un ciclo de control que comienza con la recogida de datos de sus sensores (encoders de ruedas y volante, GPS y giróscopo). Un proceso (todos los procesos están representados en cuadros de color naranja en la figura 7.23) se encarga de estimar la posición utilizando un filtro de Kalman extendido. Continuamente, en función de esta posición y del fragmento de trayectoria a seguir, el algoritmo de seguimiento calcula las acciones a tomar.

La conexión entre el robot y el PC portátil destinado a la planificación se realiza a través de los puertos YARP (ver figura 7.4 para más detalles). En este último, un proceso compilado en C++, implementa las rutinas YARP y abre un socket interno para intermediar entre el robot y el planificador. La información transmitida incluye el fragmento de trayectoria a seguir, parámetros de control (lookahead, velocidad y estado) y la posición estimada. Con esta información trabaja un hilo dedicado al control del robot, cuya tarea principal consiste en proveer de nuevos fragmentos al robot conforme éste los va completando. También evalúa situaciones en las que una nueva planificación pueda ser necesaria, en cuyo caso eleva la petición correspondiente al hilo principal. El programa principal alberga las variables globales compartidas con los otros hilos: trayectoria y escenario vigente. Además implementa la interfaz de usuario, mediante la cual éste puede elegir el escenario, ubicar en éste los obstáculos deseados, configurar las posiciones origen y destino, establecer los puntos de paso, elegir el algoritmo de planificación, determinar una primera trayectoria, verificar la conectividad con el robot y la estación de sensores, y lanzar los procesos.

Entre los cambios efectuados para los siguientes experimentos se encuentra la actualización continua en pantalla del escenario, además de la visualización del seguimiento de las trayectorias. Esto se realiza gracias a un tercer hilo dedicado a la comunicación con la estación base encargada de vigilar el estado de los sensores. Cualquier evento es trasladado al escenario, actualizándolo. Este hilo también comprueba si el evento implica la colisión del vehículo por intersección de un nuevo obstáculo con la trayectoria actual. Sólo en este caso avisa al hilo principal para que tome las medidas pertinentes.

La red de sensores centraliza la información en una estación base, que consiste en un PC portátil provisto de radio módem y que está destinado a monitorizar constantemente el estado de los sensores, compararlos con una tabla e informar de cualquier cambio producido a la aplicación del planificador.

Cuando se detecta un cambio en el escenario a través de los sensores, el planificador actualiza en todo caso la información, pero evita alterar el comportamiento del robot. Estos cambios pueden consistir en la ocupación de una plaza de aparcamiento o en su abandono. Lo último no es relevante para el planificador, pues no impide el seguimiento de la trayectoria actual. Sin embargo la ocupación de nuevas plazas sí puede bloquear el paso del robot. Sólo en este caso la aplicación alterará la trayectoria. Para ello en primer lugar determina el tramo interrumpido. Si es un tramo que aún no se ha comenzado a recorrer, vuelve a planificar dicho tramo sin interactuar con el robot. Si por el contrario se trata del tramo actual en curso, el planificador detiene el robot, vuelve a calcular la trayectoria a partir de la posición actual y reanuda la marcha. Esto sucede en tiempos próximos al segundo, muy pequeños en comparación con el tiempo de ejecución de un tramo.

En la figura 7.24 se muestra el escenario utilizado, la ubicación de los sensores en color verde y los puntos de paso en color azul. En concreto se han utilizado tres sensores, designados en la figura con las letras A, B, y C; y se han establecido tres puntos de paso señalados con su correspondiente número de orden en la secuencia de tránsito del robot.

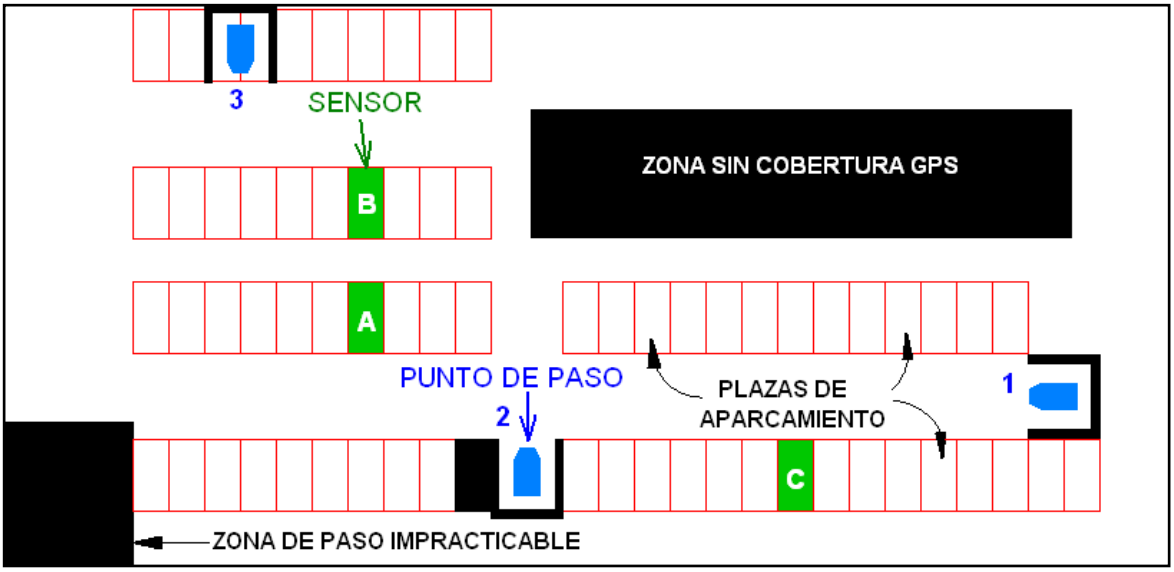


Figura 7.24: Escenario utilizado

En la figura 7.25 se representa una trayectoria ofrecida por el planificador cuando los sensores aún no han entrado en juego. Es decir, las plazas vigiladas por los sensores están libres, y sólo hay algunas ocupadas. En amarillo se muestra la huella de la

trayectoria deseada. El robot debe comenzar en la posición ‘1’, avanzar hasta el inversor ‘2’ y retroceder hasta alcanzar el punto de paso ‘3’. En este lugar volverá a cambiar el sentido de la marcha para llegar al inversor 4, desde donde retrocederá hasta la posición 5. Para cerrar el ciclo el robot debe volver a la posición inicial ‘1’ pasando por el último inversor, el número ‘6’.

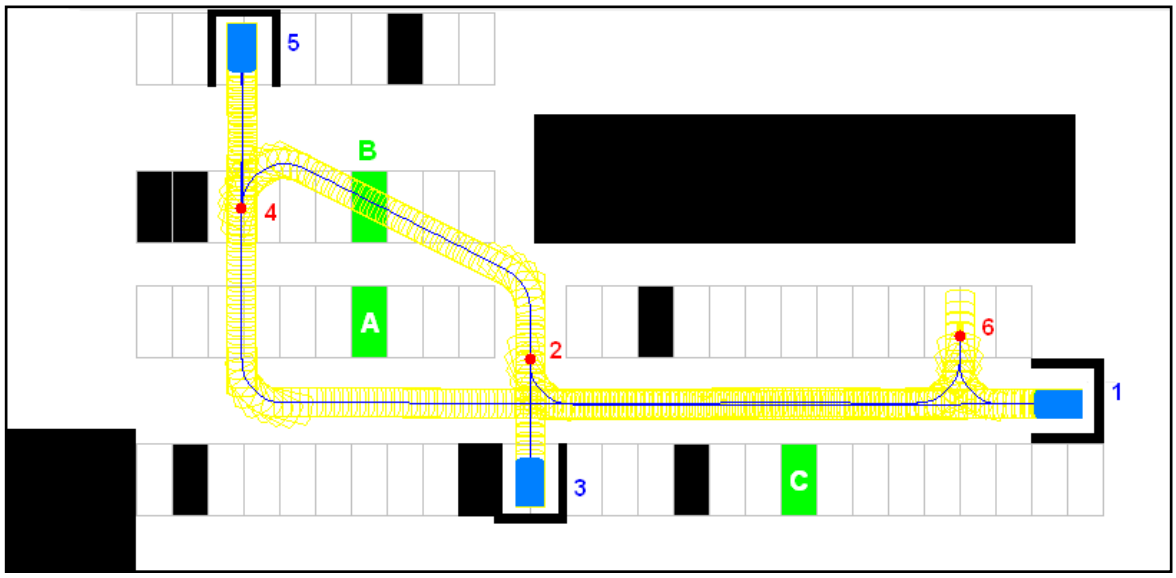


Figura 7.25: Planificación de trayectoria entre puntos de paso sin obstáculos en los sensores.

Visto el escenario y la trayectoria deseada se procede a realizar diferentes experimentos según el comportamiento de los sensores. En el primero se activan sensores que no interrumpen el camino del robot (sensores A y C) con lo que la trayectoria a seguir es la original mostrada en 7.25, el resultado se observa en la fig. 7.26. En el segundo experimento se activa un sensor que sí la interrumpe (sensor B). Según el planificador elegido, Vodec o RRT se tienen dos respuestas diferentes (figuras 7.27 para el algoritmo VODEC y 7.29 para el algoritmo RRT).

En la figura 7.26 se muestra el resultado del experimento realizado con el robot sobre la trayectoria prevista, donde el planificador no altera la trayectoria de acuerdo a la información de los sensores. En este caso los sensores A y C han detectado la presencia de vehículos estacionados mientras que B indica que esta plaza está libre. La línea azul indica la trayectoria deseada y la roja señala la posición estimada del robot. Como puede apreciarse el seguimiento de la trayectoria se hace de forma bastante precisa y permite ubicar al robot en los distintos puntos de paso.

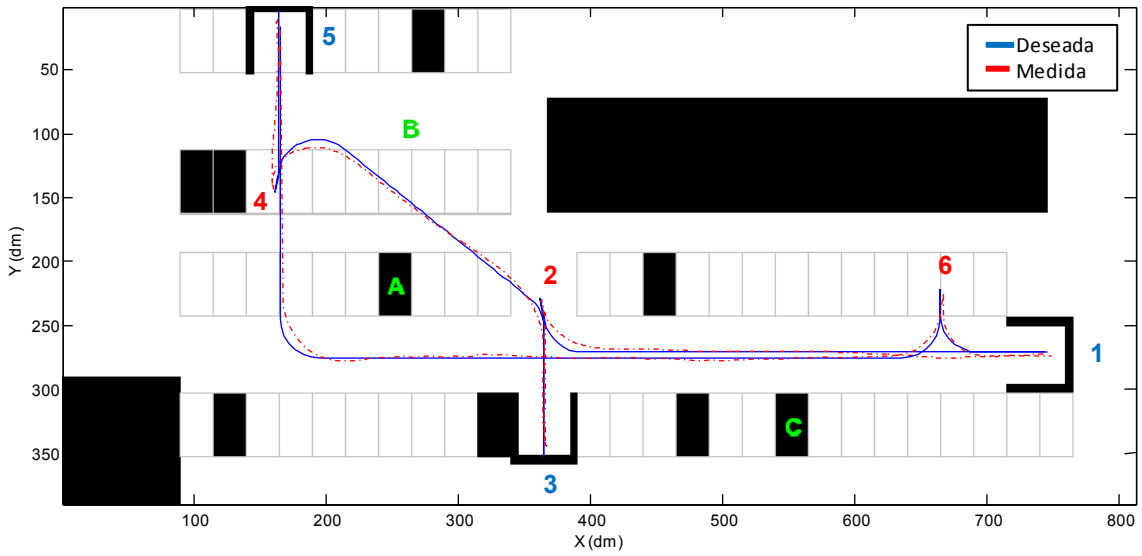
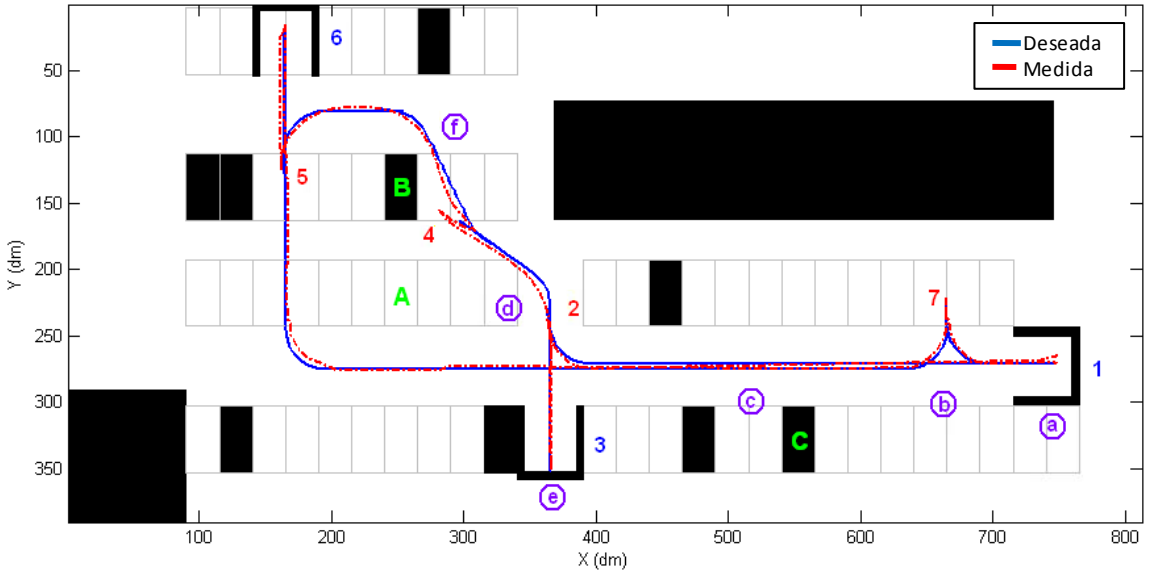
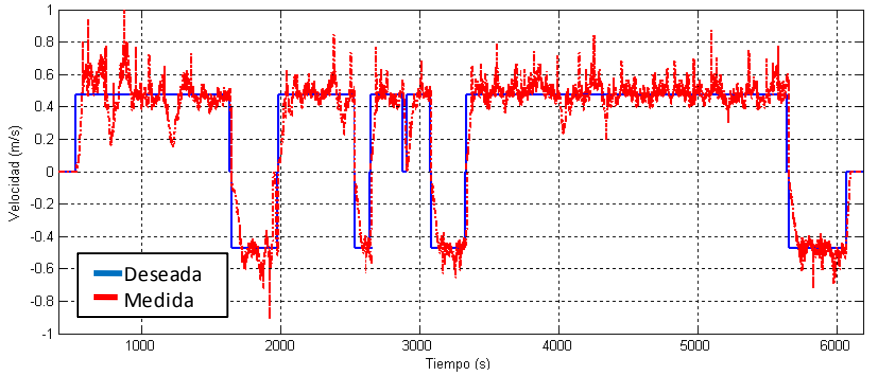


Figura 7.26: Planificación considerando los obstáculos en sensores A y D.

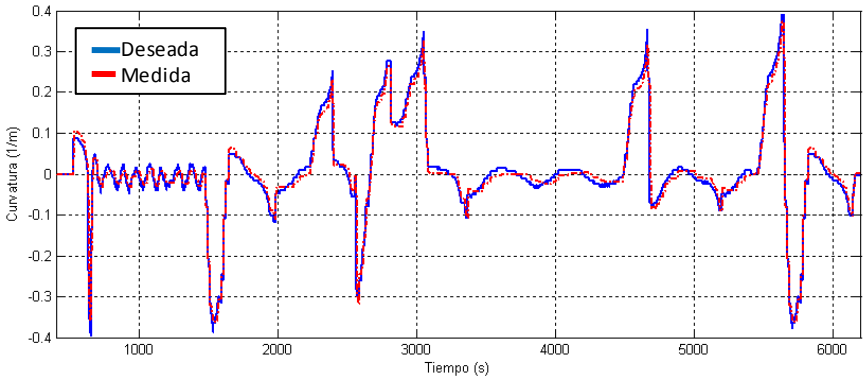
La figura 7.27a representa el resultado cuando un nuevo obstáculo ocluye la trayectoria. La figura 7.28 expone fotografías tomadas de este experimento en distintos puntos. Las letras que identifican cada fotografía de la figura 7.28 corresponden con las posiciones marcadas con las mismas letras rodeadas con un círculo en la figura 7.27a. Todos los inversores y puntos de paso están numerados en el orden en que fueron jalonados. En color rojo se indican los inversores derivados de la planificación. En color azul los puntos paso. Las letras mayúsculas representan las plazas monitorizadas por sensores. Nuevamente la línea azul representa la trayectoria deseada y la roja la seguida de acuerdo a la información obtenida por los sensores. Este experimento comenzó igual que el anterior, partiendo del punto '1'. La figura 7.28a muestra una fotografía del robot en esta posición. El robot inicia la marcha hacia el inversor '2' (fotografías 7.28b, 7.28c y 7.28d). Se detiene y retrocede hasta el punto de paso '3' (fotografía 7.28e). Vuelve a cambiar el sentido de la marcha e inicia su recorrido hacia el inversor '5'. Recuérdese que hasta este momento el sensor 'A' está activado (hay un vehículo estacionado en su plaza) y el 'B' no; por tanto la trayectoria a seguir es la que se muestra en la figura 7.26a. Cuando el robot está en la posición '4', el vehículo estacionado en la plaza 'A' se marcha para ocupar la plaza 'B'. La red de sensores informa de la eventualidad y el planificador detiene al robot. La replanificación dura apenas un segundo. El resultado es una nueva trayectoria que evita el nuevo obstáculo. En la figura 7.27c se muestra la gráfica de la curvatura del vehículo, donde se puede apreciar la variación continua y el escaso error en el seguimiento de la consigna.



a) Trayectoria proporcionada por Vodec.

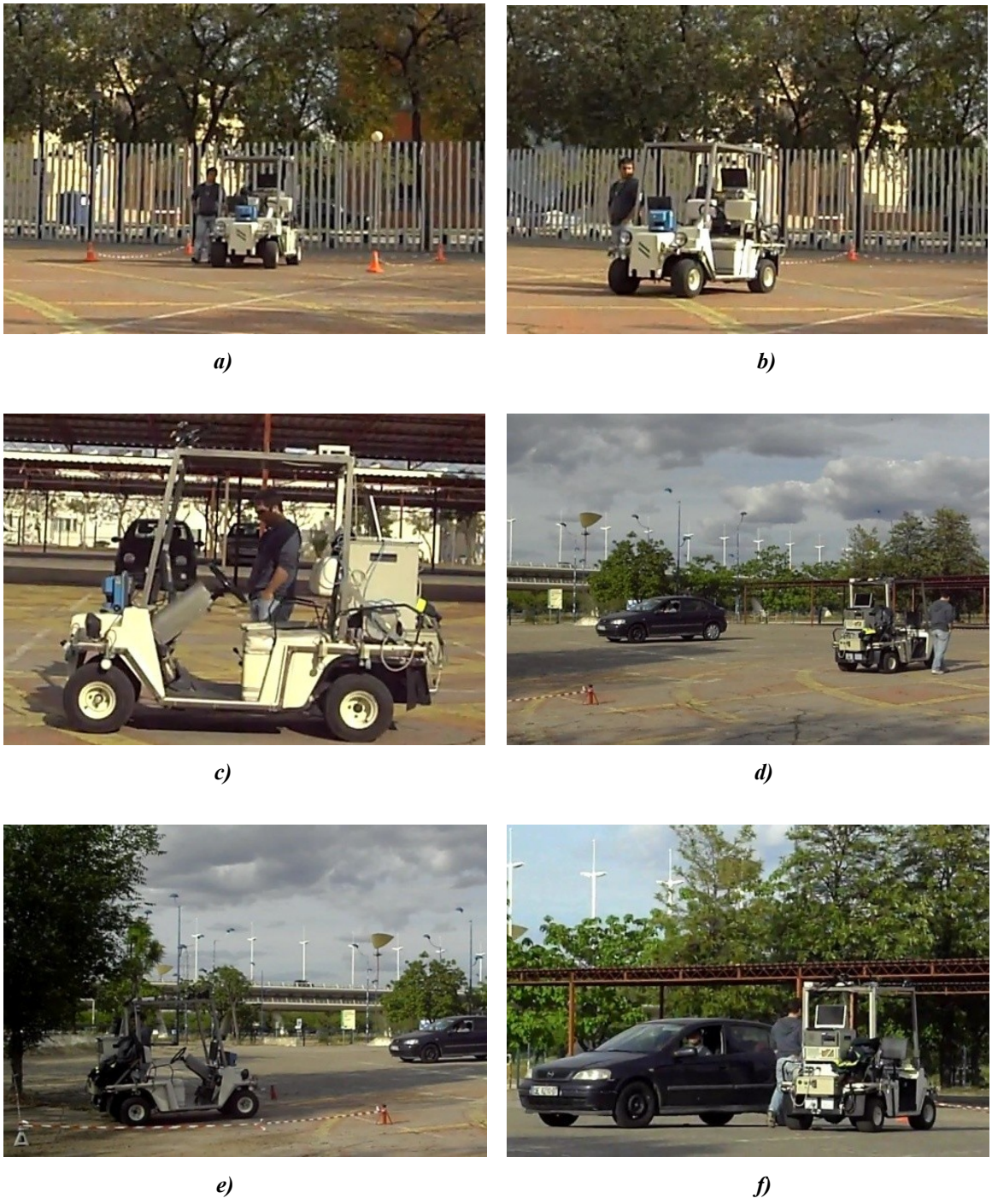


b) Velocidad



c) Curvatura

Figura 7.27: Experimento realizado con el algoritmo Vodec.



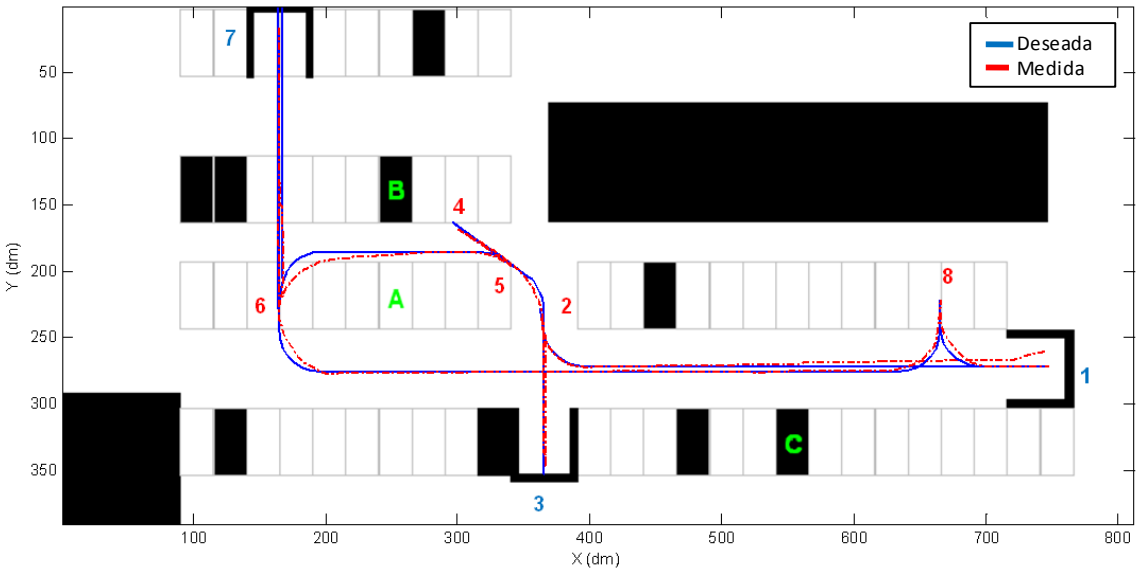
a) *b)*

c) *d)*

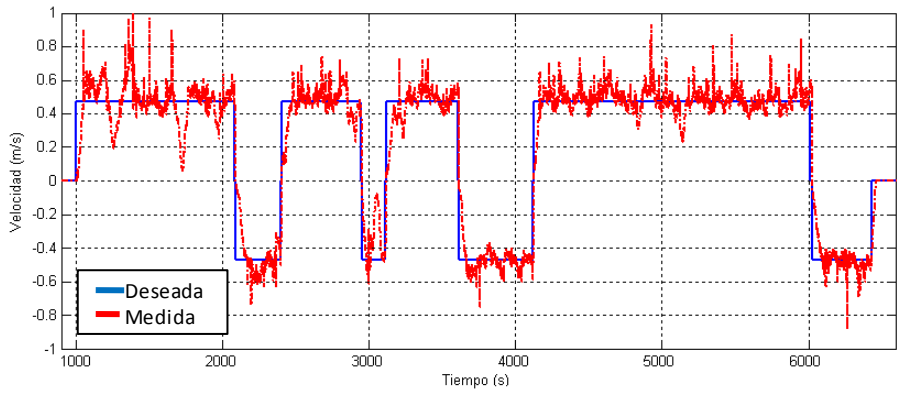
e) *f)*

Figura 7.28: Imágenes correspondientes al experimento representado en la figura 7.27a.

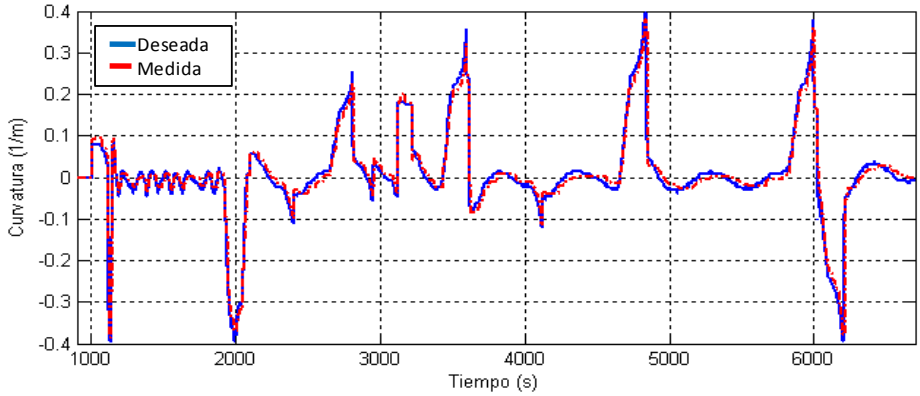
Esta trayectoria y su replanificación fueron ejecutadas con el método Vodec. Tal y como se explicó en el apartado 7.2.3, este procedimiento selecciona la ruta más corta. El RRT, en cambio, debido a su naturaleza aleatoria puede proporcionar otras. Es el caso del experimento descrito en la figura 7.29.



a) Trayectoria ofrecida por el algoritmo RRT.



b) Velocidad



c) Curvatura

Figura 7.29: Experimento realizado con el algoritmo RRT

En el caso de la figura 7.29 el planificador resuelve la conexión desde el punto de parada ‘4’ hasta el punto de paso ‘7’ rodeando el obstáculo ‘B’ por el camino más largo. Así, el robot vuelve hasta el punto ‘5’ y cambia el sentido para llegar al punto ‘6’. En este nuevo inversor se detiene para alcanzar marcha atrás el punto de paso ‘7’. Desde el punto de paso ‘7’ hasta el ‘1’, la vuelta es muy similar a los experimentos precedentes.

En cuanto a la gráfica de velocidad (7.29b), nuevamente las oscilaciones que aparecen en la velocidad seguida (línea roja) respecto a la velocidad deseada (línea azul) hay que atribuirlas al procedimiento mediante el cual ésta es estimada. Los encoders asociados a cada rueda proporcionan un valor angular en cada instante de muestreo con una resolución limitada. La velocidad se estima en función de estas lecturas. Así, dada la naturaleza discreta de los encoders y tiempos se produce un error de estimación que aparece reflejado como un ruido de alta frecuencia.

No sucede lo mismo con la gráfica de curvatura (fig. 7.29c). En efecto el robot alteraba continuamente la posición del volante para ajustarse a la trayectoria solicitada. La gráfica muestra además la proximidad entre la curvatura deseada y la medida. Por tanto se puede extrapolar que este sensor es más preciso y ofrece más confianza que los anteriores.

7.4 Conclusiones

En este capítulo se han presentado los resultados obtenidos tras la experimentación práctica con un robot real de los diferentes sistemas de planificación desarrollados a lo largo de la tesis. Para ello se ha realizado un extenso trabajo de integración de diferentes sistemas (robot, planificador, red de sensores inalámbricos, estación base), usando recursos informáticos en diferentes lenguajes (C++ y JAVA) y diferentes sistemas operativos (Linux y Windows). Si bien esta diversidad ha supuesto trabas y retos a superar, el principal obstáculo ha sido la imprecisión de los sensores. Como se ha comentado anteriormente, sólo el GPS podía mostrar variaciones en la posición de casi 5 metros. Las gráficas de velocidad muestran la severa imprecisión de los encoders de las ruedas. La odometría por sí sola resulta incierta a medida que las trayectorias se alargan. Por tanto la localización del robot ha sido el aspecto que más ha limitado la calidad de los experimentos. A pesar de estas severas limitaciones técnicas,

en todos estos experimentos se logró ubicar el robot dentro del espacio equivalente a dos plazas de aparcamiento, tal y como se muestra en las figuras.

En este sentido ha sido un éxito comprobar que los algoritmos descritos (RRT y Vodec) son capaces de realizar la planificación en línea. En cuanto al seguimiento, las mejoras realizadas y sobre todo el cuidadoso ajuste de la llegada a los inversores han permitido una elevada proximidad entre la trayectoria solicitada y la posición estimada.

CAPÍTULO 8 Conclusiones y futuros trabajos

8.1 Introducción

En este capítulo se procederá a comentar las conclusiones relativas a los principales temas tratados en esta tesis: el algoritmo RRT, el método Vodec, las maniobras restringidas y la evaluación de trayectorias. Además se añadirán apreciaciones sobre la potencialidad de las técnicas empleadas y se introducirán las líneas de investigación abiertas y motivo de trabajos a realizar en un futuro próximo.

8.2 Algoritmo RRT

El algoritmo RRT ha demostrado ser un método sencillo, de fácil implementación, inmune al número de vértices de los obstáculos, y no precisa su engrosamiento. Como

planificador sin restricciones resulta bastante eficaz, sobre todo en conexiones entre configuraciones próximas.

Cuando se tratan robots con restricciones no holónomas, resulta preferible usar el RRT combinado con las maniobras restringidas, en vez de utilizar la adaptación del RRT con las “bang-motions”, como se ha constatado en el apartado 5.1.2. Una idea que surge de forma natural aquí es sustituir las “bang-motions” por maniobras restringidas. Por un lado sería más costoso desde el punto de vista computacional, pero por otro se incorporarían maniobras completas, con lo que la trayectoria final previsiblemente sería menos entrecortada, es decir, más factible. Quedaría por ver su calidad final y su velocidad en comparación con el método utilizado en esta tesis.

8.3 Maniobras Restringidas

Las aportaciones aquí presentadas se refieren a sólo dos casos por lo demás bastante habituales como son el robot diferencial y el Ackerman. Sin embargo la misma filosofía puede ser aplicada a otros robots con restricciones. Bastaría desarrollar un conjunto apropiado de maniobras restringidas y aplicar un postprocesado similar.

No obstante, hasta ahora sólo se pueden utilizar maniobras restringidas sobre robots con restricciones cinemáticas. Queda por estudiar la posibilidad de integrar restricciones dinámicas. Para ello habría que construir un conjunto de maniobras restringidas variando las velocidades de entrada y de salida. Por ejemplo podrían utilizarse tres velocidades: la de marcha o velocidad de trabajo cuando no hay obstáculos próximos, una velocidad reducida de maniobra, y nula para las inversiones. De este modo el conjunto de maniobras no sería excesivo y se podría contender con las restricciones dinámicas. Este salto cualitativo facultaría al algoritmo para abordar tareas de planificación en aeronaves, barcos e incluso tratar problemas poco estudiados como el derrape en vehículos rodados. En efecto bastaría con definir la maniobra restringida de derrape basándose en datos experimentales y a partir de ahí incluirla como una más en el algoritmo de postprocesado. Esta es otra línea abierta de investigación futura.

8.4 Método Vodec

VODEC ha sido un producto generado por la necesidad de un planificador específico para su uso combinado con las maniobras restringidas. La combinación de VODEC con el postprocesado resulta ser un efectivo método de planificación para robots no holónomos con prestaciones superiores en según qué casos a otros métodos.

Aisladamente VODEC presenta características interesantes como diagrama de Voronoi que deben ser estudiadas. En particular sería necesario desarrollar una comparativa con otros métodos, ponderando sus ventajas e inconvenientes. La principal diferencia consiste en la información que VODEC almacena en las funciones de flanco, cuando los demás métodos sólo devuelven el grafo de Voronoi. De este modo VODEC dispone de información métrica redundante que puede ser utilizada para diversos fines. En concreto, en el apartado 3.5 se ha estudiado su explotación en escenarios con obstáculos dinámicos. Quedaría por ver su comparativa con otras estructuras de datos como las mostradas en Guibas y otros, (1992), y Roos, (1993).

Otra de las características propias de VODEC es su capacidad para limitar el cómputo del diagrama de Voronoi a una región del escenario. Esta característica ha sido utilizada en el apartado 3.4 para ahorrar esfuerzo de cálculo restringiéndolo a las celdas directamente implicadas en la trayectoria. Es posible que existan otras aplicaciones de esta interesante propiedad. Por ejemplo, un trabajo futuro consistirá en la implementación de VODEC sobre un algoritmo capaz de procesar las celdas de forma paralela. Hoy en día las tarjetas gráficas cuentan con matrices de varios centenares de procesadores diseñados para trabajar en paralelo. Por ello son relevantes las publicaciones que aprovechan esta característica. VODEC, por su naturaleza intrínseca, resulta idóneo para ello. Sólo queda estudiar el modo en el que son redefinidas las funciones de flanco, única estructura de datos que puede verse afectada simultáneamente por el procesado de dos celdas diferentes.

Por otro lado, es muy común encontrar escenarios con obstáculos rectangulares, o cuyos segmentos sean paralelos a los ejes x e y del plano. Estos escenarios son especialmente adecuados para la descomposición vertical y por lo tanto para VODEC, ya que precisan un menor número de celdas. Así, aparcamientos, almacenes, plantas

industriales, e innumerables entornos suelen estar compuestos por obstáculos de estas características debido al uso de corredores ortogonales.

Los diagramas de Voronoi en tres o más dimensiones han sido objeto de investigación en fechas recientes (Vleugels y Overmars, 1995; Kenneth y otros, 1999; Maurer y Raghavan, 2003). VODEC puede generalizarse a la tercera dimensión utilizando el lema 1 extendido. Es decir, en lugar de la recta x=m se tendría el plano asociado a dicha condición. Entre dos puntos separados por este plano podría establecerse la misma descomposición métrica: la distancia de un punto hasta su proyección sobre el plano y la distancia entre éste último y aquél situado más allá del plano. De este modo se puede independizar la distancia entre puntos interiores a una celda de tres dimensiones y el resto del escenario a través de funciones definidas sobre los planos ortogonales que la delimitan.

El modo de utilizar esta propiedad varía y son objeto de estudio en la actualidad. El más obvio consiste en descomponer el escenario en celdas definidas por planos ortogonales (paralelos al plano *xy* y *xz* por ejemplo) y los propios obstáculos. Nuevamente serían los vértices quienes definirían la ubicación de dichos planos, sobre los que se definirían las funciones de distancia análogas a las funciones de flanco.

Otro modo sería reduciendo el problema 3D a varios 2D cortando el volumen considerado en diferentes planos paralelos a *xy* cuya coordenada *z* sea igual a la de cada uno de los vértices de obstáculo. No bastaría con ejecutar VODEC sobre las secciones detectadas en estos planos, sino que habría que considerar unas “sombras” o figuras adicionales debidas a la proximidad de las caras de los obstáculos situadas entre los diferentes planos. Estas sombras se proyectarían desde unos planos a otros y se propagarían al estilo de las invasiones de flanco.

Otro aspecto digno de estudio consiste en optimizar el postprocesado utilizando para ello la información de proximidad ofrecida por VODEC. En efecto, la trayectoria ofrecida por VODEC es una línea poligonal en cuyos vértices puede adjuntarse un valor métrico, es decir, la distancia de ese nodo al obstáculo más cercano. De este modo puede construirse una función que proporciona la separación de los obstáculos a partir de la posición de la trayectoria.

La experiencia muestra que el algoritmo de postprocesado suele encontrar maniobras cuando tiene espacio para ello, sin embargo debe segmentar en secuencias

de maniobras cortas cuando encuentra pasos estrechos. Si a priori el algoritmo de postprocesado puede contar con esta información puede elegir de modo más eficiente los tramos de búsqueda logrando previsiblemente trayectorias más suaves y en un tiempo de cómputo inferior.

También es posible tratar de suavizar la línea poligonal que genera VODEC directamente con el uso de clotoides (trayectorias cuya curvatura evoluciona de forma lineal), si no completamente, sí al menos allí donde sea posible, dejando sólo los pasos más abruptos para las maniobras restringidas.

8.5 Evaluación y optimización

Tal y como se anunciaba en la introducción, el problema planteado consiste en lograr una trayectoria entre dos puntos de un escenario poblado de obstáculos y con un vehículo sujeto a restricciones. En el capítulo 6 se muestran los estudios realizados con el objetivo de aproximarse a la solución óptima de este problema. La primera dificultad consiste en definir una función de coste, para lo cual se han utilizado técnicas de decisión multicriterio y etiquetas lingüísticas para la definición de los umbrales de preferencia. Con estos recursos se facilita el traslado del conocimiento experto en la valoración de las trayectorias. Estas valoraciones se realizan a posteriori, una vez calculada la trayectoria, con lo cual es posible comparar trayectorias entre sí y decidir cuál es la mejor de ellas.

Tal y como se muestra en el apartado 6.3, podría obtenerse una aproximación a la solución óptima si se utiliza un algoritmo genético adecuado con una amplia población y un número elevado de iteraciones. O bien, se podría intentar realizar un muestreo sistemático del espacio de soluciones. Esta tentativa sería ardua, implicaría una discretización del espacio y estaría limitada en cuanto al número de inversores (cada inversión aumenta la dimensión del espacio de soluciones). Lo cual no es práctico como método de planificación pero permitiría aproximarse con cierta garantía a la solución óptima. Comparando el resultado, considerado ideal, con el devuelto por el postprocesado, podría mejorarse este último. Esta idea surge de la experiencia remodelando continuamente dicho algoritmo y sus efectos sobre la solución obtenida. Por ejemplo, resulta crucial la secuencia en la cual se van comprobando las maniobras

en orden a lograr una trayectoria suave, sencilla y próxima a la que un ser humano escogería realizar. Así mismo, existen otros parámetros susceptibles de variación como los puntos de trayectoria a unir, la posibilidad de simultanear soluciones utilizando diferentes series de búsqueda, el postprocesado recurrente, etc.

Existen muchas posibilidades basadas en recursos sencillos que pueden dar buenos resultados. Por ejemplo, para escenarios altamente poblados de obstáculos que exijan secuencias de maniobras muy específicas, es decir, soluciones difíciles de hallar, se ha utilizado la técnica de robots de planta creciente. Es decir, cuando no se ha encontrado solución con la planta del robot deseada, se ha calculado una planta reducida del mismo, decrementando los valores de sus vértices en un valor dado. Con dicha planta se ha buscado una solución sobre el mismo escenario. Dado que los polígonos son más pequeños, la probabilidad de hallar una solución aumenta. Una vez hallada, dicha solución sirve de partida para realizar otra vez el postprocesado pero ahora con el robot real. Dado que la solución de partida se encuentra muy próxima a la solicitada, el postprocesado halla con facilidad las correcciones necesarias y permite lograr una solución.

8.6 Futuras líneas de investigación

A continuación se resumen en una lista las líneas de investigación abiertas y desarrolladas en los apartados precedentes:

1. Extensión de las maniobras restringidas a otros tipos de robots e incorporación de restricciones dinámicas.

2. Implementación de VODEC en sistemas de proceso paralelo.

3. Generalización de VODEC a escenarios 3D.

4. Explotación de la información métrica de VODEC en el postprocesado.

5. Suavizado con curvas clotoides.

6. Exploración exhaustiva del espacio de soluciones para mejorar el postprocesado.

APÉNDICE A El Planificador

El trabajo de investigación se ha realizado a través del desarrollo de esta aplicación. Se trata de un planificador de itinerarios de robots móviles que integra distintos tipos de algoritmos además de distintos tipos de postprocesado. Dado el interés en la difusión de resultados se ha implementado bajo java, garantizando la portabilidad de esta aplicación y previendo en un futuro próximo su integración dentro de un portal web, de modo que cualquier investigador pueda comprobar vía telemática los logros obtenidos. La figura A.1 muestra la ventana principal de la aplicación.

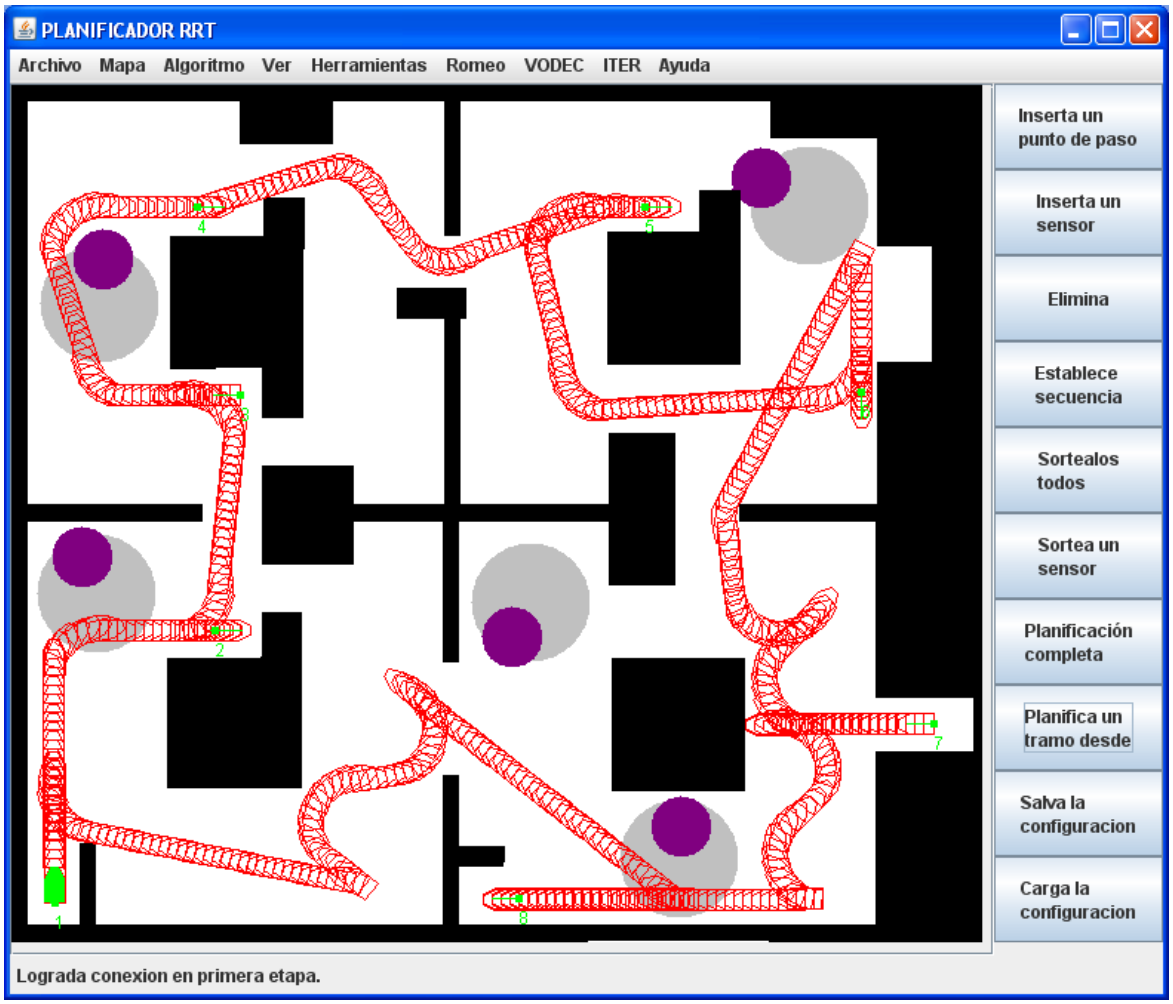


Figura A.1: Ventana principal de la aplicación

Todas las ilustraciones contenidas en este documento relativas al RRT han sido elaboradas con esta aplicación, que incluye entre sus funciones los siguientes algoritmos:

- RRT en todas las variantes aquí comentadas (RRT-Ext-Ext, RRT-Ext-Con, etc.) para vehículos holónomos.

- RRT Ext-Ext para vehículos tipo “Ackerman” y diferencial.

- Descomposición Vertical en celdas.

- Diagramas de visibilidad.

- VODEC.

- Postprocesado para vehículos holónomos, no holónomos basado en maniobras restringidas para el tipo Ackerman y Diferencial.

- Cálculo de datos métricos para comparación de trayectorias.

- RRT-Distribuido para escenario con sensores.

- Planificación en línea con comunicación hacia el robot y hacia los sensores.

En cuanto al entorno de trabajo se ha ido ampliando a medida que los experimentos han ido requiriendo nuevas utilidades. En concreto, para la creación y edición de los escenarios se han aportado las siguientes herramientas:

- Definición de los obstáculos mediante sus vértices.

- Definición de los obstáculos mediante la función de arrastrar y soltar del ratón (rectángulos llenos o vacíos para modelar los obstáculos).

- Incorporación de escenarios en formato BMP, JPG o PNG desde un archivo.

- Incorporación de obstáculos en forma de archivo TXT con la lista de sus vértices.

- Generación automática de obstáculos basada en número y posición aleatoria de vértices sobre una corona circular dada.

- Conversión de obstáculos descritos como matrices BMP a polígonos definidos por sus vértices.

- Capacidad de ubicar sensores y puntos de paso con el ratón, así como su orientación, y de eliminarlos de forma selectiva.

La aplicación permite además importar y exportar configuraciones y datos (como los árboles obtenidos, las secuencias de maniobras, las trayectorias en sí, y otros). En la figura A.2 se muestra la ventana de configuración general, donde es posible importar la planta de un robot e indicar sus parámetros a efectos de planificación, tales como la definición de un margen de seguridad en torno a de dicha planta (en color rojo), el radio de curvatura mínimo, etc.

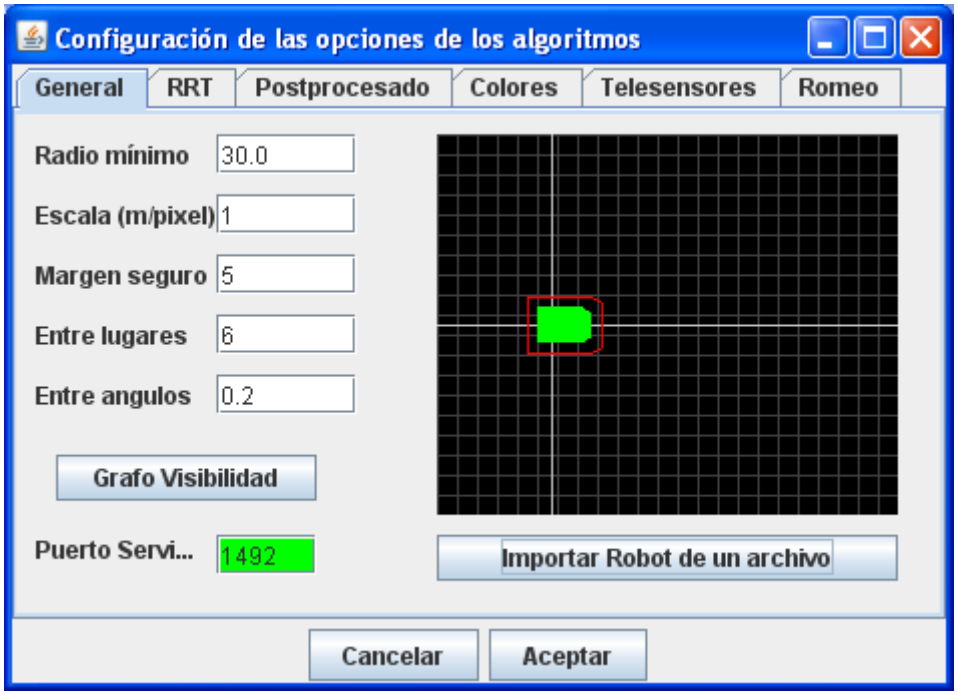


Figura A.2: Menú de opciones de la aplicación

Se ha realizado especial hincapié en la valoración de las trayectorias obtenidas, ofreciendo datos como radios de curvatura, longitud total, número de cambios de sentido, etc.

Los requisitos de ciertos experimentos han propiciado el diseño de menús específicos para ellos, como es el caso del proyecto ITER o el control del robot Romeo4R.

En conjunto se trata de una aplicación que agrupa diferentes técnicas de planificación, con un interfaz de usuario gráfico sencillo y amigable, y que atesora la mayoría de las aportaciones científicas realizadas en esta tesis. Está prevista su implementación futura en un applet de JAVA para poder ser ejecutada en una página web.

Referencias

[1]. Ahuja, N., Chien, R.T., Yen, R., Bridwell, N: (1980) “Interference detection and collision avoidance among three dimensional objects”. Proceedings of the First AAAI Conference, Stanford, 44-48.

[2]. Aichholzer, O., Aurenhammer, F. (1995) “Straight skeletons for general polygonal figures”. Technical Report 432, Inst. for Theor. Comput. Sci., Graz Univ. Of Technology, Graz, Austria.

[3]. Amato, N.M., Yan Wu. (1996) “A randomized roadmap method for path and manipulation planning”.

[4]. Angelier, P. (2002) “Algorithmique des graphes de visibilité”. Tesis doctoral.

[5]. Aurenhammer, F. (1991) “Voronoi diagrams: A survey of fundamental geometric data structure”. ACM Comput. Surv.,nº 23: pags 345-405

[6]. Barba-Romero, S., Pomerol, J. (2000) “Multicriterion decision in management: Principles and practice”. Kluwer Academic Publishers, Dordrecht/Boston/London.

[7]. Barraquand, J., Latombe, J.C. (1989) “Robot motion planning: a distributed representation approach”, Report Nº STAN-CS-89-1257, Department of Computer Science, Stanford University.

[8]. Batalin, M.G., Sukhatme, S. y Hatting. M., (2004) “Mobile robot navigation using a sensor network,” in Proc. of the 2004 IEEE Intl. Conference on Robotics & Automation, April, pp. 636–641.

[9]. Bernard, R. (1996) “Multicriteria Methodology for Decision Analysis”, Kluwer Academic Publishers.

[10]. Boada, I., Coll, N., Sellares, J.A. (2002) “Hierarchical Planar Voronoi Diagram Approximations” Proceedings of the 14th Canadian Conference on Computational Geometry.

[11]. Boissonnat, J.D., Devillers, O., Schott, R., Teillaud, M., Yvinec, M. (1992) “Applications of random sampling to on-line algorithms in computational geometry”. Discrete Comput. Geom., 8:51-71.

[12]. Brans, J., Vincke, P. (1985) “A preference ranking organization method, the PROMETHEE method”. *Management Science*, 31, 647-656.

[13]. Brassard, G., Bratley, P. (1996) “Fundamentals of Algoritmics”. Prentice Hall.

[14]. Bruce, J., Veloso, M. (2002) "Real-Time Randomized Path Planning for Robot Navigation", Computer Science Department, Carnegie Mellon University.

[15]. Campion, G., Bastin, G., D'Andrea-Novel, B. (1996) "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots". IEEE Transactions on Robotics and Automation, vol. 12.

[16]. Canny, J., Donald, B.R. (1988) "Simplified Voronoi diagrams". Discrete Comput.Geom., 3:219-236.

[17]. Carbone, G., Ceccarelli, M., Oliveira, P.J., Saramago, S.F.P., Carvalho, J.C.M. (2008) "An Optimum Path Planning of CaPaMan (Cassino Parallel Manipulator) by Using Inverse Dynamics", Robotica: An International Journal, Vol.26, N.2, pp.229-239.

[18]. Chazelle, B. (1987) "Aproximation and Decomposition of Shapes" in Algorithmic and Geometric Aspects of Robotics, Lawrence Erlbaum Associates, Hillsdale, NJ, 145-185.

[19]. Cuesta, F., Gómez-Bravo, F., Ollero, A. (2004) "Parking Manoeuvres of Industrial-Like Electrical Vehicles With and Without Trailer". *IEEE Trans. on Industrial Electronics.* Vol. 51 nº 2- pp. 257-269.

[20]. Daily, R., Bevly, D. (2004) "The use of gps for vehicle stability control". IEEE Trans. on Ind. Electron, pages 270–277.

[21]. De Berg, M., Matousek, J., Schwarzkopf, O. (1993) "Piecewise linear paths among convex obstacles". In Proc. 25th Annu. ACM Sympos. Theory Comput., pages 505-514.

[22]. De Berg, van Kreveld, M., Overmars, M., Schwarzkopf, O. (2000) "Computational Geometry: Algorithms and Applications", 2nd Ed. Springer-Verlag, Berlin.

[23]. Dubins, L. E. (1957) "On curves of minimal length with a constraint on average curvatureand with prescribed initial and terminal positions and tangents", Amer. J. Math., 79, 497-516.

[24]. Fortune, S. Wilfong, G. (1988) "Planning Constrained Motion". Proccedings of the Fourth ACM Symposium on Computation Geometry, 445-459.

[25]. Fortune, S. (1987) "A sweepline algorithm for Voronoi diagrams". Algorithmica, 2:153-174.

[26]. Fujimura, K., Samet, H. (1989) "A hierarchical strategy for path planning among moving obstacles", IEEE Transactions on Robotics and Automation, 5(1), 61-69.

[27]. Garrido, S., Moreno, L., Blanco, D., Muñoz, M.L. (2007a) "Sensor-based global planning for mobile robot navigation". Robotica: pags. 1 of 11. Cambridge University Press.

[28]. Garrido, S., Moreno, L., Blanco, D., Martín, F. (2007b) "Exploratory Navigation based on Voronoi Transform and Fast Marching". IEEE International Symposium on Intelligent Signal Processing.

[29]. Geldermann, J., Spengler, T., Rentz, O. (2000) “Fuzzy outranking for environmental assessment. Case study: iron and stell making industry”, Fuzzy sets and systems, volume 115, pages 45-65

[30]. Goldberg, D.E. (1989) “Genetic Algorithms in Search Optimisation, and Machine Learning”. Addison-Wesley, Reading, MA.

[31]. Gómez-Bravo, F. (2001a) “Planificación de maniobras en sistemas robóticos no holónomos. Aplicación en robots móviles”. Tesis doctoral. Universidad de Sevilla.

[32]. Gómez-Bravo, F., Cuesta, F., Ollero, A. (2001b) “Parallel and diagonal parking in nonholonomic autonomous vehicles”. Engineering Aplication of Artificial Inteligence. Vol 14 No. 1, pp 419-434.

[33]. Gómez-Bravo, F., Martín, J.M., López, D., Polo, M.P., del Toro, M., Cortés, E., Fortes, J.C. (2006) “Aplicación De Técnicas De Decisión Multicriterio A La Planificación De Maniobras En Robots No Holónomos”. II Simposium de Control Inteligente CEA-IFAC.

[34]. Gomez-Bravo, F., Lopez, D., Ollero, A., Cuesta, F., (2007a) “RRT-D: A motion planning approach for autonomous vehicles based on wireless sensor network information”. 6th IFAC Symposium on Intelligent AutonomousVehicles.

[35]. Gomez-Bravo, F., Ollero, A., Cuesta, F., Lopez, D. (2007b) “A New Approach for Car-Like Robots Manoeuvering.” Robótica - 7th Conference on Mobile Robots and Competitions. Paderne (Albufeira), Portugal. Centro Ciência Viva Do Algarve.

[36]. Gomez-Bravo, F., Ollero, A., Lopez, D., Cuesta, F., del Toro, M. (2007c) “Rrt-D: Planificación Distribuida de Caminos Basada en la Información de una Red de Sensores wireless”. XXVIII Jornadas de Automática

[37]. Gomez-Bravo, F., Ollero, A., Cuesta, F., López, D. (2008a) “A new approach for car like robots manoeuvring based on RRT,” Robótica: Automaçao, Controlo e Instrumentaçao. Núm. 71. Pag. 10-14.

[38]. Gómez-Bravo, F., López, D., Macías, J., Martín, J.M., del Toro, M., Fortes, J.C. (2008b) “Diseño de Trayectorias para el Guiado de Vehículos Autónomos en Entornos Hospitalarios”. Todo Hospital. Núm. 252. Pag. 572-577.

[39]. Gomez-Bravo, F., Cuesta, F., Ollero, A., Viguria, L. A. (2008c) “Continuous Curvature Path Generation Based on β-Spline Curves for Parking Manoeuvres”. Robotics and Autonomous Systems. Vol. 56. Núm. 4. 2008. Pag. 360-372.

[40]. Gomez-Bravo, F., López, D., Real, F.J., Merino, L., Sánchez-Matamoros, J.M. (2009) “Integrated Path Planning and Tracking for Autonomous Car-Like Vehicles Maneuvering”. Proceedings 6th International Conference on Informatics in Control, Automation and Robotics (Icinco). Num. 6. (Insticc). Pag. 457-464.

[41]. Grandham, R.P.S.R., M. Dawande and S. Venkatesan (2003) “Energy efficient schemes for wireless sensor networks with multiple base station” in Proceedings of the IEEE Globecom 2003, vol. 1, San Francisco, CA, pp. 377-381.

[42]. Green, P.J., Sibson, R.R. (1978) “Computing Dirichlet tessellations in the plane”. Comput. J., 21:168-173.

[43]. Grewal, M.S., Andrews, A.P. (1993) “Kalman Filtering Theory and Practice”. Prentice-Hall.

[44]. Guibas, L.J., Mitchell, J.S.B., Roos, T. (1992) “Voronoi diagrams of moving points in the plane”. In Proc. 17th Internat. Workshop Graph-Theoret. Concepts Comput. Sci., volume 570 of Lecture Notes Comput. Sci., pages 113-125. Springer-Verlag.

[45]. Halton, J.H. (1960) “On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals”. Numerische Mathematik, 2:84–90.

[46]. Hammersley, J.M. (1960). “Monte-Carlo methods for solving multivariable problems”. Annals of the New York Academy of Science, 86:844–874.

[47]. Hayward, V. (1986) “Fast collision detection schema by recursive decomposition of a manipulator workspace”, Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 1044-1049.

[48]. Herman, M. (1986) “Fast, three-dimensional, collision-free motion planning”, Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 1056-1063.

[49]. Hwang F.K. (1979) “An O(n log n) algorithm for rectilinear minimal spanning tree”. J. ACM, 26:177-182.

[50]. Jackins, C.L., Tanimoto, S.L. (1980) “Octrees and their use in representing three-dimensional objects”, Computer Graphics and Information Processing, 14(3).

[51]. Jünger M., Kaibel, V., Thienel, S. (1993) “Computing Delaunay-Triangulations in Manhattan and Maximum Metric”. Technical report, Universität zu Köln.

[52]. Kansal, A., Somasundara A. A., Jea, D. D., Srivasvata, M. B. y Estrin, D., (2004) “Intelligent fluid infrastructure for embedded networks.” In Proceedings of the 2nd ACM/ SIGMOBILE International Conference on Mobile Systems, Applications, and Services, MobySys, Boston, MA, June 6–9 2004, pp.111–124.

[53]. Kao, T.C., Mount, D.M. (1991) “An algorithm for computing compacted Voronoi diagrams defined by convex distance functions”. In Proc. 3rd Canad. Conf. Comput. Geom., pages 104-109.

[54]. Kavraki, L.E., Kolountzakis, M.N., Latombe, J-C. (1996a) “Analysis of Probabilistic roadmaps for path planning” IEEE Int’l Conf. on Robotics and Automation.

[55]. Kavraki, L.E., Svestka, P., Latombe, J-C., Overmars, M.H. (1996b) “Probabilistic Roadmaps for path planning in high-dimensional configuration spaces”. IEEE Trans. on Robotics and Automation, n12.

[56]. Kenneth, E.H., Culver, T., Keyser, J., Lin, M., Manocha, D. (1999) “Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware”.

Proceedings of the 26th annual conference on Computer graphics and interactive techniques. Pages: 277 – 286.

[57]. Khatib, O. (1980) “Commande Dynamique dans l’Espace Opérationnel des Robots Manipulateurs en Présence d’Obstacles”. Thesis Dissertation. Ecole Nationale Supérieure de l’Aéronautique et de l’Espace. Toulouse.

[58]. Kim J.O., Khosla, P. (1991) “Real time obstacle avoidance using harmonic potential functions”. Proc. IEEE Int’l Conference on Robotics and Automation.

[59]. Kirkpatrick, D.G. (1979) “Efficient computation of continuous skeletons”. In Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci., pages 18-27.

[60]. Klein R., Mehlhorn, K., Meiser, S. (1993) “Randomized incremental construction of abstract Voronoi diagrams”. Comput. Geom. Theory Appl., (3):157-184.

[61]. Kuffner, J.J., LaValle, S.M. (2000) “RRT-connect: An efficient approach to single-query path planning”. In Proc. IEEE Int'l Conf. on Robotics and Automation, pages 995-1001.

[62]. Latombe, J C. (1991) "Robot Motion Planning", Kluwer Academic Pulisher.

[63]. Laumond, J.P., Jacobs, P.E., Taix, M., Murray, M. (1994). “A Motion Planner for Nonholonomic Mobile Robots”. *IEEE Trans. on Robotics and Autom*, Vol 10, No 5: 577-593.

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

[65]. LaValle, S.M., Kuffner, J.J. (1999) “Randomized kinodynamic planning”. In Proc. IEEE Int'l Conf. on Robotics and Automation, pages 473-479.

[66]. LaValle, S.M., Kuffner, J.J. (2000) “Rapidly-Exploring Random Trees: Progress and Prospects”. Proc of the Workshop on the Algorithmic Foundation of Robotics.

[67]. LaValle, S.M. (2006) “Planning algorithms” Cambridge University Press

[68]. Lee, D.T. (1982) “Medial axis transformation of a planar shape”. IEEE Trans. Pattern Anal. Mach. Intell., PAMI-4(4):363-369.

[69]. Lee, D.T. (1980a) “Two-dimensional Voronoi diagrams in the Lp-metric”. J. ACM, 27(4):604-618.

[70]. Lee, D.T., Wong, C.K. (1980b) “Voronoi diagrams in L1 metrics with 2-dimensional storage applications”. SIAM J. Comput., 9(1):200-211.

[71]. Li, Q., DeRosaand, M. y Rus, D. (2003), “Distributed algorithms for guiding navigation across a sensornetwork,”. URL:citeseer.ist.psu.edu/li03distributed .html

[72]. Liotta, G., Preparata, F.P., Tamassia, R. (1998) “Robust Proximity Queries: An illustration of degree-driven algorithm design”. SIAM Journal on Computing, 28,3, 864-889.

[73]. López, D., Gómez-Bravo, F., Cuesta, F., Ollero, A. (2004) “Aplicación del algoritmo RRT a la planificación de movimientos en robots no holónomos”. XXV Jornadas de Automática

[74]. López, D., Gómez-Bravo, F., Cuesta, F., Ollero, A. (2006) “Aplicación del algoritmo RRT a la planificación de movimientos en robots no holónomos”. Revista Iberoamericana de Automática e Informática Industrial (RIAI).Vol.: 3 pp: 56-67.

[75]. López, D., Gómez-Bravo, F. (2007) “Comparativa entre planificadores de trayectorias para su uso combinado en la generación de maniobras” XXVIII Jornadas de automática.

[76]. Martin, J.M., Blanco, A., Requena, I., Fajardo, W. (2003) “Constructing Linguistic Versions For The Multicriteria Decision Support Systems Preference Ranking Organization Method For Enrichment Evaluation I And II”. International Journal Of Intelligent Systems, 18, 711-731.

[77]. Martin, J.M., Blanco, A., Delgado, M., Polo, M.A. (2001) “Fuzzy Methods of Multicriteria Decision. The Methods Fpromethee(2t) I and II”. World Multiconference On Systemics, Cybernetics And Informatics, pp 93-98.

[78]. Martin, J.M., Blanco, A., Delgado, M., Polo, M.A. (2003) “Multicriteria Decisions With Qualitative Information”. Intelligent Control Systems And Signal Processing: (ICONS 2003). Proceedings volume from the IFAC International Conference : pp 75-82.

[79]. Martin, J.M., López, D., Gómez Bravo, F., Blanco, A. (2010) “Application of Multicriteria Decision-Making Techniques to Manoeuvre Planning in Nonholonomic Robots”. Expert Systems With Applications. Pag. 3962-3976.

[80]. Maurer, C.R., Raghavan, V. (2003) “A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions” IEEE Trans. on Pattern analysis and machine intelligence, vol 25. nº2.

[81]. McAllister, M., Kirkpatrick, D., Snoeyink, J. (1996) “A compact piecewise-linear Voronoi diagram for convex sites in the plane”. Discrete Comput. Geom., 15:73-105.

[82]. Moore, K. L., Chen, Y. y Song, Z., (2004) “Diffusion-based path planning in mobile actuator-sensor networks (mas-net): Some preliminary results.” in Proceedings of SPIE, April.

[83]. Muñoz, V. (1995). “*Planificación de Trayectorias para Robots Móviles*”. Tesis Doctoral. Universidad de Málaga.

[84]. Muñoz, V., García-Cerezo, A. Cruz, A. (1999) “A Mobile Robots Trajectory Planning Approach under Motion Restrictions”. Integrated Computer-Aided Engineering, 6(4), 331-347. IOS Press Amsterdam.

[85]. Nader, M. Al-Jaroodi, J. Jawhar, I. (2008) “Middleware For Robotics: A Survey”, Proc. of the IEEE Intl. Conf. on Robotics, Automation and Mechatronics, pp. 736-742. Sept. 2008 (RAM 2008).

[86]. Nilsson, N.J. (1969) “A Mobile Automaton: An application of artificial intelligence techniques”. Proceedings of the 1st International Joint Conference on Artificial Intelligence, Washington D.C., 509-520.

[87]. Nilsson, N.J. (1980) “Principles of Artificial Intelligence”, Morgan Kaufmann, Los Altos, CA.

[88]. Ohya, T., Iri M., Murota, K. (1984) “Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms”. J.Oper. Res. Soc. Japan, 27(4):306-336.

[89]. Ollero, A. (2001). “Robótica: manipuladores y Robots Móviles”. Marcombo Boixareu.

[90]. Ollero, A., Arrue, B.C., Ferruz, J., Heredia, G., Cuesta, F., Lopez-Pichaco, F., Nogales, C. (1999). “Control and perception componenets for autonomous vehicle guidance. application to the romeo vehicles”. Control Eng. Practice, pages 1291–1299.

[91]. Overmars, M.H., Welzl, E. (1988) “New Methods for visibility graphs”. Symposium on Computational Geometry.

[92]. Papadopoulou, E., Lee, D.T. (2001) “The L_∞ Voronoi Diagram of segments and VLSI applications”. Int. Journal of Computational Geometry & Applications 11(5):Pag. 503-528.

[93]. Ranganathan, A. (2003) “Quantitatively evaluating performance of RRTs”, CS-7630 Project Report.

[94]. Reeds, J.A., Shepp, L.A. (1990) “Optimal paths for a car that goes both forward and backward”. Pacific Journal of Mathematics. Vol. 145. Nº2.

[95]. Rimon, E., Koditschek, D.E. (1992) “Exact robot navigation using artificial potential functions”. IEEE Trans. on Robotics and Automation.

[96]. Roos, T. (1993) “Tighter bounds on Voronoi diagrams of moving points”. In Proc. 5th Canad. Conf. Comput. Geom., pages 358-363.

[97]. Samet, H. (1980) “Region representation: quadtrees from boundary codes”, Communications of the ACM, 23(3), 163-170.

[98]. Sethian, J.A. (1996) “A fast marching level set method for monotonically advancing fronts”. Proc. Natl. Acad. Sci. USA Vol. 93, pp. 1591-1595. Applied Mathematics.

[99]. Shah, S. J. R. C., Roy, S. y Brunette, W., (2003) “Data MULEs: Modeling a three-tier architecture for sparse sensor networks.” In Proceedings of the First IEEE

International Workshop on Sensor Network Protocols and Applications, SNPA, Anchorage, AK, May 11 2003, pp. 30–41.

[100]. Shamos, M.I., Hoey, D. (1975) “Closest-point problems”. In Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci., pages 151-162.

[101]. Spivak, M. (1979) “A comprehensive introduction to differential geometry”, Publish or Perish, Wilmington D. E.

[102]. Stappen, A.F., Overmars, M.A., de Berg, M.T., Vleugels, J.M. (1997) “Motion planning in enviroments with low obstacle density”. Technical Report UU-CS-1997-19.Department of Information and Computing Sciences, Utrecht University.

[103]. Sugihara, K., Iri, M. (1992) “Construction of the Voronoi diagram for `one million' generators in single-precision arithmetic”. Proc. IEEE, 80(9):1471-1484.

[104]. Sukharev, A.G. (1971) “Optimal strategies of the search for an extremum”. U.S.S.R. Computational Mathematics and Mathematical Physics, 11(4), Translated from Russian, Zh. Vychisl. Mat. i Mat. Fiz., 11, 4, 910-924.

[105]. Tong, Q. Z. L. y Adireddy, S., (2003) “Sensor networks with mobile agents.” in Proceedings of the IEEE Military Communication Conference, MILCOM 2003, vol. 1, Boston, MA, October 13–16, pp. 705–710.

[106]. Van der Corput, J.G. (1935). “Verteilungsfunktionen I”. Akademie van Wetenschappen, 38:813–821.

[107]. Venkitasubramaniam, S. A. P. y Tong, L. (2004) “Sensor networks with mobile agents: Optimal random access and coding.” IEEE Journal on Selected Areas in Communications, vol. 22, no. 6, pp. 1058– 1068, August.

[108]. Vleugels, J., Overmars, M. (1995) “Approximating Generalized Voronoi Diagrams in Any Dimension” Technical Report UU-CS-1995-14, Utrech University.

[109]. Vincke, P. (1992) “Multicriterio decision aid”. John Wiley & Sons.

[110]. Volpe, R., Khosla, P. (1987) “Artificial potentials with elliptical isopotential contours for obstacle avoidance”. Proc. IEEE Int’l Conf. on Robotics and Automation.

[111]. Wang, E. M. Z. M., Basagni, S. y Petrioli, C. (2005) “Exploiting sink mobility for maximizing sensor networks lifetime.” In Proceedings of the 38th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 3–6.

[112]. Wilmarth, S.A., Amato N.M., Stiller, P.F. (1998) “A probabilistic roadmap planner with Sampling on the medial Axis of the Free Space”. TR-Department of Computer Science Texas A&M University.

[113]. Xiaoshan, P. (2005) "Wheelchair Robotic Motion Planning" Civil and Enviromental Engineering, Standford University.

[114]. Yap, C.K. (1987) “An O(nlogn) algorithm for the Voronoi diagram of a set of simple curve segments”. Discrete Comput. Geom., 2:365-393.

[115]. Yves, L., Pictet, J., Simos, J. (1994) “Méthodes multicritère ELECTRE”. Presses Polytechniques et Universitaires Romandes

[116]. Zadeh, L.A. (1975). “The concept of a linguistic variable and its application to approximate reasoning”, Information Sciences, 8, 199-249(I), 301-357(II).